

Universidades de Burgos, León y  
Valladolid

Máster universitario

## **Inteligencia de Negocio y Big Data en Entornos Seguros**



**TFM del Máster Inteligencia de Negocio  
y Big Data en Entornos Seguros**

**Estudio de la rotura del hormigón**

Presentado por Víctor Pérez Esteban  
en Universidad de Burgos — 1 de septiembre  
de 2025

Tutor: Pedro Latorre Carmona





# Universidades de Burgos, León y Valladolid



## Máster universitario en Inteligencia de Negocio y Big Data en Entornos Seguros

D. Pedro Latorre Carmona, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Víctor Pérez Esteban, con DNI 71279579A, ha realizado el Trabajo final de Máster en Inteligencia de Negocio y Big Data en Entornos Seguros titulado título de TFM.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 1 de septiembre de 2025

Vº. Bº. del Tutor:

D. Pedro Latorre Carmona

Vº. Bº. del co-tutor:

D. Álvar Arnaiz González





## **Resumen**

La utilización de materiales en la construcción de estructuras hace necesario que se puedan realizar controles que permitan conocer y garantizar el desempeño de estos materiales. Por tanto, se hace imprescindible la realización de métodos que cuantifiquen sus propiedades y otorguen una medida sobre los límites de estos materiales. De esta forma, se puede conocer y planificar su uso para prevenir posibles fallos que deriven en consecuencias potencialmente catastróficas, en concreto su resistencia a la ruptura y la aparición de grietas que debiliten la estructura.

El problema para la realización de estas medidas surge en que los métodos clásicos utilizados son de un carácter manual o requieren de equipo de sensores que es generalmente muy costoso. El problema de los métodos manuales reside en que dependen del conocimiento y la experiencia del especialista que lleva a cabo el proceso. Mientras que los métodos que usan sensores, cuentan con el problema del coste y de que no son capaces de seguir la trayectoria en la generación de las grietas.

En este trabajo se va a utilizar el tercer método, consistente en el análisis de imágenes. En concreto, se han utilizado pares de imágenes que junto con modelos de inteligencia artificial y de análisis de los resultados obtenidos, sean capaces de detectar las grietas que se van formando para poder estudiar su evolución.

## **Descriptores**

Hormigón, Grieta, 3D, IA, Visión Estéreo ...

## Abstract

The use of materials in the construction of structures makes it necessary to carry out inspections that allow us to understand and guarantee the performance of these materials. Therefore, it is essential to implement methods that quantify their properties and provide a measure of their limits. In this way, their use can be understood and planned to prevent possible failures that could lead to potentially catastrophic consequences, in particular their resistance to fracture and the appearance of cracks that weaken the structure.

The challenge in performing these measurements arises because the classic methods used are either manual in nature or require sensor equipment that is generally very costly. The drawback of manual methods lies in their dependence on the knowledge and experience of the specialist conducting the process, whereas sensor-based methods suffer from high costs and are unable to track the path of crack development.

In this work, the third method—image analysis—will be employed. Specifically, pairs of images have been used which, together with artificial intelligence models and analysis of the results obtained, are capable of detecting the cracks as they form in order to study their evolution.

## Keywords

Concrete, Crack, 3D, AI, Stereo Vision

---

# Índice general

---

|   |     |
|---|-----|
| Índice general                              | iii |
| Índice de figuras                           | v   |
| 1. Introducción                             | 3   |
| Memoria                                     | 3   |
| 2. Objetivos del proyecto                   | 5   |
| 3. Conceptos teóricos                       | 7   |
| 3.1. Rectificado de imágenes . . . . .      | 7   |
| 3.2. FoundationStereo . . . . .             | 12  |
| 3.3. DEFOM-Stereo . . . . .                 | 20  |
| 3.4. VGGT . . . . .                         | 27  |
| 3.5. Detectores Esquinas . . . . .          | 31  |
| 3.6. Detectores de líneas . . . . .         | 45  |
| 3.7. Detección de la profundidad . . . . .  | 47  |
| 4. Técnicas y herramientas                  | 57  |
| 4.1. MatLab . . . . .                       | 57  |
| 4.2. Python y Conda . . . . .               | 57  |
| 4.3. PyQt5 y VTK . . . . .                  | 58  |
| 7. Conclusiones y Líneas de trabajo futuras | 59  |
| 7.1. Conclusiones . . . . .                 | 59  |
| 7.2. Líneas de trabajo futuras . . . . .    | 60  |

|  |           |
|--|-----------|
| <b>Apéndices</b>   | <b>63</b> |
| <b>Apéndice A Documentación técnica de programación</b>          | <b>65</b> |
| A.1. Introducción . . . . .                                      | 65        |
| A.2. Estructura de directorios . . . . .                         | 65        |
| A.3. Manual del programador . . . . .                            | 66        |
| A.4. Compilación, instalación y ejecución del proyecto . . . . . | 66        |
| <b>Apéndice B Documentación de usuario</b>                       | <b>69</b> |
| B.1. Introducción . . . . .                                      | 69        |
| B.2. Requisitos de usuarios . . . . .                            | 69        |
| B.3. Instalación . . . . .                                       | 69        |
| B.4. Manual del usuario . . . . .                                | 70        |
| <b>Bibliografía</b>  | <b>73</b> |

---

# Índice de figuras

---

|   |    |
|---|----|
| 3.1. Plano epipolar [5] . . . . .   | 9  |
| 3.2. Líneas epipolares [5] . . . . .  | 9  |
| 3.3. La homografía correspondiente al plano II [5] . . . . .  | 11 |
| 3.4. Esquema del modelo de FoundationStereo [20] . . . . .  | 12 |
| 3.5. Visualización de resultados de FoundationStereo al inicio del experimento . . . . .              | 18 |
| 3.6. Visualización de resultados de FoundationStereo en un punto intermedio del experimento . . . . . | 19 |
| 3.7. Visualización de resultados de FoundationStereo al final del experimento . . . . .               | 20 |
| 3.8. Esquema del modelo de DEFOM Stereo [6] . . . . .   | 21 |
| 3.9. Visualización de resultados de DEFOM Stereo al inicio del experimento . . . . .                  | 25 |
| 3.10. Visualización de resultados de DEFOM Stereo en un punto intermedio del experimento . . . . .    | 26 |
| 3.11. Visualización de resultados de DEFOM Stereo al final del experimento . . . . .                  | 27 |
| 3.12. Esquema del modelo de VGGT Stereo [19] . . . . .  | 28 |
| 3.13. Visualización de resultados de VGGT al inicio del experimento . . . . .                         | 30 |
| 3.14. Visualización de resultados de VGGT en un punto intermedio del experimento . . . . .            | 31 |
| 3.15. Visualización de resultados de VGGT al final del experimento . . . . .                          | 31 |
| 3.16. Visualización de resultados del detector de Harris . . . . .                                    | 33 |
| 3.17. Visualización de resultados del detector de Shi-Tomasi . . . . .                                | 35 |
| 3.18. Visualización de resultados del detector de FAST . . . . .                                      | 37 |
| 3.19. Visualización de resultados del detector de ORB . . . . .                                       | 39 |
| 3.20. Visualización de resultados del detector de SuperPoint . . . . .                                | 41 |

|   |    |
|---|----|
| 3.21. Visualización de resultados del detector de D2-Net . . . . .  | 42 |
| 3.22. Visualización de resultados del detector de R2D2 . . . . .  | 44 |
| 3.23. Visualización de resultados del detector Probabilistic Hough . .  | 47 |
| 3.24. Máscara obtenida usando el gradiente de Sobel. Los puntos<br>blancos son zonas donde se habría detectado zonas de grieta, y<br>los negros el resto de puntos. . . . . | 49 |
| 3.25. Máscara obtenida usando el gradiente de Sobel para la primera<br>imagen sin grieta, que se usará como referencia para ROI. . . . .                                    | 51 |
| 3.26. Zona seleccionada como ROI. . . . .   | 52 |
| 3.27. Zonas seleccionadas como grietas. . . . .   | 53 |
| 3.28. Distancias entre los puntos indicando el tamaño de las grietas<br>detectadas. . . . .   | 55 |
| 7.29. Par de imágenes estéreo generadas en Unity, con un patrón<br>aleatorio, para simular un bloque de hormigón . . . . .  | 61 |
| 7.30. Resultados de profundidad obtenidos al aplicar Foundation Stereo<br>en el par generado con Unity . . . . .  | 61 |
| B.1. Pantalla inicial . . . . .   | 70 |
| B.2. Pantalla con los resultados . . . . .  | 71 |

# **Memoria**



---

# Introducción

---

El estudio de los materiales de construcción como el hormigón, que se emplea en todo tipo de estructuras y edificaciones como puentes, túneles y distintos tipos de edificios, es una parte fundamental para conocer los límites y el comportamiento de los mismos. En el caso concreto del hormigón, características como su durabilidad o su resistencia a los impactos ha hecho que sea ampliamente usado como material de construcción. Pero otra de las propiedades de este material es su fragilidad, lo que le hace propenso a que aparezcan grietas debido a distintos factores como pueden ser las variaciones de temperatura, la humedad o la presión a la que se es sometido en las distintas estructuras de las que forma parte. [21] Por ello, es necesario la creación de métodos que permitan medir sus propiedades para poder predecir el comportamiento del mismo, adecuar las construcciones a estas propiedades, y que estos métodos permitan la revisión y detección de posibles problemas a lo largo de la vida de las construcciones.

Los tres tipos de métodos principales de medición y análisis usados han sido [11] la medición manual, la medición a través de distintos tipos de sensores, y el análisis de imágenes.

Los métodos manuales consisten en que un técnico especialista revise de forma visual las estructuras en búsqueda de la posible aparición de grietas. Hacen uso de distintos tipos de patrones y esquemas para poder analizar el posible origen de la grieta y cómo afecta al conjunto de la estructura, apoyándose en mediciones sobre la anchura de las grietas. La principal ventaja de este método es su bajo coste y su simplicidad, con lo que han sido los más utilizados de forma tradicional para realizar las tareas de mantenimiento de las estructuras de hormigón. Y su mayor desventaja es que son mediciones subjetivas que dependen del conocimiento y la experiencia del técnico que las realiza.

Las mediciones a través de sensores se utilizan para solventar este problema. Hay diversos tipos de sensores y de tipos de mediciones como ultrasonidos [9], medidores de tensión, sensores acústicos, de infrarrojos... Todos estos conjuntos de sensores permiten realizar medidas objetivas para analizar la aparición de las grietas y analizar el estado actual de los materiales. Pero las desventajas que presentan es su elevado coste y que sólo pueden realizar las medidas en puntos concretos, lo que hace difícil usarlos para estudiar la evolución de las grietas mientras estas se producen.

El método de análisis de imágenes intenta solventar los problemas de coste y de seguimiento de únicamente de puntos concretos de los sensores. Para ello, se utilizan diferentes técnicas de análisis de imágenes [7] como la transformada de wavelet, la correlación digital de imágenes (DIC), métodos de percolación, detección de bordes Canny, bosques aleatorios, redes neuronales convolucionales... Estos métodos utilizan una serie de imágenes capturadas a lo largo del tiempo para analizar un conjunto de puntos y su desplazamiento a lo largo del tiempo de la captura. Además de ser más flexibles en cuanto a su colocación para la realización de las capturas de imágenes que los sensores, también son menos invasivos debido a que esa captura se puede realizar sin tener que alterar el material sobre el que se está realizando la medición.

En este trabajo se ha utilizado este último método. Se ha contado con un conjunto de pares de imágenes capturadas mientras se sometía a un bloque de hormigón a una prueba de estrés por presión. Estos pares de imágenes correspondían a dos cámaras de tal forma que fuera posible a través de la diferencia de perspectiva, usar la visión estereoscópica para realizar una representación en tres dimensiones de la escena. Para este proceso se han utilizado modelos basados en inteligencia artificial, que nos han permitido obtener la posición en el espacio de los píxeles de las imágenes. Y a partir de esa posición, conocer su profundidad y combinar estas dos medidas para poder detectar donde se ha producido una grieta y medir su tamaño.

---

## **Objetivos del proyecto**

---

Los objetivos del proyecto son:

1. A partir de pares de imágenes, obtener una representación 3D del bloque de hormigón.
2. Una vez obtenida una representación en el espacio tridimensional, analizar si las grietas son detectadas y representadas en el modelo en 3D.
3. A partir del modelo 3D con la grietas, poder realizar una medición del tamaño de las mismas.
4. Crear un método automatizado para la medida de las grietas que permita el estudio de su evolución.

Para ello se han probado distintos modelos de inteligencia artifical, que nos permiten obtener una posición en un espacio tridimensional de los píxeles de las imágenes obtenidas. Y usando los datos de esas representaciones, como las posiciones de los puntos y su profundidad, poder detectar dónde se ha producido una grieta y cuál es su tamaño.



---

# **Conceptos teóricos**

---

En esta sección vamos a explicar los modelos y técnicas empleados para el desarrollo del proyecto y el análisis de las imágenes.

## **3.1. Rectificado de imágenes**

Para la obtención del par de imágenes se han usado dos cámaras que están paralelas la una a la otra, tomando una imagen de la escena cada una al mismo tiempo.

Para poder hacer el análisis de las imágenes, necesitamos saber que puntos o píxeles de una imagen se corresponden con los de la otra imagen.

Debido a las variaciones que pueda haber a la hora de tomar las imágenes en cada una de las cámaras, además de la perspectiva de cada una de ellas, esa correlación no es directa, con lo que hace falta un proceso de búsqueda de esos puntos.

Y para facilitar esa búsqueda, se realiza el proceso de rectificación de imágenes.

### **Parámetros intrínsecos**

Los parámetros intrínsecos de cada cámara corresponden a los valores que definen las características de la cámara que toma la fotografía.

Estos son:

1.  $f$ : la longitud focal

2.  $\gamma$ : la relación de aspecto
3.  $s$ : la desviación de los ejes de la cámara
4.  $x_0, y_0$  : el punto principal, que es la proyección ortogonal del punto focal de la cámara en la imagen.

Estos parámetros definen cómo captura la cámara la imagen, y son necesarios para calcular la matriz  $K$  o matriz de parámetros intrínsecos, que es necesaria para poder determinar cómo se debe rotar y reescalar la imagen para asegurar que las líneas epipolares queden horizontales.

$$K = \begin{pmatrix} \gamma f & sf & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$

Estos parámetros se obtienen a través de un proceso de calibración de las cámaras usando un patrón conocido, como por ejemplo, un patrón de tablero de ajedrez.

### Líneas epipolares

Las líneas epipolares son las líneas que son el resultado de la intersección del plano epipolar, formado por el punto del mundo real y los puntos correspondientes a los puntos focales de las cámaras, con la imagen captada por las cámaras.

La propiedad fundamental de estas líneas es que el punto del mundo real estará representado en estas dos líneas, lo que nos va a permitir hacer una correspondencia entre puntos de una y otra imagen.

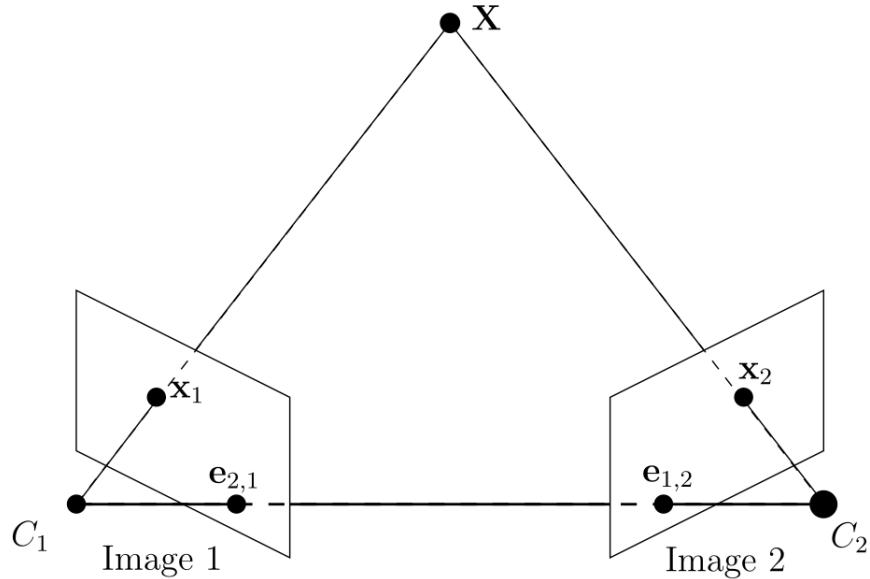


Figura 3.1: Plano epipolar [5]

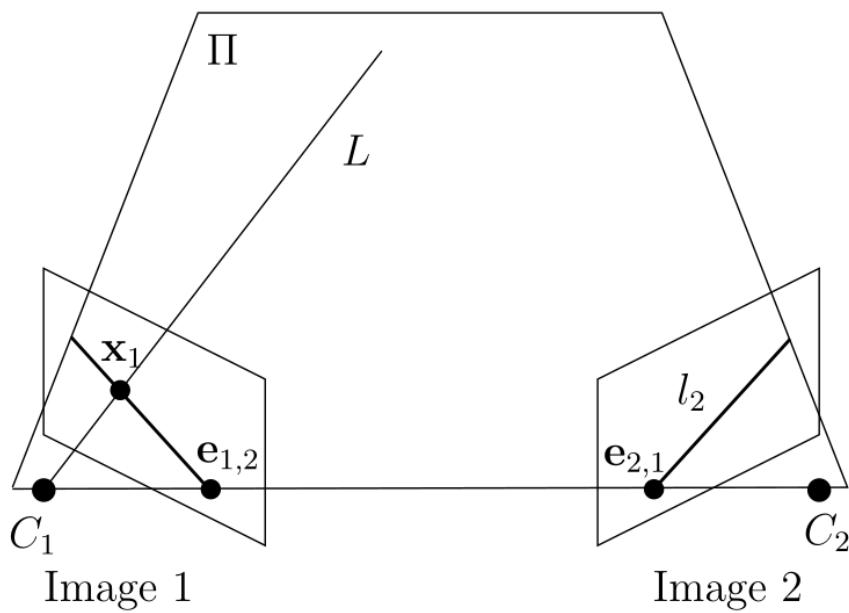


Figura 3.2: Líneas epipolares [5]

## Parámetros extrínsecos

Los parámetros extrínsecos son los que van a describir la posición y la orientación de las cámaras en el mundo real. En concreto, están formados por una matriz  $R$  que va a definir hacia donde mira la cámara, y un vector  $T$  que va a indicar en qué posición se encuentra la cámara.

Estos valores nos van a permitir alinear las vistas de las cámaras y poder realizar la rectificación de las imágenes.

## Homografías

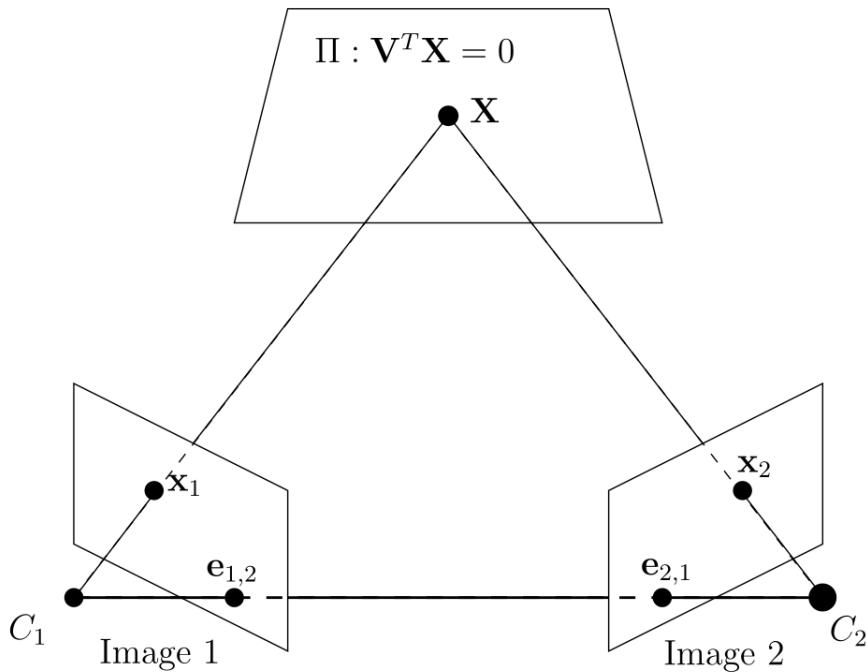
Las homografías son transformaciones que convierten puntos de un plano en un espacio tridimensional, en esos mismos puntos en otro plano del mismo espacio tridimensional. Esto sirve, por ejemplo, para tener dos vistas distintas dentro de ese espacio.

En el caso de la rectificación de imágenes, nos van a servir para que las líneas epipolares pasen a ser filas horizontales, lo que va a facilitar la búsqueda y relación de puntos entre el par de imágenes captados por las cámaras.

$$H_1 = K_{\text{new}} R_{\text{rect}} R K_1^{-1}, \quad (3.1)$$

$$H_2 = K_{\text{new}} R_{\text{rect}} R K_2^{-1}. \quad (3.2)$$

Donde  $K_{\text{new}}$  es la matriz intrínseca promediada entre las dos cámaras, y  $R_{\text{rect}}$  es una matriz de rotación que deja las líneas epipolares paralelas alineando el vector  $T$  con el eje x.

Figura 3.3: La homografía correspondiente al plano  $\Pi$  [5]

## Proceso de rectificación

El proceso de rectificación de imágenes consiste en alinear las líneas epipolares de cada imagen entre sí, de tal forma que la búsqueda de puntos comunes en ambas imágenes se simplifique a la búsqueda de esos puntos en una sola línea o fila de la imagen, en vez de en toda la imagen a través de todas sus filas. Se pasa de una búsqueda en dos dimensiones a una búsqueda en una sola dimensión, simplificando enormemente el proceso.

Con todos los elementos obtenidos y calculados anteriormente, se pueden obtener un par de imágenes rectificadas a partir de aplicar un warping proyectivo a las imágenes captadas por las cámaras con las homografías correspondientes a cada imagen.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = H^{-1} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \implies x = \frac{h_{11}x' + h_{12}y' + h_{13}}{h_{31}x' + h_{32}y' + h_{33}}, \quad y = \frac{h_{21}x' + h_{22}y' + h_{23}}{h_{31}x' + h_{32}y' + h_{33}}.$$

### 3.2. FoundationStereo

FoundationStereo es un modelo desarrollado por nVidia que permite que a partir de un par de imágenes de una escena, representar cada píxel de la imagen en un espacio en tres dimensiones. Es un modelo generalista o "zero-shot", que quiere decir que se ha diseñado para ser capaz de analizar cualquier tipo de par de imágenes sin necesitar ajustes específicos para obtener los resultados esperados.

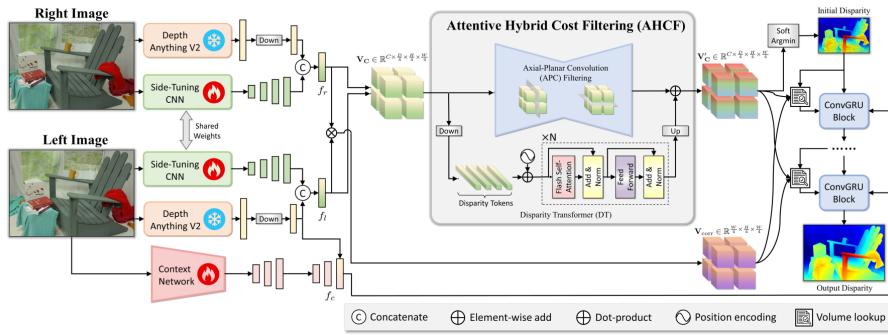


Figura 3.4: Esquema del modelo de FoundationStereo [20]

### Problemas para crear un modelo generalista

En otros campos de la inteligencia artificial, se ha tenido un gran avance en los últimos años a la hora de crear modelos generalistas que proporcionen grandes resultados. Pero en el caso concreto de los modelos de visión estéreo, tienen problemas concretos que han supuesto grandes dificultades a solventar.

Uno de los mayores problemas es que estos modelos se entran con un conjunto de datos limitados (como SceneFlow o KITTI), que aunque resultan muy útiles para el entrenamiento de los modelos, no proporcionan la diversidad suficiente de escenas y condiciones que pueden aparecer en un entorno real. Esto provoca que los modelos aprendan a solventar estos casos concretos, pero fallen cuando se enfrentan a condiciones diferentes que no se parezcan a los de estos conjuntos de datos.

Para solventar este problema, se ha recurrido al entrenamiento con conjuntos de datos sintéticos. Esto proporciona la ventaja de un menor coste a la hora de generar estos conjuntos de datos, que además proporcionan un ground-truth perfecto (unas medidas de la respuesta correcta perfecta). El problema de este tipo de conjunto de datos, es que fallan a la hora de

reflejar las condiciones reales en las que se toman las imágenes, como pueden ser el ruido de los sensores, las variaciones de las ópticas, la iluminación, condiciones atmosféricas... Con lo que cuando se enfrentan estos modelos a imágenes con las condiciones del mundo real bajan su rendimiento.

Todos estos inconvenientes ha llevado a que se creen modelos que están especializados en algún dominio concreto, pero que no ofrecen los resultados esperados cuando se aplican a escenarios generalistas. Y por este motivo surgió FoundationStereo, como un modelo para solventar este problema de generalización.

## Cómo funciona FoundationStereo

FoundationStereo utiliza varios pasos para obtener la mayor información posible de las imágenes y así realizar una predicción lo más precisa posible.

### Primera fase

La primera fase consiste en varios procesamientos paralelos de las imágenes a través de modelos preentrenados diferentes. Por un lado, se procesan las imágenes con Depth Anything y EdgeNeXt-S para obtener información de las imágenes en distintos desplazamientos, que se utilizará posteriormente en el cálculo del volumen de coste. Y por otro, se procesan las imágenes para obtener información global de la escena para un proceso llamado Context Network, que servirá para realizar una mejor correspondencia entre las imágenes en las regiones más complicadas.

Depth Anything es un modelo que estima la profundidad a partir de una sola imagen. Este proceso se utiliza, no para obtener datos sobre la profundidad, si no para obtener un conocimiento general de la imagen que ayude a la predicción de la profundidad posteriormente. Depth Anything ha sido entrenado con una cantidad inmensa de imágenes del mundo real, de tal forma que ayuda al modelo de Foundation Stereo a obtener datos sobre las regiones más ambigüas. Esto lo hace a través de darle el conocimiento que ha adquirido este modelo sobre regiones difíciles de diferenciar como paredes lisas, objetos transparentes, sombras y luces o estructuras finas.

Paralelamente, se procesan las imágenes con la fase que denominan SideTuning CNN (con EdgeNeXt-S). Con este proceso obtienen, al contrario que pasaba con Depth Anything, conocimiento de los detalles de las imágenes. Esto es necesario porque los resultados que proporciona Depth Anything provienen de procesar una sola imagen individual, con lo que no están alineados con el plano epipolar para poder relacionar las dos imágenes en

estéreo. Lo que hace el SideTunnig CNN es tomar la información global de la imagen que proporciona Depth Anything, con los detalles locales que proporciona EdgeNeXt-S, como detalles de la disparidad entre las dos imágenes, y las une para que pueda ser utilizada posteriormente con el par estéreo.

Una vez obtenidas las características de las imágenes a partir de la combinación de DepthAnything y la SideTunnig CNN, se construye lo que se denomina un volumen de coste. Este volumen constituye una representación que se utilizará en el proceso de predicción de disparidad, y que almacena para cada posible desplazamiento entre las dos imágenes, una medida de la similitud entre los píxeles correspondientes. De esta forma, en lugar de comparar píxeles de manera individual, el modelo trabaja con un espacio de hipótesis que recoge todas las correspondencias posibles en una estructura multidimensional.

El volumen de coste en FoundationStereo se construye de forma híbrida combinando dos enfoques distintos. Por un lado, se calcula una correlación por grupos (group-wise correlation), en la que se dividen las características en subconjuntos y se mide su similitud a través del producto escalar. Esto proporciona una estimación de como se correlacionan los píxeles del par de imágenes al aplicar un cierto desplazamiento. Por otro lado, se realiza una concatenación de las características de ambas imágenes tras aplicar ese mismo desplazamiento. Este segundo enfoque permite que la información proveniente de DepthAnything no se pierda, quedando reflejada en el propio volumen de coste.

Esto permite que el volumen de coste no solo contenga información sobre las similitudes entre los píxeles, sino también un conocimiento global derivado de la información proporcionada por Depth Anything. De esta forma se mejora el desempeño en regiones problemáticas, como superficies lisas, reflejos o estructuras finas, en las que los métodos tradicionales obtienen peores resultados.

El Context Network se realiza en paralelo al procesado anterior. En este caso se utiliza una CNN de nVidia que extrae características a distintas escalas. Estas características de contexto se utilizarán posteriormente para inicializar el estado interno de los módulos ConvGRU, encargados de refinrar la disparidad. De este modo, el refinamiento no depende únicamente de la similitud entre píxeles, sino que utilizará una representación que añade información sobre el global de la imagen y sus características.

Esto permite solventar algunas de las limitaciones que surgían en los modelos clásicos como en superficies lisas, objetos translúcidos o regiones

con iluminación compleja. Este contexto se une al que ya proporciona el procesado que se realiza con DepthAnything, lo que mejora los resultados cuando se presentan escenas del mundo real con distintas características.

## Segunda fase

La segunda fase se denomina Attentive Hybrid Cost Filtering (AHCF). Esta fase usa los valores proporcionados por el volumen de coste de la fase anterior para obtener una estimación inicial de la disparidad. Para ello, se combinan dos procesos: por un lado, un filtrado convolucional (Axial-Planar Convolution o APC) que maneja la dimensión de disparidad, que son las hipótesis de las predicciones de la disparidad provenientes del volumen de coste. Y por otro lado, un mecanismo de atención (Disparity Transformer o DT) que permite mejorar la predicción comparando todas las dimensiones entre sí buscando información de contexto de la imagen que se encuentra en esas dimensiones.

El primer proceso, Axial-Planar Convolution, está formado por dos pasos. El primer paso se encarga de procesar la altura y la anchura, y el segundo paso se encarga de procesar la disparidad. Con el procesamiento de la altura y la anchura lo que se consigue es que se tenga en cuenta la información de los píxeles vecinos de la imagen. Esto sirve para mejorar la predicción cuando surgen defectos o características en la imagen que podrían afectar al resultado, pero teniendo en cuenta la información de la región y la zona se evitan en gran medida estos problemas. Y con el procesamiento de las disparidad, se tiene en cuenta al mismo tiempo todas las predicciones para cada píxel, de esta forma se puede tener en cuenta la probabilidad de la disparidad calculada de cada dimensión y poder analizar la evolución en cada una de ellas para poder obtener un resultado más preciso.

El segundo proceso, correspondiente a Disparity Transformer, compara todas las predicciones de disparidad de todas las dimensiones entre sí, para poder calcular cuál de todas ellas son las más probables y coherentes teniendo en cuenta el contexto global. De este modo se tiene en cuenta la información que aporta cada dimensión para poder hacer una elección lo más informada posible.

Al combinarse estos dos procesos, se proporciona una estimación inicial de la disparidad mucho más precisa y coherente con la imagen para terminar su predicción en la última fase del proceso.

### Tercera fase

Después de que el volumen de coste ha sido filtrado por el proceso AHCF, se calcula una primera estimación de la disparidad utilizando un proceso llamado Soft-Ergmin. Lo que hace este método es convertir los valores de las probabilidades de la disparidad de cada píxel, de modo que los valores más coherentes tengan más peso. A partir de ahí, se obtiene un valor único que representa la disparidad inicial de cada píxel. Esta primera predicción no es aún la definitiva, pero ya es bastante correcta debido a todo el procesado anterior.

Con esa predicción inicial, se pasa al siguiente proceso que es el refinamiento iterativo, que se realiza mediante unos bloques denominados ConvGRU. Estos bloques funcionan de forma iterativa en un bucle que va corregiendo la predicción. En cada iteración, comparan la disparidad estimada con el volumen de coste original y con las características de contexto de la imagen. Si la predicción no es correcta con todo ello, el modelo la ajusta un poco y se repite este proceso mejorando el resultado anterior.

Los bloques ConvGRU realizan este refinamiento de la predicción de forma progresiva, comenzando en una versión más reducida de la imagen y pasando después a resoluciones mayores. Esto permite que primero se resuelva la estructura general de la escena y, en los últimos pasos, se añadan los detalles finos. Finalmente, la disparidad se reescalía hasta la resolución completa de la imagen.

## De disparidad a profundidad

El resultado final es un mapa de disparidades, es decir, para cada píxel de la imagen izquierda se estima cuántos píxeles se desplaza su correspondencia en la imagen derecha. Sin embargo, lo que se quiere obtener es un mapa de profundidad que indique la distancia desde la cámara hasta cada punto de la imagen.

A partir de la disparidad se puede calcular la profundidad usando la geometría. Si se conocen los parámetros intrínsecos y extrínsecos, entonces la profundidad se puede calcular mediante la siguiente fórmula:

$$Z = \frac{f \cdot B}{d}$$

En esta fórmula, la disparidad  $d$  se mide en píxeles, mientras que  $f$  y  $B$  están en unidades físicas (por ejemplo, milímetros). A mayor disparidad,

menor profundidad. Esto es debido a que los objetos cercanos producen desplazamientos grandes entre la imagen izquierda y la derecha, mientras que los objetos lejanos apenas presentan diferencia de posición.

Cuando se utiliza el modelo de FoundationStereo sin los parámetros intrínsecos y extrínsecos, los resultados de la profundidad al no poder aplicarse esta fórmula, son resultados relativos. Es decir, que su escala de distancias es correcta entre sí, pero los valores no se corresponden con los valores de distancia del mundo real.

## Resultados

Los resultados obtenidos por FoundationStereo, como se pueden ver en la visualización con mapa de calor, son bastante buenos a la hora de detectar la profundidad de la imagen y su estructura general. Un problema que nos vamos a encontrar, es que como se puede observar en la secuencia de imágenes, es que aunque es capaz de detectar que en el espacio donde se encuentra la grieta y proporcionar una diferencia de profundidad, la diferencia con los valores de profundidad del entorno es muy pequeño, lo que va a ser problemático a la hora de detectarlo de forma automatizada.

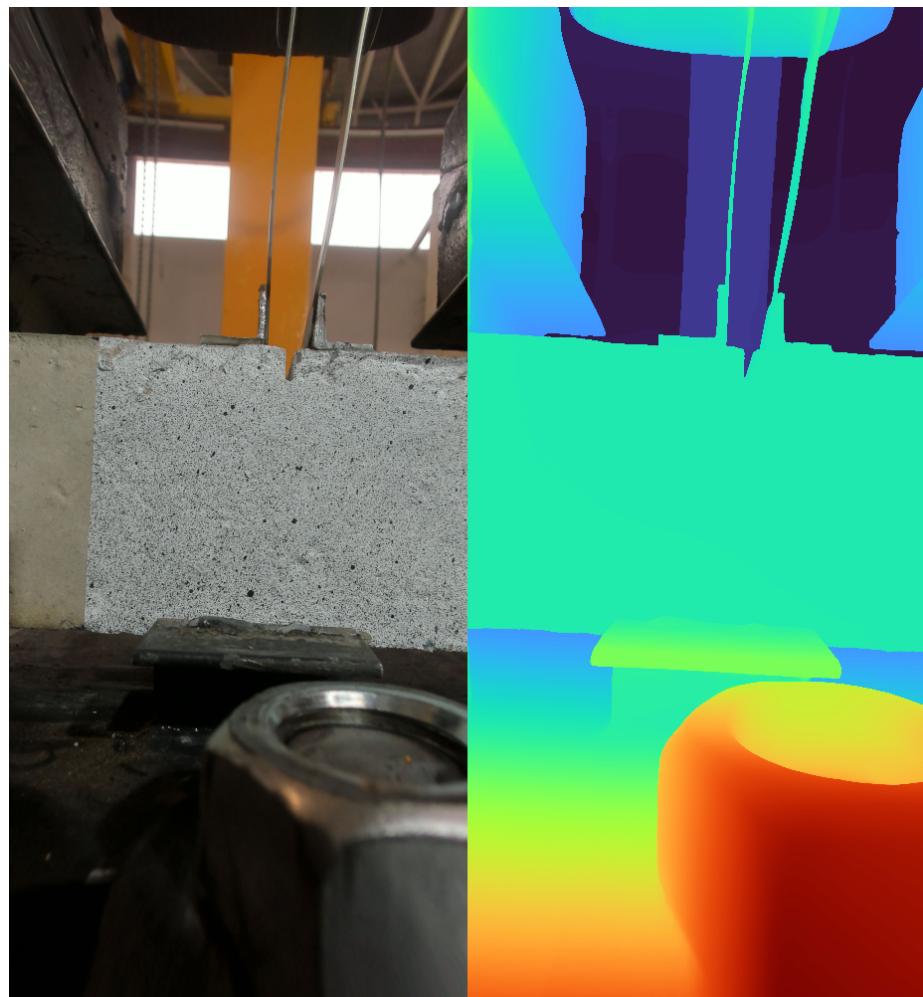


Figura 3.5: Visualización de resultados de FoundationStereo al inicio del experimento

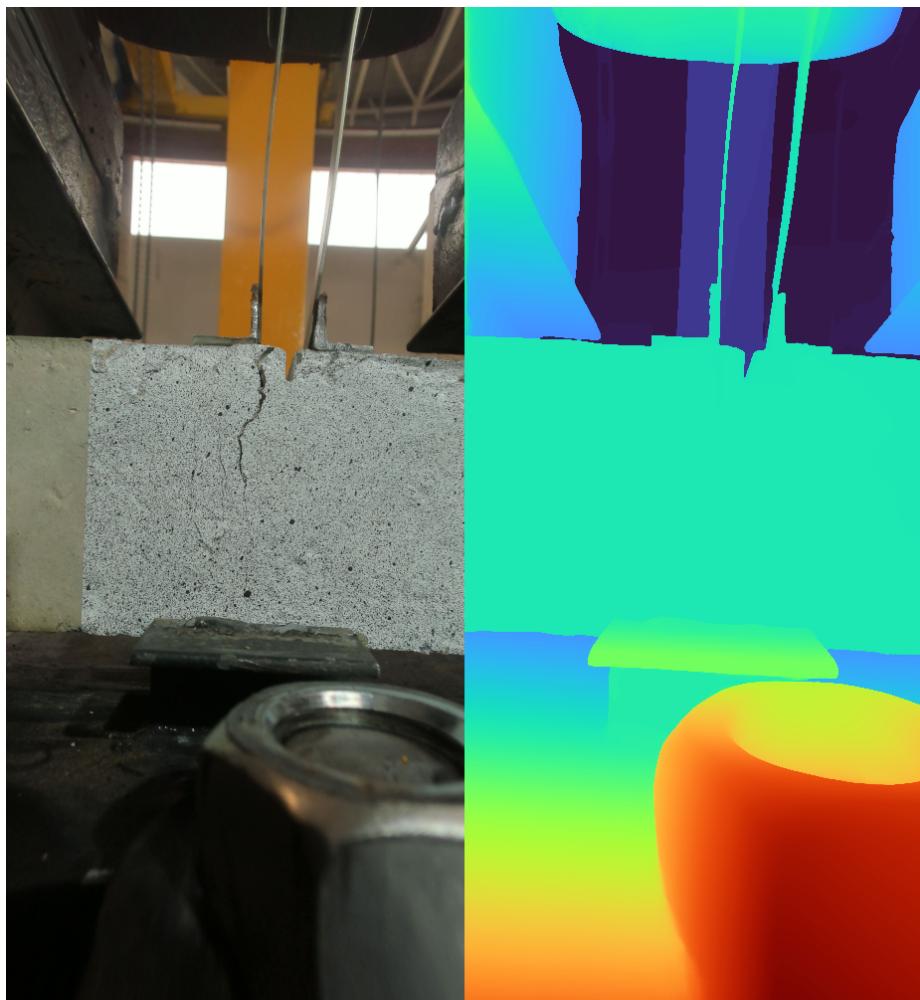


Figura 3.6: Visualización de resultados de FoundationStereo en un punto intermedio del experimento

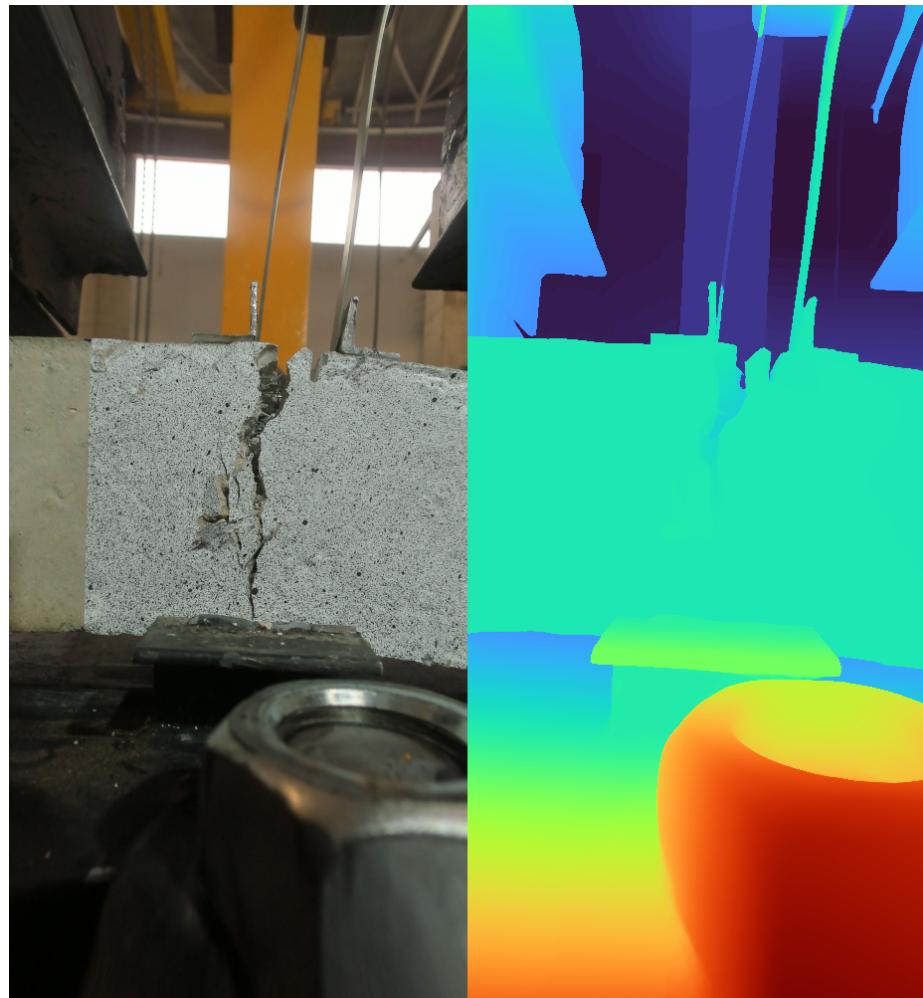


Figura 3.7: Visualización de resultados de FoundationStereo al final del experimento

### 3.3. DEFOM-Stereo

DEFOM-Stereo es un modelo desarrollado por investigadores de Insta360 Research y la Chinese University of Hong Kong que combina las ventajas de la estimación monocular de profundidad con el emparejamiento estéreo. Como en el caso de FoundationStereo, su objetivo es mejorar la obtención de mapas de disparidad en escenas reales, donde problemas como las zonas sin textura, las occlusiones o la iluminación complicada suelen generar que no se obtengan buenos resultados.

Como pasaba con FoundationStereo, va a hacer uso de Depth Anything2 para obtener información del contexto de las imágenes e inicializar el mapa de disparidad. Además, introduce un módulo de actualización de escala que corrige las inconsistencias que aparecen al pasar de profundidad relativa a disparidad métrica, refinando de manera iterativa los resultados hasta recuperar una geometría más precisa.

DEFOM-Stereo busca lograr un modelo generalista o zero-shot, es decir, que pueda aplicarse a escenas nuevas sin necesidad de ajuste específico.

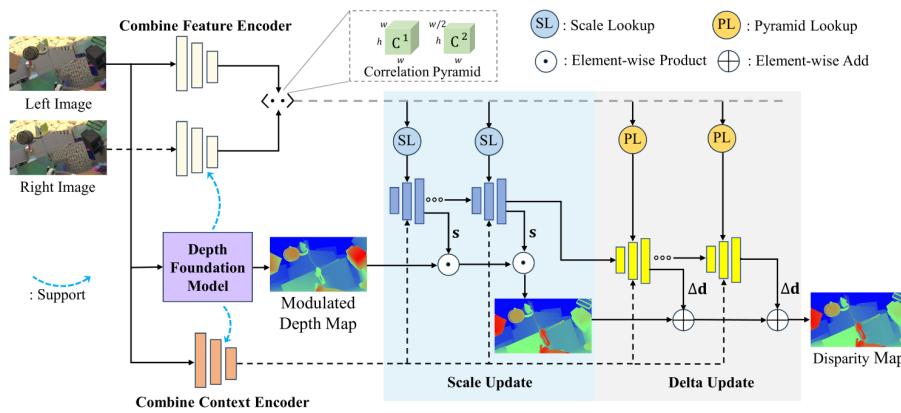


Figura 3.8: Esquema del modelo de DEFOM Stereo [6]

## Cómo funciona DEFOM-Stereo

### Primera fase

El primer paso de DEFOM-Stereo comienza con la integración de un Depth Foundation Model. Igual que FoundationStereo, se utiliza Depth Anything, un modelo entrenado para predecir mapas de profundidad partir de imágenes individuales. Y de igual manera, se usa por su capacidad de captar información global de la escena, reconociendo relaciones espaciales aunque falten texturas o haya occlusiones. E igualmente, en DEFOM-Stereo se aprovecha esa capacidad global de representación, no para quedarse únicamente con el mapa de profundidad, sino para enriquecer la forma en la que se extraen las características que después se usarán en el emparejamiento estereóeo.

Para conseguirlo, se construye lo que llaman un Combined Feature Encoder. En esta parte, combinan el uso de la red convolucional de RAFT-Stereo,

que aporta detalles locales de las imágenes, y las une con las características globales que aporta Depth Anything. Además, en el caso de DEFOM Stereo, añaden un nuevo modelo DPT entrenable a Depth Anything, que se encarga de transformar los resultados de las representaciones globales de Depth Anything. Este paso es necesario porque los resultados del primer paso de Depth Anything son representaciones abstractas que deben ser transformadas para poder ser usadas. Y en vez de obtener únicamente una representación de la profundidad como hace Depth Anything, se añade esta nueva DPT que da como resultado las características globales de las imágenes. Aquí el resultado va a ser un volumen de correlación entre las dos imágenes del par estéreo basados en las características globales y locales de ambas imágenes.

Junto a esto, DEFOM-Stereo incorpora también un Combined Context Encoder, que en este caso solo se aplica a la imagen izquierda. Este paso se encarga de aportar el contexto necesario para guiar el proceso recurrente que irá refinando la disparidad. Al igual que en el Combined Feature Encoder, aquí también se combinan las dos fuentes. Por un lado la red convolucional de RAFT-Stereo, que aporta información del contexto local de las imágenes. Y por otro lado, los dos pasos que hemos visto anteriormente usando Depth Anything y el modelo DPT entrenable para obtener los detalles de contexto globales. De esta forma va a generar unos mapas de contexto que servirán para inicializar el estado de los módulos ConvGru y posteriormente para ayudar en el refinamiento del mapa de disparidad en las sucesivas iteraciones.

## Segunda fase

En la segunda fase, llamada Scale Update, el objetivo principal es corregir la disparidad inicializada con el mapa de profundidad de Depth Anything. Este mapa es una buena aproximación, pero tiene el problema de que no está en la escala correcta y además puede presentar inconsistencias dentro de la propia imagen.

Para esta fase se van a utilizar las tres piezas de información de la fase anterior. El mapa de disparidad de Depth Anything, el volumen de correlación que proviene del Combined Feature Encoder y que contiene la información de como se pueden relacionar entre ellas, y los mapas de contexto que se han generado en el Combined Context Encoder.

Con todos estos elementos, la fase de Scale Update utiliza un módulo ConvGRU diseñado para predecir un mapa de disparidad. En cada iteración, este ConvGRU actualiza el valor de la disparidad. El objetivo de este ConvGru es actualizar en cada iteración la disparidad teniendo en cuenta los

detalles de contexto globales y locales, mientras que el ajuste de los detalles se deja para una fase posterior.

Para ello, usa el volumen de correlación obtenido de la fase anterior con el Combined Feature Encoder. Con ello construye un conjunto de valores de correlación que le indican al ConvGRU si una cierta escala hace que los píxeles coincidan bien entre izquierda y derecha. Esa comparación permite que el modelo aprenda a ajustar la escala correcta de manera iterativa, incluso en escenas con disparidades grandes o regiones difíciles.

Los mapas de contexto que se obtuvieron con el Combined Context Encoder, en esta fase se usan para inicializar el estado del ConvGRU y dar una primera estructura de la escena. Y posteriormente en cada iteración, se usan para obtener información del contexto de la imagen y poder hacer predicciones más precisas que si sólo se basara en las correlaciones de las dos imágenes del par estéreo.

### Tercera fase

La última fase de DEFOM Stereo se denomina Delta Update. Su objetivo es refinrar los detalles locales de la disparidad corrigiendo bordes, contornos y pequeñas inconsistencias. Es decir, mientras la fase de Scale Update se centraba en la imagen global, la fase de Delta Update trata de definir los detalles de la misma, como los bordes de los objetos, las transiciones entre superficies o pequeñas variaciones en zonas que todavía no han quedado bien definidas.

La entrada principal de esta fase es la disparidad corregida en escala que generó la fase de Scale Update. A esto se suma el estado interno del módulo ConvGRU que ya se ha ido refinando en las iteraciones anteriores gracias al volumen de correlación y a los mapas de contexto de la primera fase.

Lo que hace el ConvGRU en esta fase es refinrar de forma muy fina la disparidad, corrigiendo los errores pequeños que aún queden de las fases anteriores. Para ello, va a comparar la región alrededor del píxel actual para mejorar el ajuste de la disparidad, moviendo un píxel ligeramente hacia un lado o hacia otro para alinear mejor los pares estéreo en regiones con bordes o texturas finas.

Los mapas de contexto generados en la primera fase siguen teniéndose en cuenta en esta fase. En cada iteración del ConvGRU, se vuelven a utilizar como entradas adicionales, proporcionando una guía estructural que ayuda a que los ajustes locales respeten la coherencia general de la escena.

La salida de esta fase es el mapa de disparidad correspondiente a las imágenes del par estéreo con el que poder calcular la profundidad de cada píxel.

## **Resultados**

Los resultados que obtenemos con DEFOM Stereo son muy similares a los que obtuvimos con Foundation Stereo y presentan los mismos problemas, algo que será recurrente entre los modelos. Es capaz de detectar correctamente la profundidad general de la imagen y de detectar la grieta que se produce, pero con un cambio en la profundidad con respecto a su entorno que no va a facilitar el estudio automatizado del mismo.

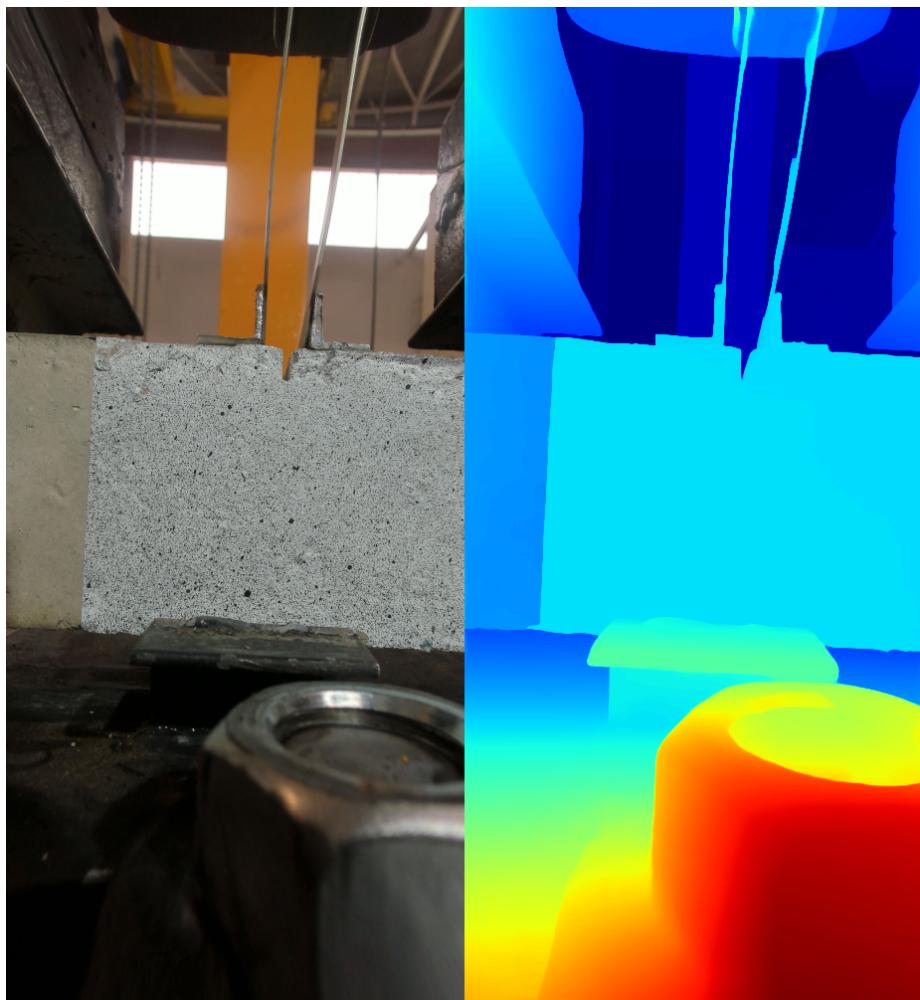


Figura 3.9: Visualización de resultados de DEFOM Stereo al inicio del experimento

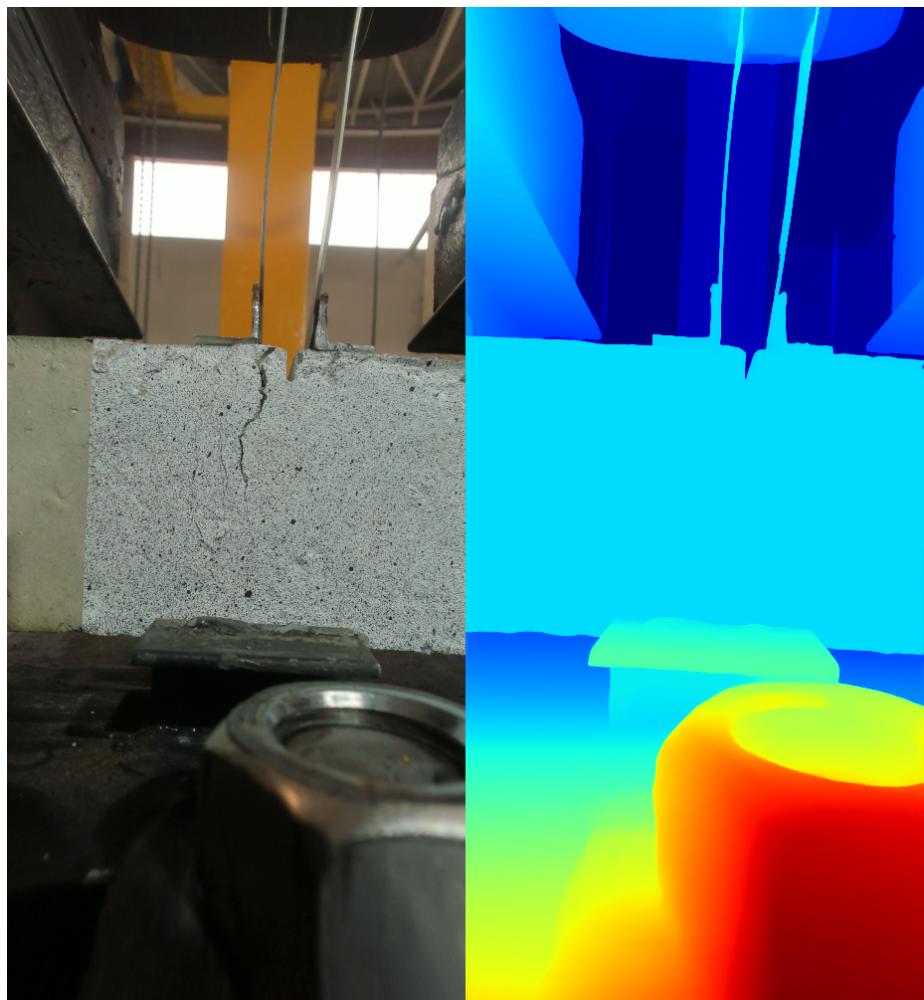


Figura 3.10: Visualización de resultados de DEFOM Stereo en un punto intermedio del experimento

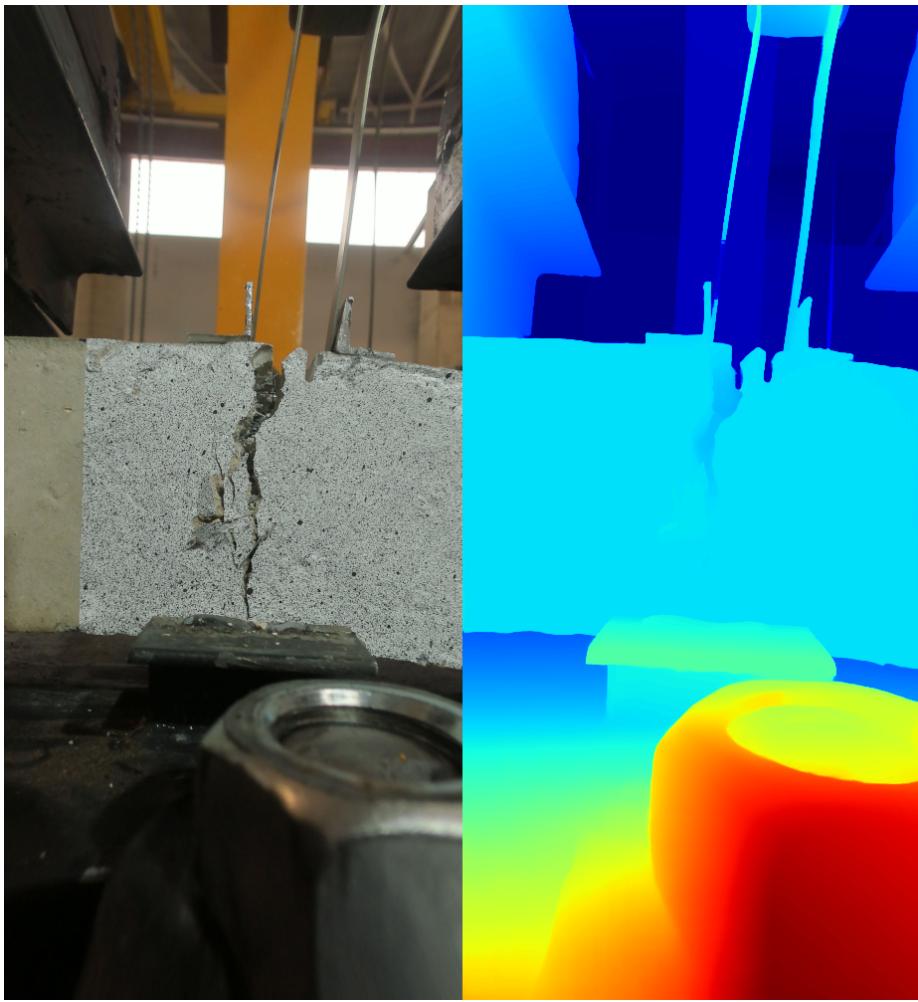


Figura 3.11: Visualización de resultados de DEFOM Stereo al final del experimento

### 3.4. VGGT

VGGT es un modelo desarrollado por el Visual Geometry Group de la Universidad de Oxford junto con Meta AI. Se trata de una red neuronal basada en transformadores capaz de, a partir de una o varias de imágenes de una escena, estimar directamente todos sus atributos 3D principales, como los parámetros de cámara, mapas de profundidad, mapas de puntos y trayectorias 3D de los píxeles.

Lo que diferencia a VGGT de otros enfoques es que no depende de métodos clásicos de geometría visual ni de procesos iterativos de optimización. En su lugar, con una sola pasada de red (feed-forward), es capaz de reconstruir la escena obteniendo unos resultados que no necesitan post-procesamiento. Igual que los modelos anteriores, es un modelo generalista que no está especializado en resolver un sólo caso concreto.

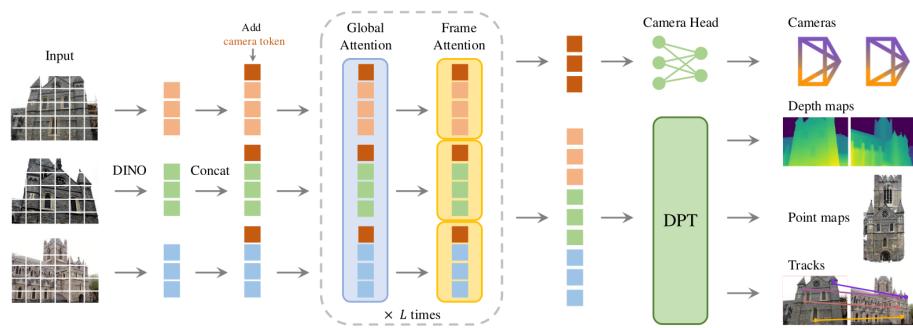


Figura 3.12: Esquema del modelo de VGGT Stereo [19]

## Cómo funciona VGGT

### Primera fase

VGGT comienza su proceso dividiendo cada imagen en pequeños fragmentos, que formarán los tokens de características mediante el modelo DINOv2. Estos tokens funcionan como representaciones numéricas de cada parte de la imagen, capturando tanto texturas como patrones geométricos. Para cada imagen, también se añaden unos tokens de cámara, que no provienen directamente de los píxeles, sino que son vectores aprendidos durante el entrenamiento. Su función es aportar la información de la cámara asociada a esa vista. Estos tokens por tanto corresponderían a una representación de la información de los parámetros intrínsecos y extrínsecos que describen cómo está posicionada y las características de la cámara.

La combinación de los tokens de las imágenes y los tokens de la cámara permite que en las siguientes fases un transformer pueda relacionar lo que se ve en cada imagen con la geometría de la cámara que la capturó. Esto hace posible que VGGT no solo identifique patrones de profundidad o correspondencias entre imágenes, sino que además lo haga usando un sistema de referencia 3D que se mantiene durante todo el proceso. Con esto consigue que la predicción de la profundidad de la escena siempre tenga

en cuenta la cámara desde la cual se tomaron las imágenes, manteniendo coherencia en todo el conjunto de pares.

### **Segunda fase**

La segunda fase de VGGT comienza cuando ya se tienen los tokens de imagen y los tokens de cámara de todas las imágenes. Con esto se combina la información de todas ellas para que cada token no sólo represente la información de una sola imagen, si no que también añade el contexto del resto de las imágenes.

Para ello utiliza dos fases. La primera se denomina Global Attention y es en este fase donde se van a utilizar todos los tokens de todas las imágenes. De esta forma se extrae información del contexto del resto de imágenes para capturar la información relativa a profundidad de una manera que sea coherente entre todas ellas. Además, se usan lo token de cámara para relacionar las distintas imágenes con sus respectivos puntos de vista.

La segunda fase se denomina Frame Attention. En esta fase sólo se utiliza los tokens de cada imagen individual. Esta fase se encarga de aportar el contexto de los detalles concretos de cada imagen, y además permite que estos detalles de cada una de ellas no se pierdan cuando se combinan todas ellas en la fase anterior donde sólo se tienen en cuenta el contexto global.

Este mecanismo se repite en bucle durante un número de iteraciones. De esta forma cada token se va refinando de tal forma que contenga tanto información de la región de la imagen de la que proviene, como información del contexto general del conjunto de imágenes que se están procesando. Este nuevo conjunto de tokens son los que se utilizarán en la próxima fase del proceso para generar las predicciones de profundidad y generar los mapas de puntos 3D.

### **Tercera fase**

La tercera fase de VGGT recibe los tokens refinados en la fase anterior y se va a encargar de producir los resultados finales de predicción de la profundidad y los puntos 3D.

El conjunto de tokens que se reciben se dividen en dos procesos. Por un lado, se seleccionan únicamente los tokens visuales y se procesarán en un DPT (Dense Prediction Transformer). Este transformer va a utilizar estos token visuales para generar los mapas de predicción de profundidad de cada píxel de las imágenes y los mapas de los puntos 3D.

Por otro lado, se van a seleccionar los tokens de la cámara y se van a procesar en un módulo denominado Camera Head. Este módulo se encarga de predecir los parámetros intrínsecos y extrínsecos de cada cámara. Es decir, este módulo se encarga de transformar los tokens de cámara generados desde la primera fase, y que eran una representación de las características de la cámara que tomó las imágenes, y con ellos hace una predicción de los valores intrínsecos y extrínsecos de las cámaras.

## Resultados

Como pasaba en los anteriores modelos, VGTT es capaz de crear una buena representación de la escena en general y predecir correctamente la profundidad de sus elementos. Pero tiene el mismo problema en la zona de la grieta, que aunque la detecta y predice una profundidad diferente a los píxeles de su entorno, la diferencia es muy pequeña. Esto representa un problema a la hora de detectar esa zona concreta en un proceso automatizado por la dificultad de diferenciarla de otras zonas de la imagen donde el proceso de predicción también da pequeñas diferencias.

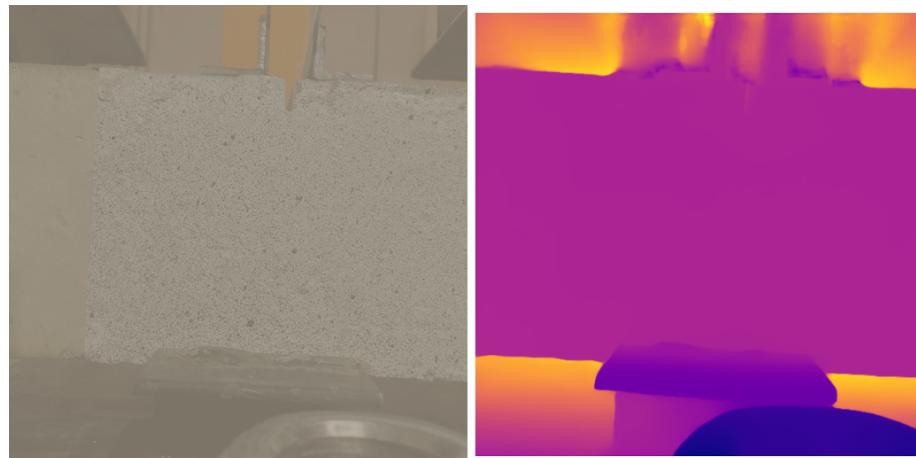


Figura 3.13: Visualización de resultados de VGTT al inicio del experimento

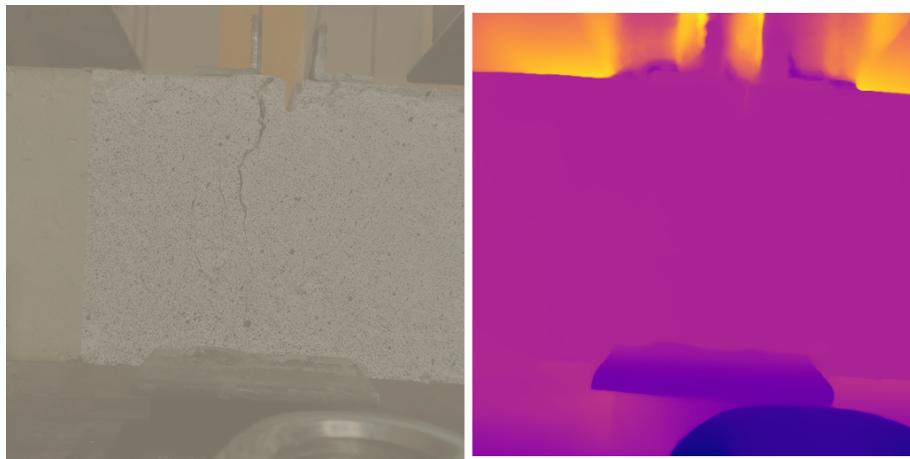


Figura 3.14: Visualización de resultados de VGGT en un punto intermedio del experimento

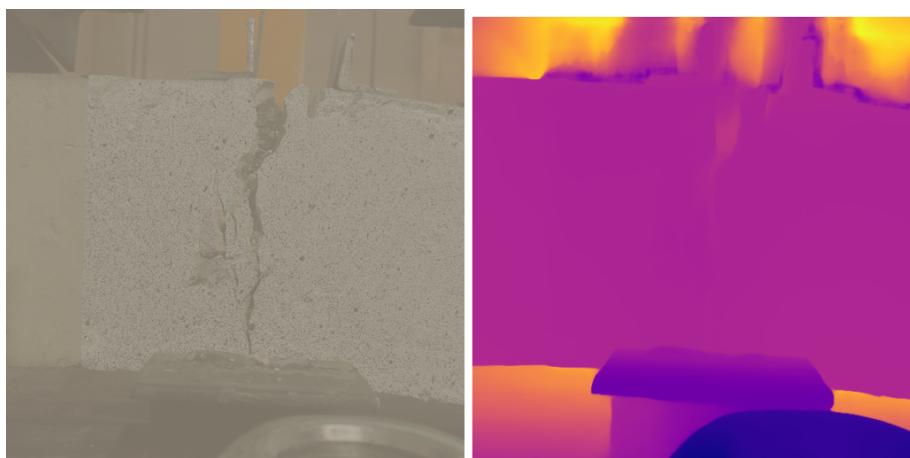


Figura 3.15: Visualización de resultados de VGGT al final del experimento

### 3.5. Detectores Esquinas

Una vez hemos podido obtener con alguno de estos modelos una predicción de la profundidad de cada píxel, necesitamos un proceso automático para poder hacer el seguimiento de los puntos. La primera aproximación ha sido detectar las esquinas del bloque de hormigón, y hacer un seguimiento de esos puntos usando los mapas de puntos 3D de los modelos.

Para ello se ha probado con los siguientes métodos.

## Harris

El detector de esquinas de Harris [4] se basa en analizar cómo varía la intensidad en una pequeña zona de la imagen cuando se desplaza en distintas direcciones. La idea es que en una zona plana los cambios son mínimos, en un borde sólo se producen de forma apreciable en una dirección, mientras que en una esquina las variaciones se pueden detectar en todas ellas. Para describir este comportamiento, Harris introduce el llamado tensor de estructura, que incluye toda la información de los gradientes de la imagen y a través de sus valores propios, determina si una región corresponde a una zona plana, un borde o una esquina.

En lugar de calcular directamente dichos valores propios, lo que es muy costoso computacionalmente, Harris y Stephens propusieron medirlo basándose en los invariantes del tensor de estructura, que son en concreto su traza y su determinante. De esta forma, definen una función de respuesta

$$R = \det(M) - k \operatorname{Tr}(M)^2,$$

donde  $k$  es un parámetro que ajusta la sensibilidad del detector. Este criterio permite distinguir entre esquinas, bordes y zonas planas de manera eficiente sin que le afecten las rotaciones o el contraste de la imagen.

El uso del detector de Harris consiste en que primero se calculan los gradientes de la imagen y se combinan con un filtrado gaussiano para reducir el ruido. Después se evalúa la respuesta  $R$  en cada píxel. Por último, se seleccionan como esquinas aquellos puntos donde esta respuesta es especialmente alta, asegurando además que sean máximos locales en su entorno. De esta manera, el algoritmo identifica de forma automática los puntos más representativos de la imagen, que suelen coincidir con esquinas.

## Resultados

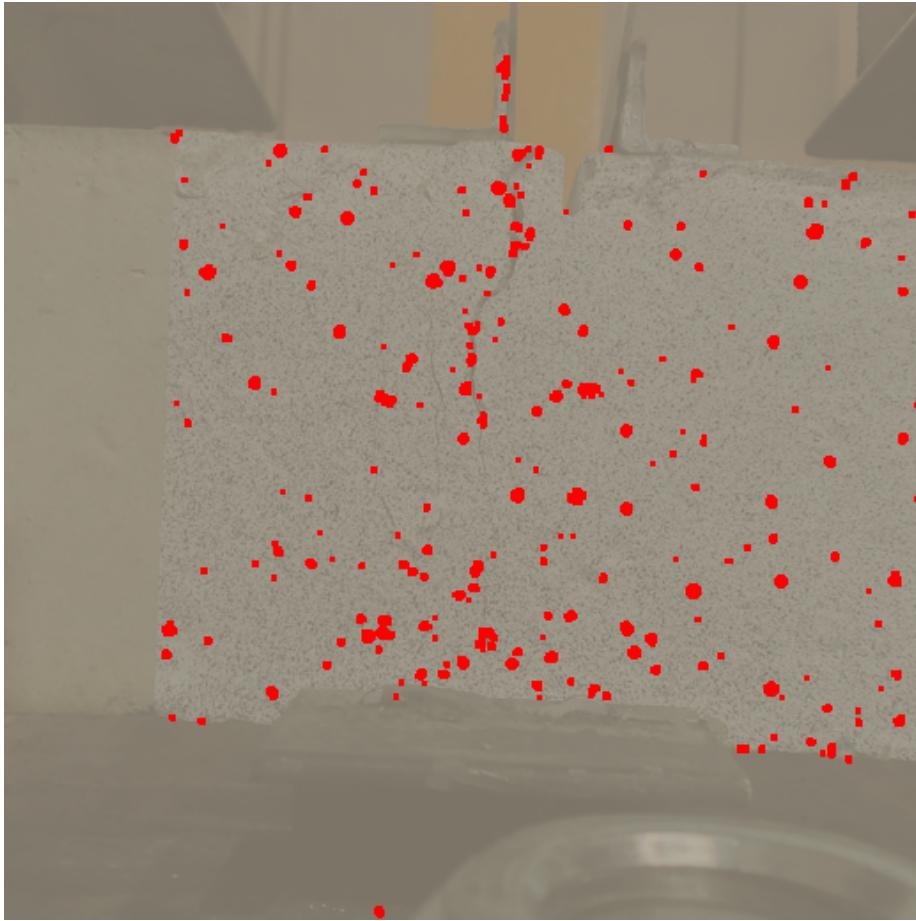


Figura 3.16: Visualización de resultados del detector de Harris

## Shi–Tomasi

El detector de esquinas de Shi–Tomasi [16] surge como una mejora directa del método de Harris. Al igual que en este, la idea principal es utilizar el tensor de estructura, que resume la información de los gradientes de la imagen dentro de una pequeña ventana. Los valores propios de este tensor indican cómo varía la intensidad de la imagen en distintas direcciones. Si ambos son pequeños la región es plana. Si uno es grande y el otro pequeño corresponde a un borde. Y si ambos son grandes se trata de una esquina.

La diferencia respecto al detector de Harris está en la forma de decidir cuándo un punto es una buena característica. Mientras que Harris propone una función de respuesta  $R$  basada en determinante y traza, Shi y Tomasi plantean evaluar únicamente el menor de los dos valores propios.

$$R = \min(\lambda_1, \lambda_2).$$

De este modo, un punto se considera una esquina válida siempre que este valor mínimo supere un cierto umbral. La ventaja de este criterio es que se descartan de manera natural los bordes, ya que en ellos uno de los valores propios será pequeño aunque el otro sea grande.

En el caso del detector de Shi-Tomasi, primero se calculan los gradientes de la imagen y se construye el tensor de estructura suavizado con un filtro gaussiano. A continuación se obtienen sus valores propios en cada píxel y se compara el menor de ellos con el umbral definido. Y por último, se seleccionan como esquinas aquellos puntos que superen el umbral y que sean máximos locales en su entorno. De esta forma el detector de Shi-Tomasi logra identificar las esquinas de la imagen.

## Resultados

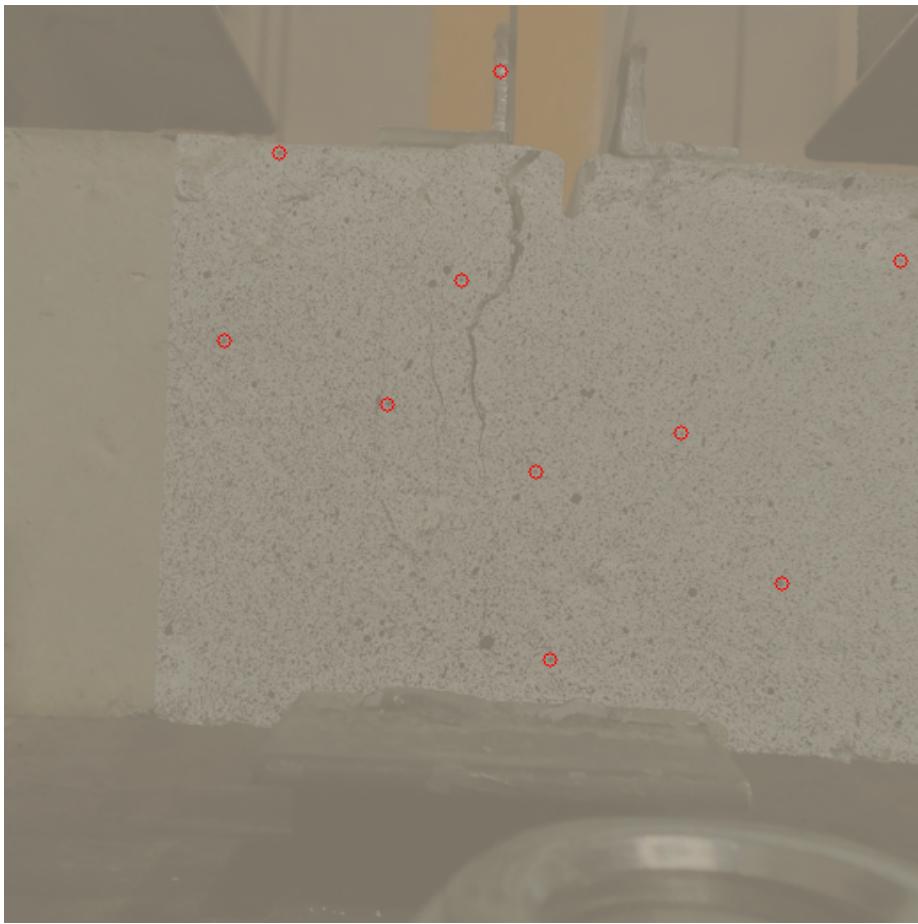


Figura 3.17: Visualización de resultados del detector de Shi-Tomasi

## FAST

El detector de esquinas FAST (Features from Accelerated Segment Test) [14] fue diseñado para ser muy rápido y poder usarse en aplicaciones en tiempo real. Mientras que otros detectores como Harris o Shi-Tomasi ofrecen buenos resultados, tienen un coste computacional alto que no permitía su uso en este tipo de aplicaciones.

El detector FAST consiste en analizar un píxel candidato  $p$  y comparar su intensidad con la de un conjunto de 16 píxeles dispuestos en círculo alrededor de él siguiendo el patrón de Bresenham. El criterio de esquina se basa en comprobar si existe un conjunto de  $n$  píxeles consecutivos en ese círculo que sean todos más brillantes que  $p$  o todos más oscuros, basándose

para ello en un umbral  $t$ . Si se cumple esta condición, el píxel central  $p$  se clasifica como esquina.

Para acelerar aún más el proceso, como primer paso se usa un *test rápido*, que consiste en que en lugar de comprobar los 16 píxeles desde el inicio, se verifican primero únicamente los de las posiciones arriba, abajo, izquierda y derecha. Si entre esos píxeles no se cumplen las condiciones necesarias, el píxel se descarta reduciendo las comparaciones y el coste computacional. Si se cumplen las condiciones en esos píxeles se comprueban el resto del conjuntos de píxeles alrededor del píxel inicial.

Para mejorar aún más el algoritmo, se mejoró empleando técnicas de aprendizaje automático que permiten optimizar el orden y la selección de las comparaciones a través de un árbol de decisión que clasifica cada píxel de manera más eficiente que el algoritmo escrito manualmente.

La mayor desventaja de FAST respecto a los detectores como el de Harris o Shi-Tomasi, es que no tiene en cuenta la escala o rotación, por lo que puede obtener peores resultados cuando la imagen contiene ruido o estructuras geométricas grandes.

## Resultados

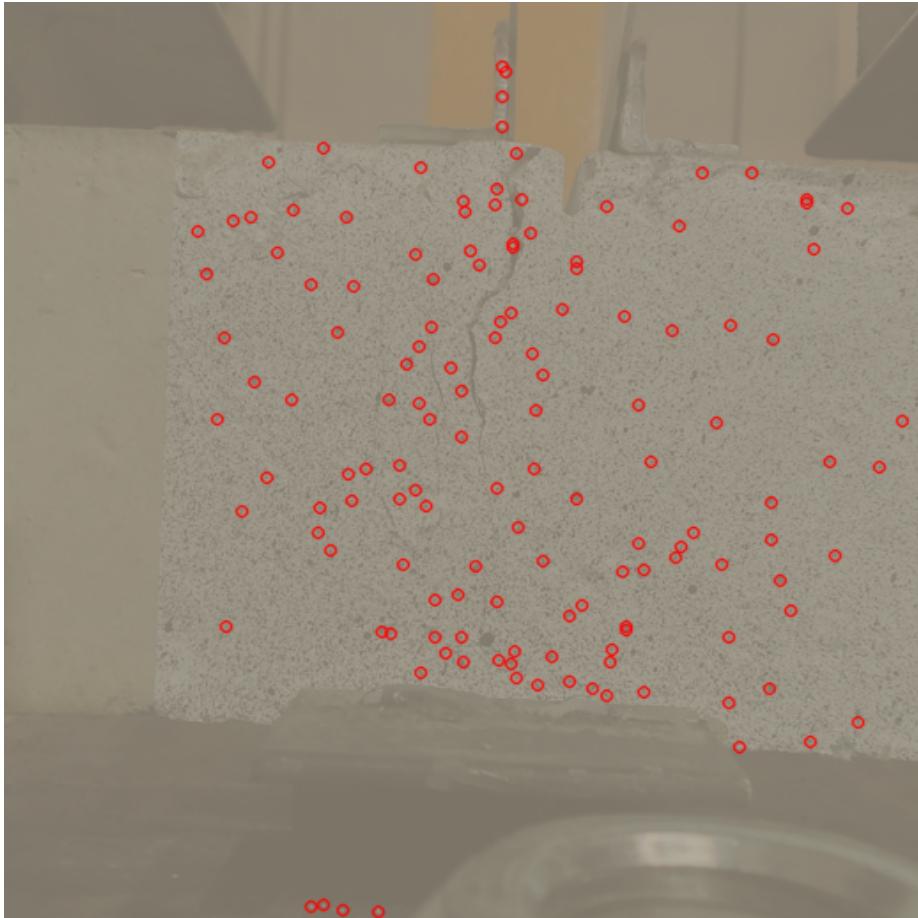


Figura 3.18: Visualización de resultados del detector de FAST

## ORB

El detector ORB (Oriented FAST and Rotated BRIEF) [15] es una alternativa más eficiente a los métodos como SIFT y SURF. Su idea es aprovechar la rapidez de FAST para detectar esquinas, pero mejorar su desempeño en imágenes rotadas y usar descriptores de las zonas para poder relacionarlas entre imágenes.

El proceso comienza con el detector FAST, que localiza los puntos candidatos. FAST no proporciona información sobre la orientación de la esquina, lo que limita su uso cuando la imagen está rotada. Para solucionarlo, ORB usa un cálculo del *centroide de intensidad* en la zona de los puntos vecinos del píxel. Este centroide permite definir un ángulo, de manera que

por cada punto detectado se tiene en cuenta no sólo su posición, sino también su orientación.

Una vez localizados los puntos clave, ORB usa un descriptor basado en BRIEF. Este descriptor compara intensidades de pares de píxeles dentro de una zona. El problema de BRIEF es que no obtiene buenos resultados con las rotaciones, pero ORB lo soluciona teniendo en cuenta la orientación estimada en el paso anterior. Además, aplica un procedimiento de selección para quedarse con las comparaciones más relevantes.

Con estos dos procesos, ORB combina la rapidez de FAST con la eficiencia de un BRIEF mejorado, alcanzando un rendimiento muy alto en tiempo real y siendo mucho más ligero que SIFT o SURF.

## **Resultados**

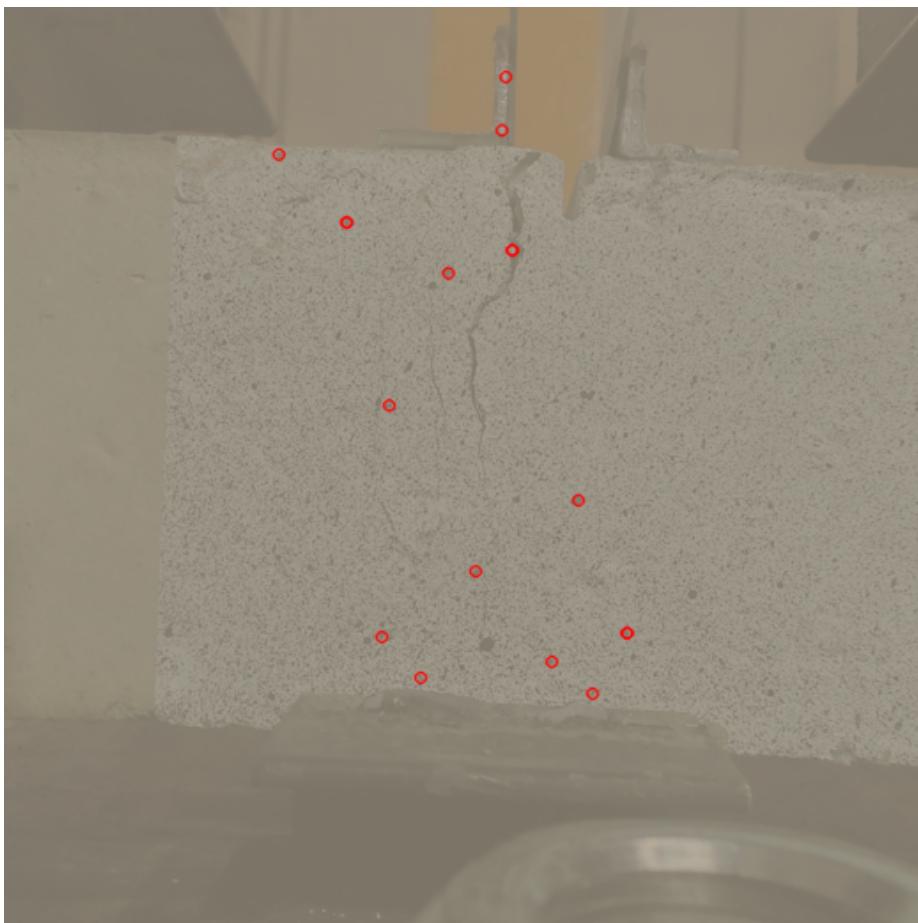


Figura 3.19: Visualización de resultados del detector de ORB

## SuperPoint

El detector de esquinas **SuperPoint** [1] surge para solventar las limitaciones de los métodos clásicos que basan sus predicciones en la intensidad de los píxeles. SuperPoint usa aprendizaje profundo para aprender qué puntos de una imagen son potencialmente interesantes y cómo describirlos.

Los primeros modelos de este tipo se entrenaron para aprender a detectar las formas básicas como triángulos, rectángulos, etc. Pero tenían el problema de que cuando se aplicaban a escenas del mundo real con formas mucho más complejas, los resultados no eran suficientemente buenos.

Para solventar este problema, se usó un procedimiento denominado **Homographic Adaptation**. Este procedimiento consiste en tomar imágenes reales sin etiquetar, deformarlas aplicando diferentes homografías y usar al

propio detector para ver qué puntos se mantienen a pesar de esas transformaciones. Esos puntos repetidos se convierten en etiquetas automáticas, que permiten seguir entrenando al modelo sin necesidad de intervención humana.

El modelo resultante de este entrenamiento es SuperPoint. Su arquitectura es una red convolucional profunda que se divide en dos ramas. Una de ellas produce un mapa de probabilidad de puntos de interés, gracias a una técnica llamada *píxel shuffle* que le da precisión sub-píxel. La otra rama genera descriptores numéricos asociados a cada punto, de forma que no sólo se localizan esquinas, sino que también se obtienen representaciones para poder compararlas entre imágenes.

De esta forma, se obtienen tanto las posiciones como los descriptores de los puntos clave. Y con ello se consigue que SuperPoint obtenga buenos resultados en escenarios con cambios de iluminación o con diferentes puntos de vista.

## Resultados

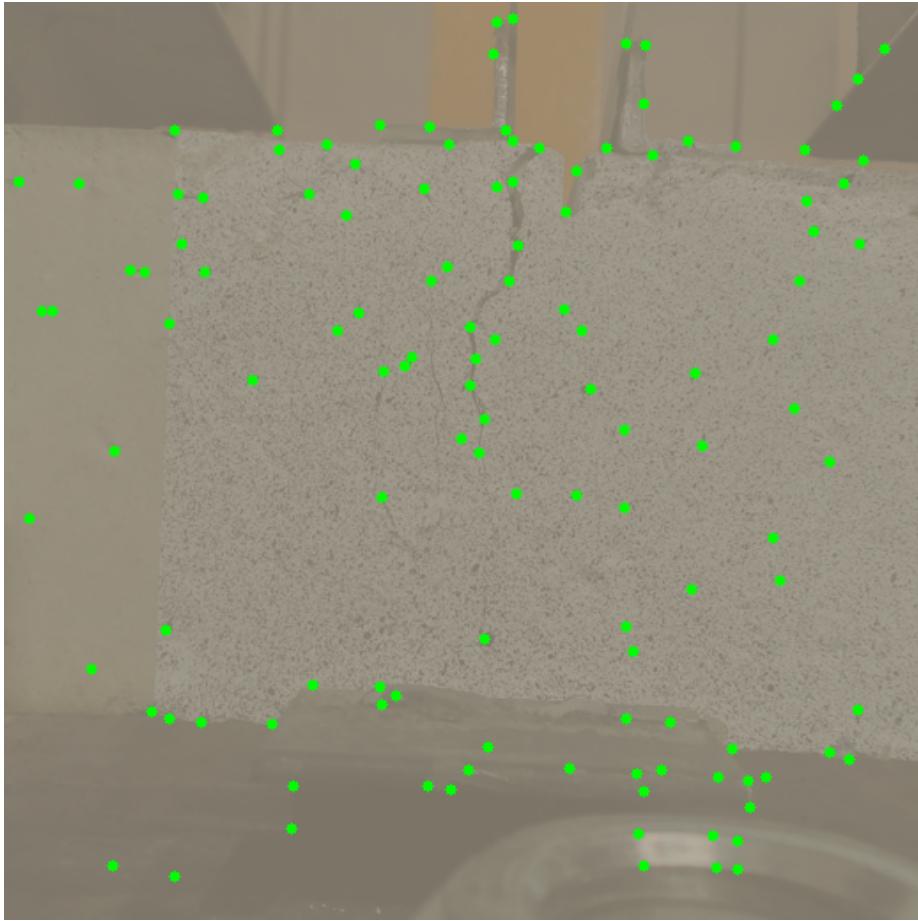


Figura 3.20: Visualización de resultados del detector de SuperPoint

## D2-Net

El detector **D2-Net** [3] cambia la forma del proceso de detección y descripción de puntos. Mientras que los métodos anteriores siguen una estrategia de detectar primero los puntos y de describir el contexto después, D2-Net realiza el proceso de forma inversa. Primero genera una representación de la imagen y a partir de ella extraer tanto los descriptores como las posiciones de los puntos clave.

Para ello, utiliza una red convolucional basada en VGG16 de la que se obtienen mapas de características que actúan de dos formas distintas. Por un lado, cada posición de estos mapas se interpreta como un descriptor local que representa la información de la imagen. Por otro lado, estos mapas permiten localizar los puntos de interés seleccionando como esquinas aquellos

píxeles que destacan como máximos locales. De esta manera, la detección ya no depende de estructuras locales simples como esquinas o bordes, sino de información de contexto de la imagen que ha aprendido la red. Esto permite obtener mejores resultados con imágenes de distinta iluminación o con diversas condiciones distintas del momento en que se toman las imágenes.

Esto permite que D2-Net obtenga menos puntos clave que los otros métodos, pero más precisos en condiciones difíciles, como el paso de día a noche o la localización en interiores con estructuras repetitivas.

## Resultados

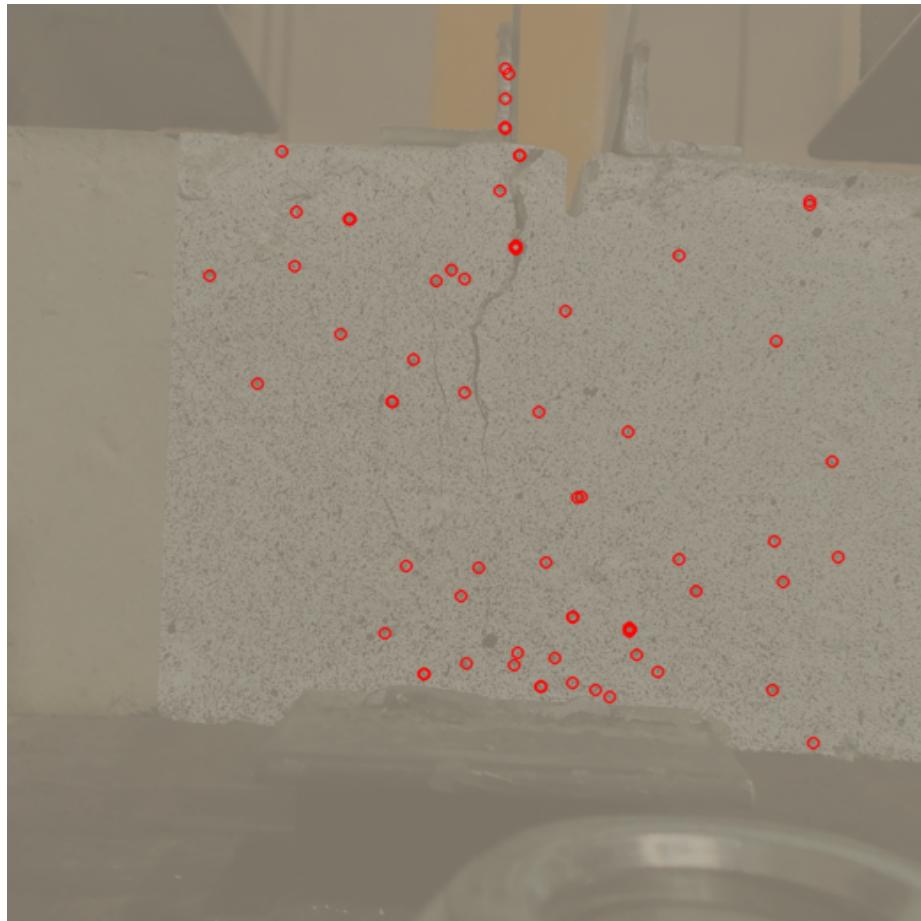


Figura 3.21: Visualización de resultados del detector de D2-Net

## R2D2

**R2D2** (Repeatable and Reliable Detector and Descriptor) [13] es un detector y descriptor que surge para solventar un problema de los métodos anteriores, y es que se centran únicamente en que los puntos detectados sean repetibles, es decir, que aparezcan de forma consistente bajo distintos cambios de vista o de iluminación. El problema es que ser repetible no siempre resulta suficientemente distintivo. Por ejemplo, en un tablero de ajedrez, todas las esquinas de las casillas son repetibles, pero no se pueden distinguir unas de otras.

R2D2 busca puntos de interés que sean repetibles y fiables. Repetibles en el sentido de que se puedan detectar bajo diferentes transformaciones. Y fiables en el sentido de que sus descriptores sean suficientemente distintivos como para emparejarlos. Para lograrlo, el método utiliza una red convolucional que analiza la imagen completa y produce tres resultados. En primer lugar, un mapa de repetibilidad que señala qué regiones son buenas candidatas a contener puntos consistentes. En segundo lugar, un mapa de fiabilidad que estima si los descriptores que allí se generen serán realmente distintivos. Y finalmente, un conjunto de descriptores locales que representan la apariencia de cada píxel de la imagen.

R2D2 combina esas tres salidas. Se seleccionan únicamente aquellos píxeles que son máximos locales en el mapa de repetibilidad y que al mismo tiempo tienen una alta fiabilidad. De este modo, se eliminan de manera automática las zonas que aunque se repitan con frecuencia, no sirven para distinguirse, como pueden ser patrones regulares de ventanas o texturas de hojas.

Igual que pasaba con D2-Net, R2D2 consigue un conjunto de puntos clave más reducido pero que ofrecen una mayor precisión. Los puntos seleccionados no solo son fáciles de detectar de nuevo en otras imágenes, sino que también se pueden emparejar con los de otras imágenes.

## Resultados

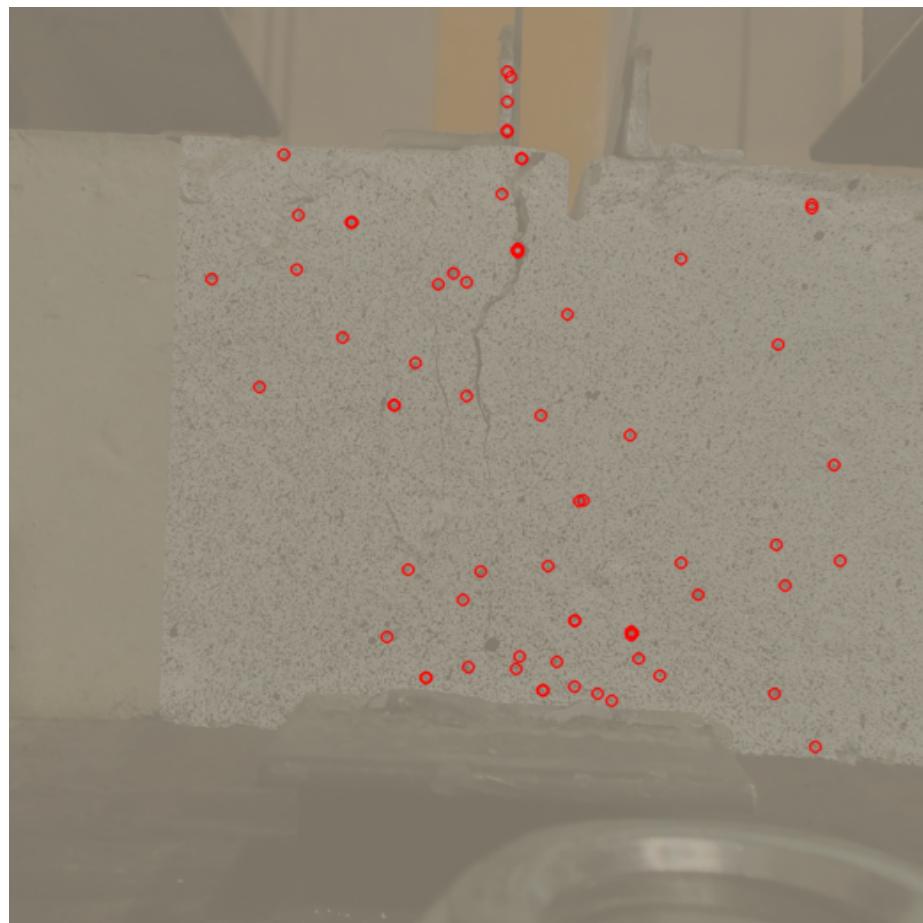


Figura 3.22: Visualización de resultados del detector de R2D2

## Conclusión

Como vemos con los resultados, probablemente debido al patrón speckle del bloque de hormigón, no conseguimos unos resultados que permitan detectar los puntos del bloque que nos lleven a poder analizar el progreso de la grieta a lo largo del tiempo y poder realizar su estudio.

Los detectores encuentran puntos de interés en cualquier parte debido a este patrón y las sombras que proyecta. Esto hace que no se haya podido usar estos métodos como punto de partida para seleccionar unos puntos de partida y a partir de ellos con los mapas 3D de los modelos de visión estéreo realizar una estimación automática de tamaño de las grietas que se generan en el bloque.

## 3.6. Detectores de líneas

Como los detectores de esquinas no produjeron los resultados esperados, se decidió usar un detector de líneas, para ver si siendo capaces de detectar la forma del bloque de hormigón a través de las líneas de sus bordes, podíamos ser capaces de encontrar las esquinas a través de la intersección de las mismas y de esta forma encontrar las esquinas. Y partir de ellas ser capaces de encontrar los puntos de interés del bloque y realizar el seguimiento de los puntos y realizar el estudio del progreso del estado del bloque y las grietas.

### Probabilistic Hough

El método que se uso fue el de Probabilistic Hough. La transformada de Hough tradicional detecta líneas en una imagen a partir de los píxeles que forman sus bordes. El procedimiento comienza obteniendo un mapa de bordes, normalmente con un detector como Canny. A continuación, cada píxel de ese mapa se transforma al espacio de parámetros de las rectas, es decir, a una representación donde cada posible línea queda definida por una orientación y una posición. En este espacio, un mismo píxel no corresponde a una sola recta, sino a un conjunto de posibles rectas que podrían pasar por él. Al realizar esta transformación para todos los píxeles de borde, los que pertenecen a la misma línea en la imagen tienen los mismos parámetros, lo que permite detectar la recta que forman. De esta manera, con los puntos con los mismos valores en el espacio de parámetros se pueden reconstruir las líneas de la imagen, incluso si aparecen fragmentadas. La mayor desventaja de este método es que es necesario procesar todos los píxeles de borde, lo que es computacionalmente costoso [2].

Una de las primeras variantes fue propuesta por Stephens [18]. En lugar de trabajar únicamente contando cuántos puntos de borde forman cada recta, plantea interpretar el proceso en términos de probabilidad. Lo que realiza es que cada línea candidata en la imagen recibe un valor que indica qué tan probable es que exista en función de cómo encajan en la línea los puntos de borde detectados. En vez de basarse solo en acumulaciones según las coincidencias de los puntos, el método busca las rectas que resultan más probables según los datos disponibles. Esto permite que funcione mejor cuando las imágenes contienen ruido u otros defectos, pero no mejora el coste computacional.

Una alternativa más práctica fue introducida por Kiryati, Eldar y Bruckstein, quienes plantearon el denominado Hough probabilístico que selecciona para realizar el espacio de parámetros únicamente una parte de los píxeles de

borde [8]. En lugar de transformar todos los puntos detectados, se selecciona aleatoriamente una fracción representativa y únicamente ellos se proyectan al espacio de parámetros. De este modo, se reduce el número de operaciones necesarias manteniendo una detección fiable siempre que la muestra aleatoria sea lo suficientemente grande para contener información sobre las líneas de la imagen.

Otra versión de este método es el trabajo de Matas, Galambos y Kittler, que con lo que denominan Hough probabilístico progresivo [10]. Lo que realizan con esta versión es no esperar a procesar todos los puntos de borde antes de decidir si una línea existe, sino ir comprobando de manera progresiva mientras se van analizando los datos. Cuando una recta candidata reúne suficientes puntos que confirman su presencia, se acepta como línea detectada y se eliminan esos puntos para no volver a utilizarlos. Esto permite detener el cálculo antes de recorrer toda la imagen y evita repetir el estudio de algunos puntos. Al contrario que en los otros métodos, en lugar de indicar la recta de la imagen como una recta infinita, este método devuelve los segmentos de línea encontrados. De esta forma se consigue reducir el coste computacional y obtener unos resultados de detección de líneas con buenos resultados.

## **Resultados**

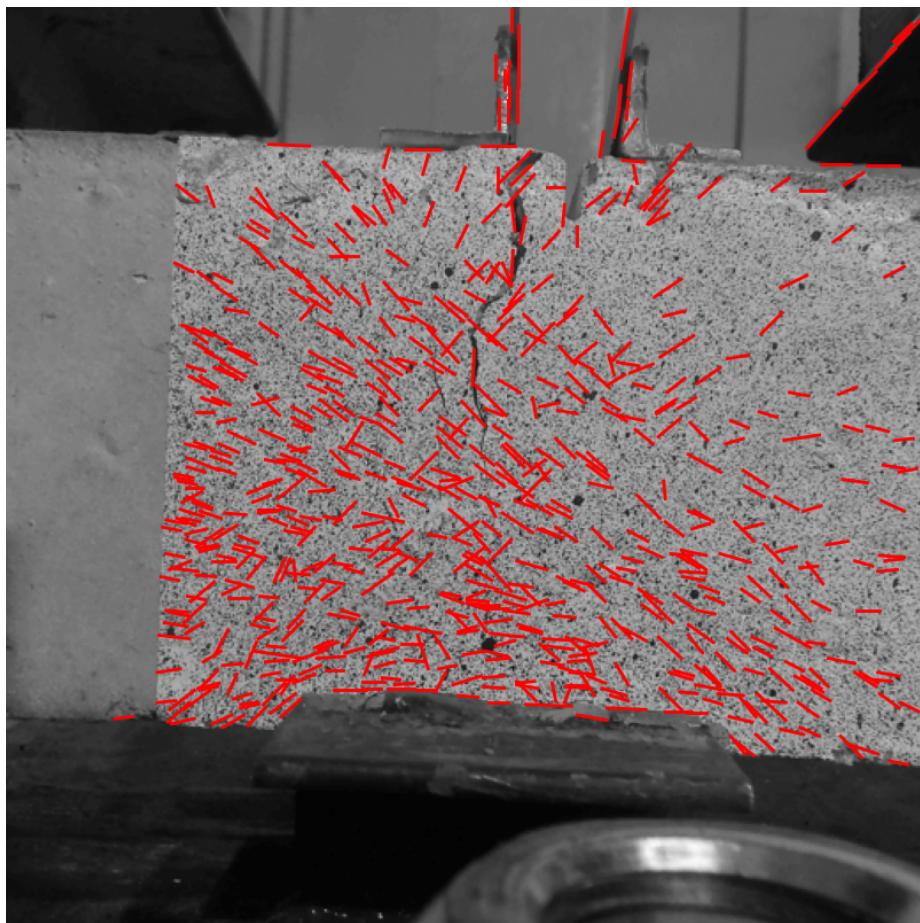


Figura 3.23: Visualización de resultados del detector Probabilistic Hough

## Conclusión

De la misma forma que ocurría con los detectores de líneas, los resultados de la detección de líneas se ven perturbados por el patrón speckle del bloque de hormigón, por lo que igual que pasaba anteriormente no nos permite usar este método para poder seleccionar puntos que nos permitan hacer un estudio de los puntos del bloque que nos permitan analizar la progresión de la grieta a lo largo del tiempo.

## 3.7. Detección de la profundidad

Como alternativa, hemos usado los mapas de profundidad que nos proporcionan los métodos de visión estéreo y analizar los cambios de profundidad

en la imagen directamente, y de esta forma ver si podemos a través de estos cambios, analizar dónde y cómo se produce la grieta en el bloque de hormigón.

## Sobel

Para detectar la zona donde se produce la grieta en el bloque de hormigón, hemos usado la matriz de profundidad generada por el modelo de vggt. Hemos usado únicamente la matriz correspondiente a cada imagen sin tener en cuenta las anteriores, porque el modelo detecta la grieta con unos valores de profundidad mínimos, y las variaciones que se producen al generar las predicciones en cada par son mayores que estos valores.

Esto se produce porque cada medición de cada par es independiente, y los valores obtenidos no son siempre los mismos. Por lo que al compararlos unos con otros el resultado es que la profundidad de la grieta suele ser el valor de diferencia mínimo haciéndola indetectable.

Al usar sólo la matriz de cada imagen, podemos detectar únicamente las diferencias de profundidad, y aunque sean muy pequeñas, podemos ser capaces de detectarlas al menos a partir de un umbral.

Para ello hemos utilizado un método de gradiente, Sobel [17], que mide la tasa de cambio de la profundidad tanto en dirección horizontal como en vertical.

El operador de **Sobel** [17] es un método basado en gradientes que se emplea para resaltar los bordes de una imagen. La idea principal es detectar las zonas donde se producen cambios bruscos de intensidad, ya que estos corresponden a los contornos de los objetos o, en nuestro caso, a discontinuidades en el mapa de profundidad que indican la presencia de una grieta.

El procedimiento se basa en la convolución de la imagen con dos filtros  $3 \times 3$ , uno orientado en dirección horizontal y otro en dirección vertical. Cada filtro estima la derivada parcial de la intensidad en una dirección. De esta manera, se obtienen dos mapas:

$$G_x = I * K_x, \quad G_y = I * K_y,$$

donde  $K_x$  y  $K_y$  son los núcleos de Sobel, diseñados para combinar la detección de bordes con un ligero suavizado que reduce la sensibilidad al ruido.

A partir de estas derivadas se calcula la magnitud del gradiente:

$$G = \sqrt{G_x^2 + G_y^2},$$

que indica en cada píxel la intensidad del cambio. Valores grandes de  $G$  corresponden a variaciones fuertes en la imagen, es decir, a bordes bien definidos. En el caso de los mapas de profundidad, esto permite localizar de forma precisa las zonas donde la superficie presenta una discontinuidad, lo que se corresponde con los bordes de la grieta.

Una vez aplicado Sobel, nos quedamos con los puntos que tienen una diferencia de profundidad con su entorno mayor a un umbral, y con estos puntos creamos una máscara que representa las zonas donde se ha podido generar la grieta. Es decir, si en el bloque que es mayormente plano hay una zona donde se ha detectado una hendidura, se marcará esa zona en la máscara como una zona donde se ha producido una grieta.

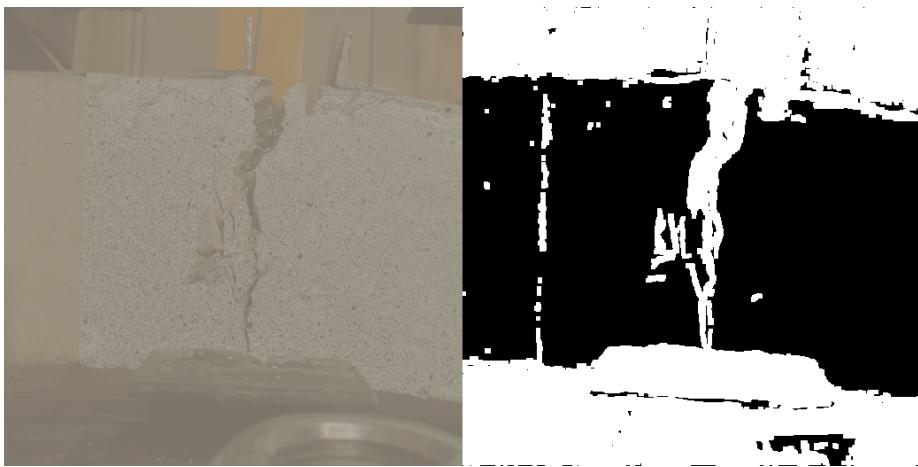


Figura 3.24: Máscara obtenida usando el gradiente de Sobel. Los puntos blancos son zonas donde se habría detectado zonas de grieta, y los negros el resto de puntos.

## Limpieza de la máscara de Sobel

Para reducir los puntos de ruido de la máscara y las zonas que no corresponden a la grieta, primero realizamos un proceso de limpieza de la máscara a través de un proceso de limpieza morfológica.

El proceso de limpieza morfológica consta de dos pasos. El primer paso denominado erosión, se encarga de eliminar los puntos blancos aislados de la máscara y de reducir los bordes de las regiones. El segundo paso denominado dilatación, revierte el proceso de reducción de bordes de las regiones para devolverlas a su tamaño, pero ya con el ruido que tenían alrededor limpiado.

Para realizar este proceso se genera un kernel, que es como una pequeña máscara o matriz que se aplica a cada píxel de la imagen. En el primer paso de erosión, si algún píxel dentro de la máscara es 0 o negro indicando que no es una zona de grieta, el píxel que se está procesando también será 0, eliminando los puntos de ruido. En el segundo paso de dilatación, si algún píxel dentro de la máscara es 1 o blanco, el píxel procesado se convertirá en 1 también, devolviendo el tamaño y remarcando las zonas que se habían erosionado en el paso anterior.

Con esto conseguimos eliminar los puntos sueltos y generar regiones más compactas.

## ROI

Para mejorar la detección de las zonas de la imagen donde se produce la grieta, usamos ROI (region of interest) de tal forma que vamos a seleccionar sólo la parte de la imagen donde se encuentra el bloque de hormigón. Y descartar lo que queda fuera de esta zona.

Esta zona la seleccionamos a partir de la máscara que generamos a partir de los valores obtenidos con Sobel. Es decir, con los valores de la diferencia de profundidad con respecto a su entorno.

Como el bloque de hormigón está en una zona donde la profundidad no varia al ser una zona plana, usando como referencia el primer par de imágenes donde el bloque aún está intacto, se va a detectar prácticamente el bloque de hormigón completo sin zonas de grietas.

A partir de esta máscara, seleccionamos una zona rectangular donde se encuentra el bloque de hormigón. Para ello buscamos el punto de la primera columna y el punto de la última columna de la imagen donde hay de forma consecutiva un número de píxeles en los que no se ha detectado una zona de grieta, es decir valores de 0. Esto es debido a que como el bloque es continuo, y está rodeado de zonas con un valor de diferencia de profundidad alto, podemos detectar la zona aproximada de donde se encuentra el bloque de hormigón, buscando los puntos donde empieza la zona continua del bloque y eligiendo esos dos puntos puntos, uno en cada lado, como esquinas desde donde partan los lados del rectángulo.



Figura 3.25: Máscara obtenida usando el gradiente de Sobel para la primera imagen sin grieta, que se usará como referencia para ROI.

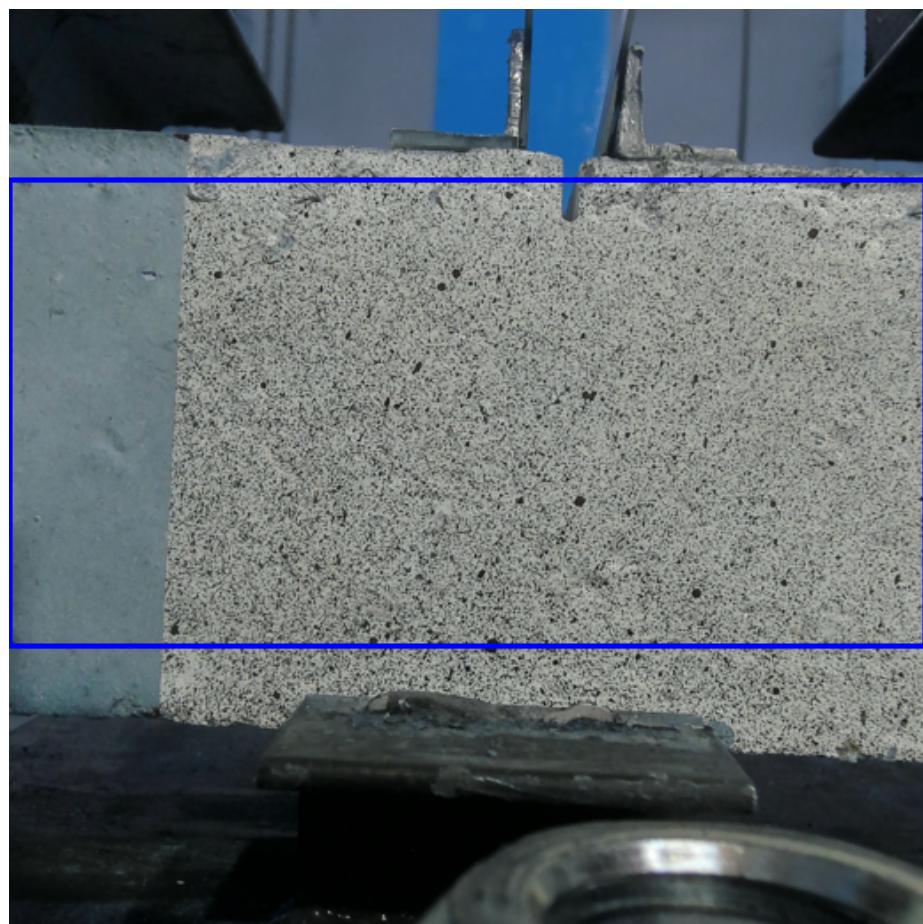


Figura 3.26: Zona seleccionada como ROI.

## Detección de zonas

Una vez tenemos la máscara preparada, lo siguiente que hacemos es seleccionar las zonas de las máscaras que corresponden a grietas. Es decir, agrupamos en zonas los píxeles de la máscaras que corresponden a una zona común.

Para realizar este proceso, se realiza un proceso de conectar los píxeles entre sí dándoles una etiqueta de zona, de tal forma que todos los píxeles que pertenezcan a una misma zona se marcarán con la misma etiqueta.

El criterio por el cual se decide que un píxel pertenece a una zona o no consiste es el de vecindad. Si un píxel está adyacente a otro en alguna de las ocho posibilidades de ser vecino (horizontal, vertical y las diagonales) entonces pertenece a la misma zona y se les aplica la misma etiqueta.

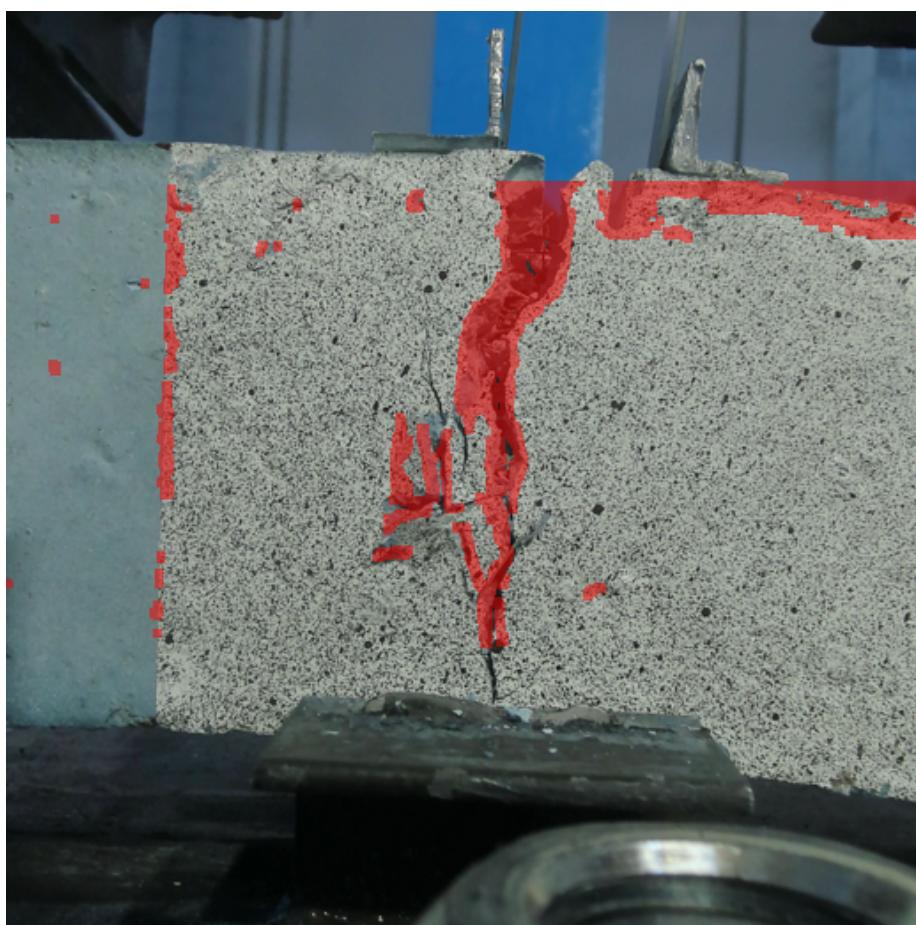


Figura 3.27: Zonas seleccionadas como grietas.

## Medición de distancias

Una vez que tenemos las zonas seleccionados, el último paso es medir la distancia entre los puntos. Para ello, vamos a medir las distancia entre los puntos de los bordes de las todas las distintas zonas.

Para medir la apertura de la grieta en cada fila seleccionada, se toman los píxeles extremos  $(x_0, y)$  y  $(x_1, y)$  y se transforman a coordenadas tridimensionales mediante la matriz `world_points`, obteniendo:

$$P_0 = \text{world\_points}[y, x_0], \quad P_1 = \text{world\_points}[y, x_1].$$

La distancia entre ambos puntos se calcula como la norma euclídea en tres dimensiones:

$$d = \|P_1 - P_0\| = \sqrt{(X_1 - X_0)^2 + (Y_1 - Y_0)^2 + (Z_1 - Z_0)^2}.$$

Para elegir que puntos representar, se ha elegido dos criterios. Representar en cada zona el punto con la mayor distancia, y para que no se tapen al representarlos unos a otros, los siguientes puntos que se representarán son lo que tengan mayor distancia pero guarden una separación vertical mínima entre ellos.

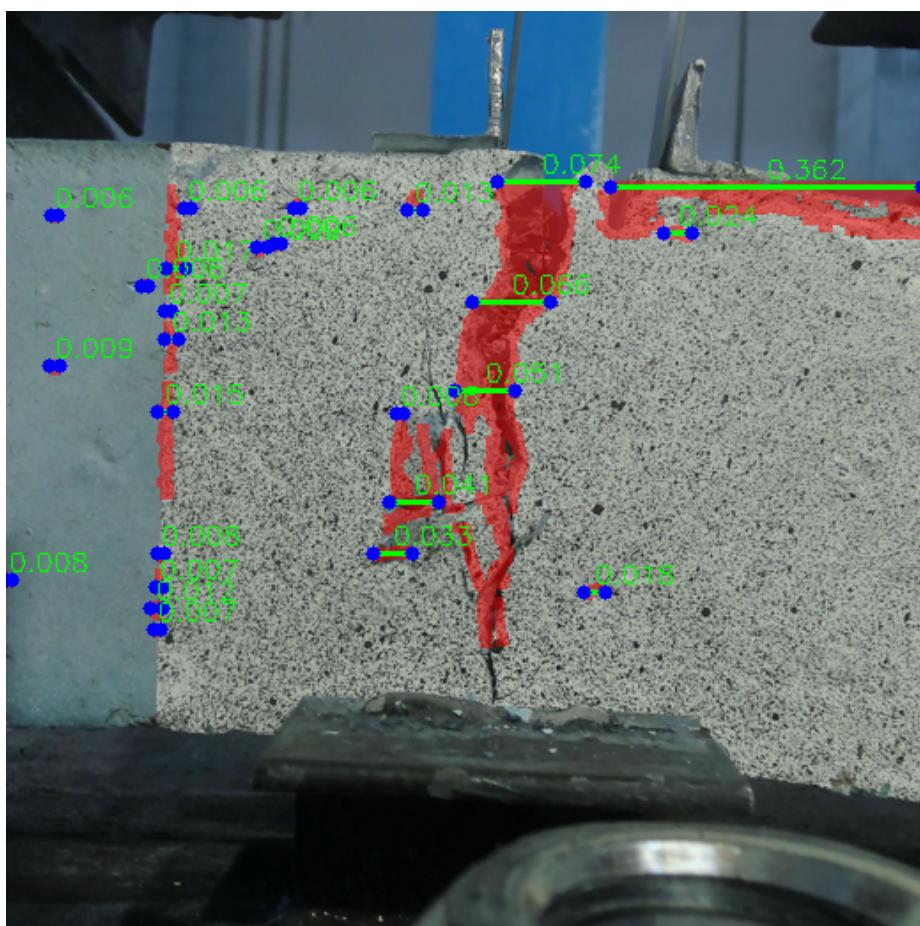


Figura 3.28: Distancias entre los puntos indicando el tamaño de las grietas detectadas.



---

# Técnicas y herramientas

---

## 4.1. MatLab

Se ha usado MatLab para el rectificado de imágenes usando las librerías que proporciona. Se han usado los parámetros de calibración estéreo calculados previamente y que fueron proporcionados en conjunto a los pares de imágenes. Contienen la información de las cámaras tanto intrínsecos como extrínsecos.

Con esos parámetros, Matlab ofrece una función llamada `rectifyStereoImages`. Esta función es la que hace el trabajo de rectificación. Toma las dos imágenes originales y las transforma para que sus líneas epipolares queden alineadas en horizontal. Esto significa que cualquier punto de una imagen que tenga correspondencia en la otra estará en la misma fila.

## 4.2. Python y Conda

Se ha usado Python y la creación de entornos con Conda para la programación. Cada modelo necesita una serie de dependencias que no son compatibles entre sí, así que se ha creado para cada modelo su propio entorno.

Se han usado las herramientas y librerías que proporciona Python para los detectores de líneas y de esquinas. En concreto se ha usado la librería OpenCV que proporciona una implementación de los detectores de líneas y esquinas más importantes

### **4.3. PyQt5 y VTK**

Se ha usado la librería PyQt5 para la creación de la interfaz de usuario de la aplicación, ya que proporciona un conjunto de elementos gráficos que nos permiten representar fácilmente las imágenes y el resto de elementos gráficos. Además se integra fácilmente con el resto de elementos que hemos usado en Python.

Para la representación 3D hemos usado VTK porque nos ofrecía la ventaja de que podía usarse de forma directa dentro de la aplicación y comunicarse con el resto de elementos, con lo que no dependíamos de ventanas externas y de buscar mecanismos de comunicación entre aplicaciones. Esto es una característica que no nos ofrecía Open3d, por ejemplo.

---

# **Conclusiones y Líneas de trabajo futuras**

---

## **7.1. Conclusiones**

Los resultados obtenidos nos muestran que es posible partiendo de un par de imágenes estéreo, calcular la profundidad y ser capaces de detectar los cambios que se producen, en nuestro caso, en un bloque de homrigon al que se le somete a presión y analizar el comportamiento de los materiales.

Pero aún queda mucho trabajo que hacer. Uno de los mayores problemas es que al usar modelos generalistas, estos están diseñados para obtener la profundidad en general de una escena, mientras que lo que estamos buscando con este experimento es buscar los detalles más precisos y concretos de una zona delimitada, y no buscar la representación de la escena en su global.

Los resultados reflejan este comportamiento. Mientras que la representación de la escena como conjunto es muy buena, cuando bajamos a los detalles para detectar la grieta, falta la precisión necesaria para llevar a cabo un estudio exhaustivo.

No es que los modelos no detecten que se produce un cambio de profundidad en esas zona, es simplemente que no lo detectan con el detalle necesario.

Esto es un problema por dos vías. La primera es que se hace difícil obtener valores consistentes de profundidad en una zona concreta, y en nuestro caso, en la zona de la grieta. Y la segunda, que viene derivada de la primera, es que sin valores consistentes no se puede hacer un seguimiento de la zona para ver y estudiar su progresión.

Pero a pesar de esto, los resultados también muestran que la detección es posible, y que lo que hace falta es refinar el procedimiento para que se mejore la detección de los detalles más concretos. Y con ello se podría realizar el segundo paso que es hacer el seguimiento a través del tiempo con un conjuntos de pares de imágenes.

En la sección de Líneas de trabajo futuras comentaremos algunas de las posibles soluciones para solventar este problema.

## 7.2. Líneas de trabajo futuras

### Unity y conjunto sintético de entrenamiento

Como los modelos generalistas se centran en una parte del problema que no es la que necesitamos, una posible solución sería generar un modelo, ya bien puede ser desde cero o través de alguna técnica de "fine tunning" sobre un modelo anterior, que se especialice en detectar la profundidad en zonas concretas de las imagen en vez de en la escena global.

Para generar este modelo, necesitaríamos un conjunto de imágenes de entrenamiento con los valores de "ground-truth" correctos sobre la posición y profundidad de las grietas. Algo que no es fácil de obtener en el mundo real.

Por lo que hemos considerado la posibilidad de entrenar un modelos usando un conjunto de datos sintéticos generados a partir de escenas 3D creadas en Unity. Esto es algo que ya se ha utilizado en algunos de los modelos que se han probado, como en Foundation Stereo.

El ser escenas 3D de las que tenemos el control total, se podría obtener los valores de profundidad y posición exactos, o entrenar al modelos con distintos valores intrínsecos y extrínsecos de las cámaras variando su posición en el espacio. O otra combinación necesaria de luz, por ejemplo. El mayor problema sería capturar las condiciones cambiantes que se dan en el mundo real, con lo que habría que explorar la posibilidad de que partiendo de un modelo que ya haya aprendido a analizar escenas reales, mejorar su precisión en los detalles que buscamos con el conjunto sintético de imágenes.

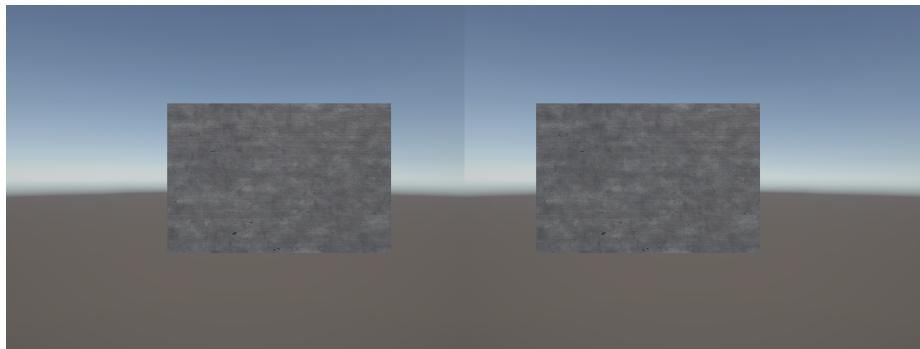


Figura 7.29: Par de imágenes estéreo generadas en Unity, con un patrón aleatorio, para simular un bloque de hormigón

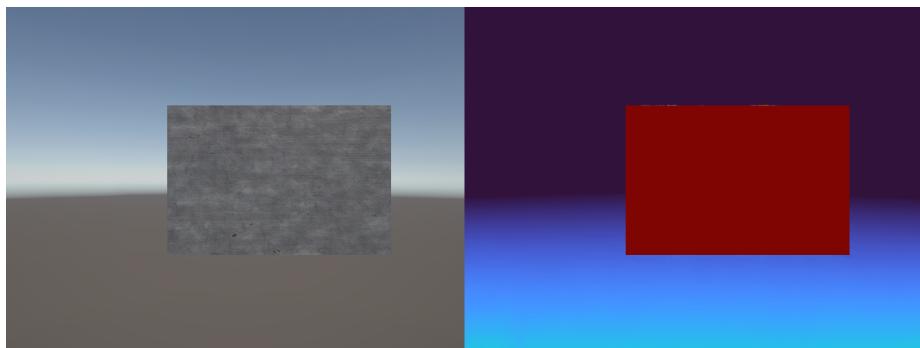


Figura 7.30: Resultados de profundidad obtenidos al aplicar Foundation Stereo en el par generado con Unity

## Modelo entrenado con secuencia de imágenes

Cuando planteamos el caso de entrenar un modelo nuevo, surje la necesidad de que lo que realmente estamos buscando no es sólo detectar la grieta, si no que es su evolución a lo largo del tiempo.

La idea que se podría plantear es no entrenar a un modelo sólo con un par de imágenes como entrada, si no con una secuencia completa de imágenes. De esta forma, que sea el modelo el que aprenda de a analizar como se forma la grieta y como evoluciona a lo largo de las imágenes.

Esto no es algo nuevo, y hemos visto modelos que ya hacen algo parecido en cuanto al análisis o generación de imágenes. En concreto, nVidia [12] en su modelo para DLSS 3, usa dos imágenes y la información de los vectores de movimiento del motor 3D para generar un nuevo frame.

Aplicando esta idea en nuestro caso concreto, se podrían usar varios pares de imágenes para que analizara las diferencias entre ellos. Y en el caso que se usara un conjunto de datos sintéticos, se podrían usar la información de movimiento para mejorar la predicción. Aunque en este último caso sería necesario estudiar si sería posible obtener esta información en los experimentos del mundo real.

## **Usar vggT y su seguimiento de puntos**

Una de las características de VGGT es que es capaz de hacer seguimiento de puntos en el espacio tridimensional a través de una secuencia de imágenes. El problema para aplicarlo a nuestro caso concreto, a parte de las limitaciones que hemos visto de ser un modelo generalista, es que VGGT por sí mismo no es capaz de detectar ningún punto.

Esto quiere decir que debemos ser nosotros lo que le digamos previamente sobre qué puntos debe intentar realizar el seguimiento. El problema es que no sabemos dónde se va a producir la grieta, así que dado el caso en que VGGT fuera capaz de hacer el seguimiento necesario en nuestro caso concreto, el problema cambiaría a realizar primero una detección de algún punto o puntos de interés que nos de información sobre la evolución del bloque de hormigón y de la grieta, y darle estos puntos a VGGT para que realice el seguimiento y ver su evolución a lo largo del tiempo.

Esto se podría hacer manualmente en un primer paso para probar cómo es capaz de hacer el seguimiento VGGT, y posteriormente intentar que de una forma automática se seleccionen estos puntos.

# Apéndices



## *Apéndice A*

---

# **Documentación técnica de programación**

---

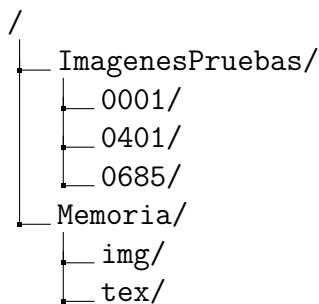
## **A.1. Introducción**

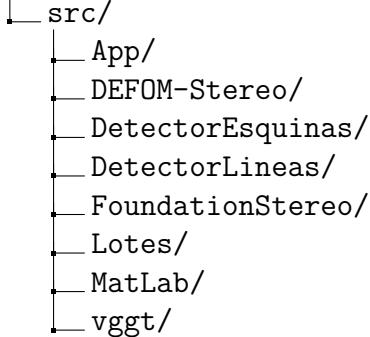
En esta sección vamos a dar las instrucciones de como poder usar la aplicación que se ha creado para visualizar en 3D el par de imágenes y obtener la predicción de las grietas y su tamaño relativo. Esta aplicación usa el modelo VGGT.

## **A.2. Estructura de directorios**

El código de la aplicación se encuentra en src/App, donde hay un script para su ejecución.

En el resto de directorios se encuentran los modelos por separados con script para su ejecución, y en el directorio de imágenes de pruebas hay tres pares de imágenes representativas del inicio, mitad y final del conjuntos de imágenes usadas para poder hacer las pruebas necesarias.





### A.3. Manual del programador

El programa está creado en Python por lo que debería ser portable a cualquier sistema, pero para su prueba y desarrollo se ha usado un sistema operativo Linux, con lo que se usarán los comandos y scripts adaptados a este sistema.

Se ha usado y se recomienda hacer un entorno de Conda para ejecutar la aplicación. De esta forma se evitan los problemas de dependencias que pueden surgir con Python. Para ello se puede usar el comando

```
conda create -n vggt python=3.10
```

Para crear el entorno con todas las dependencias necesarias se puede usar el archivo environment.yml con el siguiente comando.

```
conda env create -f environment.yml
```

Es necesario que se mantenga la estructura de directorios, con el directorio Modelos donde se ejecute la aplicación, para que pueda encontrar los archivos necesarios de VGGT.

El archivo model.pt que se puede encontrar en **VGGT model.pt** tiene que estar en el directorio ./Modelos/vggt/

### A.4. Compilación, instalación y ejecución del proyecto

Una vez que tengamos el entorno creado con las todas las dependencias, para ejecutarlo se puede usar el script creado para ello con el siguiente comando

ejecutar\_app.sh

O de forma manual con

```
conda activate vggt
python main.py
conda deactivate
```



## *Apéndice B*

---

# **Documentación de usuario**

---

## **B.1. Introducción**

En esta sección se explicará el uso de la aplicación para visualizar un par de imágenes estéreo y generar la predicción de las zonas de la grieta y su distancia

## **B.2. Requisitos de usuarios**

Tener instalado Python 3. Y se recomienda tener instalado Conda para crear un entorno donde instalar todas las dependencias de Python.

## **B.3. Instalación**

Para instalarlo se necesita clonar el repositorio o descargar el zip desde <https://github.com/vperee00/Medida-Rotura-3D-Hormig-n>

Para crear el entorno con todas las dependencias necesarias se puede usar el archivo environment.yml con el siguiente comando.

```
conda env create -f environment.yml
```

Y descargar el archivo del modelo se puede encontrar en [VGGT model.pt](#) en el directorio ./Modelos/vggt/

## B.4. Manual del usuario

Para ejecutar la aplicación se puede usar el script creado para ese efecto  
 ejecutar\_app.sh

O de forma manual con

```
conda activate vggt
python main.py
conda deactivate
```

### Uso de la aplicación

La aplicación consta de varias secciones. La barra de menú tiene el menú Archivo con la opción de Cargar 3D y de Salir. El menú Modelos con VGGT, que en el futuro se podría dar la opción a más modelos. Y el botón Ejecutar que comenzará todo el proceso de análisis del par de imágenes.

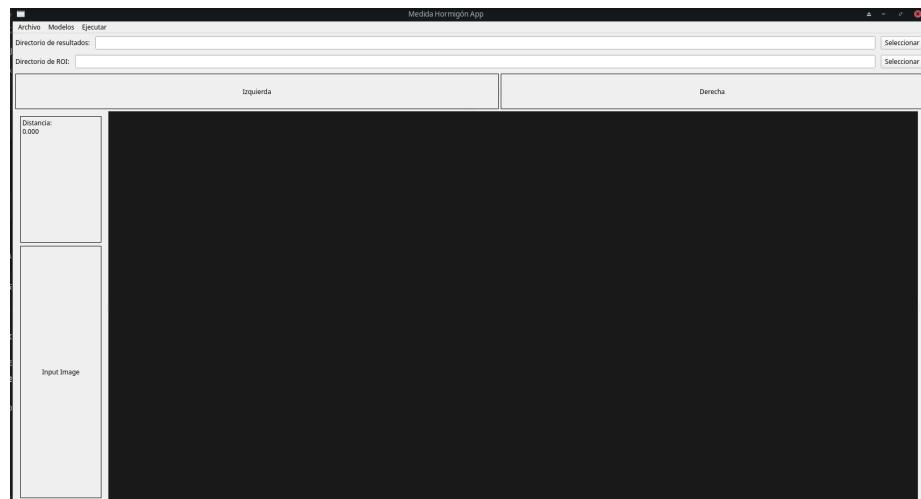


Figura B.1: Pantalla inicial

En el campo de texto de Directorio de Resultados se selecciona donde queremos que guarde todos los archivos generados. También sirve si ya los hemos generados anteriormente, seleccionamos el directorio y con la opción de Archivo -> Cargar 3D, se mostrarán los resultados sin tener que volver a generarlos.

El campo de texto de Directorio ROI se usa para seleccionar un directorio de resultados, solo que en esos resultados se usarán para seleccionar el área

de ROI. Si no se selecciona nada, se usarán los resultados del primer par de imágenes que ya se encuentran generados para ser usados por defecto.

Los botones Izquierda y Derecha se usan para seleccionar el par de imágenes respectivamente.

Una vez que hayamos seleccionado las imágenes y un directorio donde guardar los resultados, con el botón Ejecutar se procesaran las imágenes y se mostrarán los resultados.

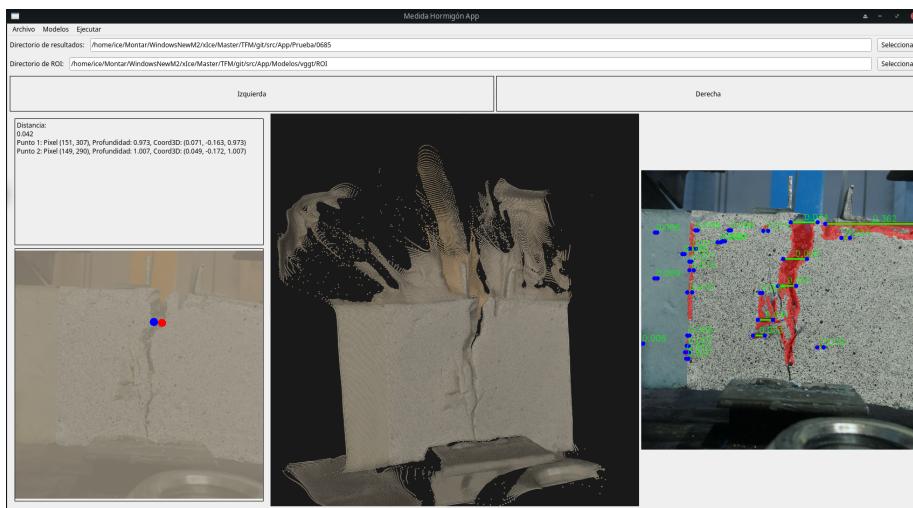


Figura B.2: Pantalla con los resultados

En la sección central se mostrará la vista en 3D del par de imágenes estereóeo, donde se puede usar el ratón para mover el modelo y la ruleta para el zoom. Al hacer click en esta zona se marcará ese punto alternativamente en la imagen de la parte izquierda, y en la zona superior tendremos información de los puntos seleccionados manualmente.

En la parte derecha se muestra la imagen con las predicciones. Las zonas en rojo corresponden a las zonas donde se ha detectado una grieta, y las líneas verdes unen puntos indicando la distancia relativa entre ellos.



---

## Bibliografía

---

- [1] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. *arXiv preprint arXiv:1712.07629*, 2018.
- [2] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.
- [3] Mihai Dusmanu, Ignacio Rocco, Tomáš Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8092–8101, 2019.
- [4] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988. Alvey Vision Club.
- [5] Anders Heyden and Marc Pollefeys. Multiple view geometry. In *Proc. IEEE/CVPR*, pages 45–60, 2001.
- [6] Hualie Jiang, Zhiqiang Lou, Laiyan Ding, Rui Xu, Minglang Tan, Wenjie Jiang, and Rui Huang. Defom-stereo: Depth foundation model based stereo matching. *arXiv preprint arXiv:2501.09466*, 2025.
- [7] A.-M. Khan, M. S.-H. Kee, K. Pathan, A.-S. and A.-A. Nahid. Image processing techniques for concrete crack detection: A scientometrics literature review. *Remote Sens.*, 15(9):2400, 2023.

- [8] Nahum Kiryati, Yuval Eldar, and Alfred M. Bruckstein. A probabilistic hough transform. *Pattern Recognition*, 24(4):303–316, 1991.
- [9] Didier Lootens, Marc Schumacher, Maxime Liard, Scott Z. Jones, Dale P. Bentz, Stefano Ricci, and Valentino Meacci. Continuous strength measurements of cement pastes and concretes by the ultrasonic wave reflection method. *Construction and Building Materials*, 242:117902, 2020.
- [10] Jiří Matas, Csaba Galambos, and Josef Kittler. Progressive probabilistic hough transform. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 256–265. BMVA Press, 2000.
- [11] Álvaro Mena-Alonso, Pedro Latorre-Carmona, Dorys C. González, José F. Díez-Pastor, Juan J. Rodríguez, Jesús Minguez, and Miguel A. Vicente. A cost-effective stereo camera-based system for measuring crack propagation in fibre-reinforced concrete. *Archives of Civil and Mechanical Engineering*, 23:1–12, 2023.
- [12] NVIDIA Corporation. Nvidia ada science: How ada advances the science of graphics with dlss 3. Technical report, NVIDIA, 2022.
- [13] Jérôme Revaud, Philippe Weinzaepfel, César De Souza, Noé Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: Repeatable and reliable detector and descriptor. *arXiv preprint arXiv:1906.06195*, 2019.
- [14] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)*, volume 1, pages 430–443. Springer, 2006.
- [15] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.
- [16] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600. IEEE, 1994.
- [17] Irwin Sobel. Camera models and machine perception. In *Proceedings of the Stanford Artificial Intelligence Project*, pages 271–272. Stanford University, Stanford, USA, 1973. Introduced the Sobel operator for edge detection.

- [18] Richard S. Stephens. Probabilistic approach to the hough transform. *Image and Vision Computing*, 9(1):66–71, 1991.
- [19] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.
- [20] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.
- [21] J. Zhang, L. Peng, S. Wen, and S. Huang. A review on concrete structural properties and damage evolution monitoring techniques. *Sensors*, 24(2):620, January 2024.