POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

AI-Driven Feature Detection, Matching, and Efficient Model Deployment for Space applications

Supervisors

Prof. Daniele JAHIER PAGLIARI

Prof. Alessio BURRELLO

Dr. Matteo STOISA - AIKO S.R.L.

Dr. Armando LA ROCCA - AIKO S.R.L.

Candidate

Alessandro FICCA (319084)

April 2025

Acknowledgements

Al termine di questo importante capitolo, colgo l'occasione per esprimere i miei ringraziamenti. Desidero ringraziare in primis la mia famiglia. Per avermi sempre mostrato amore e supporto incondizionato in ogni momento e decisione della mia vita.

Vorrei ringraziare l'azienda AIKO per avermi dato la possibilità di svolgere questo progetto. In particolare, Armando e Matteo per il loro aiuto e la loro disponibilità durante l'intero percorso.

Ringrazio inoltre il prof. Pagliari e il dott. Burrello per il loro supporto accademico.

Concludo, ringraziando con estremo affetto tutte le persone che mi sono state vicine e continuano a far parte della mia vita.

Table of Contents

1 Introduction 1

2 Background and Related Works 4
- 2.1 Introduction to Space Visual Navigation . . . 4
  - 2.1.1 Applications . . . 6
  - 2.1.2 Challenges of Space domains . . . 7
- 2.2 Deep Learning . . . 8
  - 2.2.1 MultiLayer Perceptron . . . 9
  - 2.2.2 Convolutional Neural Network . . . 10
  - 2.2.3 Transformer . . . 13
- 2.3 Knowledge Distillation . . . 16
  - 2.3.1 Knowledge . . . 16
  - 2.3.2 Distillation . . . 18
  - 2.3.3 Teacher-Student architecture . . . 19
- 2.4 Feature Detection and Matching . . . 20
  - 2.4.1 FDM: Classical algorithms . . . 21
- 2.5 FDM: Deep Learning models . . . 24
  - 2.5.1 FDM: Non-Transformer based models . . . 26
  - 2.5.2 FDM: Transformer-based models . . . 28
- 2.6 Comparison between classical algorithms and Deep Learning models 31
- 2.7 Metrics . . . 32
  - 2.7.1 Homography Estimation . . . 33
  - 2.7.2 Epipolar Geometry and Relative Camera Pose Estimation . 35

3 Space datasets and benchmark datasets 39
- 3.1 Space datasets . . . 39
  - 3.1.1 SPEED . . . 39
  - 3.1.2 SPEED+ . . . 41
  - 3.1.3 SPEED-UE-Cube . . . 43
- 3.2 Benchmark datasets . . . 45
  - 3.2.1 HPatches . . . 45

3.2.2 ScanNet . . . 46

4 Methods 48
4.1 Data preprocessing . . . 49
4.1.1 Homography generations . . . 50
4.1.2 Image pair generation from relative camera pose . . . 53
4.1.3 Trajectory set . . . 57
4.2 Training model with KD . . . 57
4.2.1 Student model architecture . . . 58
4.2.2 Distillation . . . 61
4.2.3 Homography pre-training . . . 62
4.2.4 Finetuning . . . 63

5 Experimental results 66
5.1 Feature Detection and Matching Results . . . 67
5.1.1 Homography estimation . . . 68
5.1.2 Relative Camera Pose Estimation . . . 79
5.2 Knowledge Distillation results . . . 96
5.2.1 Training results . . . 96
5.2.2 Student vs Teacher model . . . 99
5.2.3 Performance and model size comparison . . . 104

6 Discussions and Conclusions 106
6.1 Discussions . . . 106
6.2 Conclusions . . . 109

Bibliography 110

Chapter 1

Introduction

Artificial Intelligence (AI) is undergoing rapid and large-scale advancements, becoming an integral part of modern technology and daily life. The potential applications are expanding rapidly, impacting an increasing number of sectors. Computer vision remains one of the most prominent fields among the various AI-driven applications. It refers to the processes that allow machines to perceive, interpret, and understand visual information from the world. It encompasses methods for acquiring, processing, and analysing digital images to extract data that contains meaningful information.

One of the most interesting and challenging environments where AI can potentially be applied is **Space**. Space missions involve extreme and unpredictable scenarios that require real-time information processing and decision-making. These operations demand high autonomy, precision, and resilience against external factors. Traditional space systems typically depend on communication with ground control, which can introduce latency and limits in data transmission, particularly in deep space, making timely responses hard. For this reason, there is an increasing need for intelligent on-board systems that can operate autonomously in complex and uncertain situations.

While AI is now commonly used in many terrestrial applications, its integration into space systems is still early. However, the shift from traditional to AI-based systems holds the potential to drive groundbreaking innovations and progress in the coming years. Autonomous satellite operations, on-board data processing and analysis, and smart predictive maintenance are just a few examples of critical space activities that could be enhanced by deploying efficient and intelligent Machine Learning and Deep Learning models in orbit.

This thesis focuses on the task of **Feature Detection and Matching (FDM)**,

an important component of many computer vision processes that has seen remarkable advancements in recent years. The objective of FDM is to establish relevant and accurate pixel-level correspondences between images that capture the same scene from different viewpoints. These correspondences are fundamental for estimating the camera's motion around the scene and are used in applications such as autonomous navigation, image registration, pose estimation, and Simultaneous Localization and Mapping (SLAM). FDM becomes particularly critical in scenarios where only visual data is available, offering a viable alternative to active sensors like LiDAR or infrared, especially in systems where resource constraints limit the use of such technologies.

This project aims to analyse and compare different FDM methods, originally designed and tested for general applications, to assess their abilities and limitations under challenging conditions typical of space images.
The study explores the possible application to Rendezvous and Proximity Operations (RPO) scenarios from a visual perspective. In this operation, a chaser spacecraft gathers and analyses visual data from a target spacecraft. This generally occurs when the two distinct space objects are close enough, in the order of tens of meters. The relevant features extracted and matched between consecutive target images could be used inside a SLAM process to track the target's movements and localise it within the environment, enabling precise relative navigation and control.

The thesis carried out followed four main steps:

- **Literature review**: The first phase of the project gives a review of research in Feature Detection and Matching. The fundamental concepts and classical algorithms are examined alongside advanced Deep Learning-based models. It evaluates metrics applicable to FDM methods based on the available data and investigates the primary concepts of the Knowledge Distillation (KD) technique.
- **Dataset analysis and preprocessing**: The second phase analyzes space-related datasets, specifically SPEED[1], SPEED+ [2] and SPEED-UE-Cube [3], selected for their relevance in evaluating RPO conditions. In addition, benchmark datasets, HPathes [4] and ScanNet [5] are examined.
- **Comparison of classical algorithms and Deep Learning models**: The third phase benchmarks deep learning models against traditional FDM algorithms. The evaluation focuses on their performance in challenging scenarios, such as those represented in the selected datasets, to assess their robustness and adaptability to space-related image characteristics.
- **Application of Knowledge Distillation**: The final phase involves training a

Deep Learning model through Knowledge Distillation. The primary objective is to transfer knowledge from a Transformer-based model to a Convolutional architecture motivated by hardware-specific constraints.

The following chapters analyse the described stages and present the experimental results. In particular, Chapter 2 provides an overview of FDM and its space applications pertinent to the project. Chapter 3 details the adopted datasets for the subsequent steps. Chapter 4 discusses the various steps and methods employed throughout the project. Chapter 5 examines the experimental results obtained. Chapter 6 concludes the work with a comprehensive analysis and discussions.

Chapter 2

Background and Related Works

2.1 Introduction to Space Visual Navigation

Optical navigation (ON) has always been relevant to space missions and advancements in computer vision have greatly enhanced its capabilities [6]. It is defined as the use of pictures taken by on-board cameras to help determine the spacecraft's trajectory [6]. The images can depict planets, asteroids, orbital debris, or other spacecrafts. From these visual data, it is possible to infer information about the state of the observing spacecraft and the properties of the observed target. This information can aid in the navigation process.

Serving either as a complement to traditional navigation methods or as an alternative, ON has become a key technology for satellite autonomous navigation associated with different navigation approaches such as cruising, fly-by, terrain relative navigation, landing, rendezvous and docking between spacecrafts (RVD) and proximity operations (RPO) [7]. These capabilities can be used for addressing critical missions like Active Debris Removal (ADR) [8] and On-Orbit Servicing (OOS) [9], especially when dealing with non-cooperative targets.
ADR aims to mitigate the growing problem of space debris in Low-Earth Orbit, avoiding future collisions and reducing the risk for operative spacecraft. OOS includes the inspection and maintenance of orbiting active satellites and operational spacecraft, potentially extending their life. Another relevant application is autonomous target exploration in deep space, where spacecraft must operate independently due to limited or delayed communication with Earth. In such situations, optical navigation enables the spacecraft to localize itself and make navigation decisions without relying on ground-based control. These scenarios illustrate the

importance of vision-based autonomy in modern spaceflight. As missions become more complex and communication delays more restrictive, particularly in deep space, the role of optical navigation continues to grow, supporting safer, more efficient and autonomous spacecraft operations.

One of ON's main advantages is its use of passive, low-power image sensors, such as cameras, instead of active and energy-intensive sensors like LiDAR (Light Detection and Ranging) and RADAR (Radio Detection and Ranging).
Cameras are typically smaller, lighter, and more cost-effective, making them ideal for space systems where power consumption, mass, and payload volume are important constraints. Active sensors tend to generate large volumes of data that are computationally demanding to process in real time and difficult to transmit to Earth due to bandwidth limitations. In contrast, image data is generally more compact and can be processed on-board with lower computational requirements, leading to faster processing and easier transmission.

A specific use case is the **relative visual navigation** in RPO. It refers to spaceflight scenarios in which two actors interact: the *chaser* and the *target*. The chaser vehicle is a spacecraft equipped with both attitude and translational control capabilities. It is able to acquire relative navigational information with respect to the target and decide how to manoeuvre accordingly [10]. The target can take various forms, such as space debris, asteroids or other spacecraft, and it can be classified based on its behaviour and design [11] :

- *Cooperative*: targets can communicate with the chaser and provide state information such as position and attitude. It can be the case of OOS operations. However, target satellites may become inoperative or experience system failures, rendering them unable to communicate.
- *Non-cooperative*: targets do not assist the chaser in any navigation phase. The chaser has to autonomously estimate the state of the objective. It is typical of space debris in ADR missions.
- *Prepared*: Designed for rendezvous operations and equipped with features such as visual markers, docking fixtures, or navigation aids.
- *Non-prepared*: Lacking any of these equipments

Additionally, the target can be *tumbling* or *non-tumbling*. In the first case, the object is in an uncontrolled rotational motion, and its spin may be unpredictable, creating difficulties for operations. The second case describes a stable state where the object's orientation is controlled and predictable.

The situation analyzed in this project is categorized as a proximity operation involving a non-cooperative spacecraft, which is a generic scenario of the spacecraft's Guidance, Control, and Navigation (GNC) system. The chaser captures images of the target using a monocular on-board camera and performs relative navigation.

The complete on-board navigation process is a complex, multi-stage pipeline involving several components that are responsible for selecting the optimal rate for image acquisition. Informative frames are identified and retained, while redundant or low-quality images are discarded. This work does not cover the entire navigation process. Instead, it assumes that a suitable pair of images has already been acquired and processed. The focus is on the subsequent algorithmic stages, which include **feature points detection**, **feature matching** and the estimation of the geometric transformation. Various boundary conditions that affect the performance of the feature detection and matching phases, as well as the estimation process, are analyzed.

2.1.1 Applications

The pixel correspondences produced in the FDM process serve as input for higher-level computer vision tasks used in visual navigation. The matched keypoints enable the reconstruction of the geometric relationship between different views and support advanced techniques such as Visual Odometry, Visual Simultaneous Localization and Mapping, and Structure from Motion (SfM).

Visual Odometry (VO) is the process of estimating the pose (i.e. position and orientation) and motion of the camera from a sequence of images captured during the trajectory. In a traditional VO pipeline, this is achieved by using feature-based methods, which detect interest points and track them across consecutive frames. The correspondences found allow us to find the geometric structure, such as the relative camera pose between pairs of images. Analyzing the whole sequence, the information estimated is accumulated to reconstruct the full camera motion [12] [13].

Visual Simultaneous Localization and Mapping (V-SLAM) derives from Simultaneous Localization and Mapping (SLAM). It refers to the process of localizing the position of a camera relative to its surroundings while simultaneously creating a map of the environment. In the V-SLAM, cameras are the main sources and information is extracted from monocular, stereo and RGB-D images. The VO is involved in estimating the camera motion and reconstructing the scene [14]. Then, the trajectory is optimized to reduce the error accumulated, and the 3D map is built by using techniques such as bundle adjustment [15].

Structure from Motion (SfM) refers to the process of creating a 3D model of an object from different views. This reconstruction typically requires detecting feature points on the object and matching them in distinct pictures. Incorrect correspondences are filtered out, while the correct ones are used to estimate the camera's motion around the target and accurately determine the 3D coordinates of each point. An interesting application of SfM in space is the estimation of the shape of unknown space debris, as illustrated in Figure 2.1.

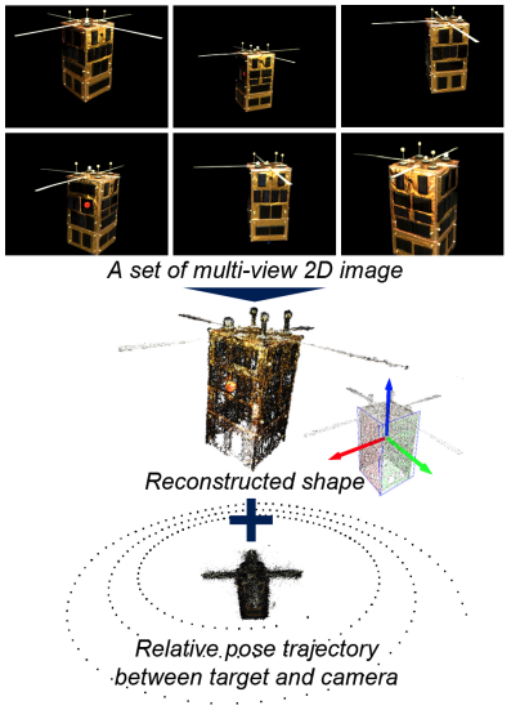

Figure 2.1: Reconstruction of unknown space debris trough SfM techniques [16]

2.1.2 Challenges of Space domains

Space scenarios have unique characteristics that distinguish them from ground-based and more generic visual contexts. These factors can influence the efficacy and limit the potential application of FDM methods. Some of the most critical characteristics that emerge in space images are:

- **Illumination variability**: Space images cover a high range of lighting conditions, from complete darkness to intense sunlight. This causes overexposure and underexposure in different regions of the image. The variations may occur in consecutive frames, making the process of multiple images more difficult.

- **Lack of Texture**: Frame captures in space often display large and uniform areas that show little or no variation in texture or visual information. For instance, objects such as spacecraft, asteroids, or debris are commonly set against a black background, and the surface of a planet may have very few distinctive points. Unlike generic environments, which are rich in visual details, the lack of variation in space makes it challenging to extract relevant and repeatable features.

- **Influence of the background**: Some space scenes may appear clean, while others include stars, Earth, and various backgrounds. They can distract in extracting meaningful features from the foreground object, which is the target for the relative navigation process.

- **Target's motion**: The motion of the spacecraft relative to the camera can create complications. In proximity operations or rendezvous scenarios, the relative distance and orientation can change significantly due to the high speed and unpredictable direction and spin of the target. As a result, the apparent size of the object in the image may vary rapidly. This leads to consecutive frames captured from different poses, reducing visual overlap and the number of detectable features.

- **Sensor-related limitations**: Space-grade cameras often face strict power and hardware limitations, which can lead to lower-resolution images, a limited dynamic range and increased noise. Environmental factors such as radiation can deteriorate sensor performance over time. Other issues, such as motion blur, lens artifacts, and image compression can also negatively affect image quality.

- **Lack of space-borne images**: One significant challenge is the lack of a large quantity of authentic space images that can be used to train and evaluate FDM models in space scenarios. While synthetic datasets provide an alternative, they sometimes fail to capture the complexities of real-world domains.

All these factors can make the application of computer vision techniques and models, particularly FDM methods, difficult.

2.2 Deep Learning

Although Artificial Intelligence (AI) and Machine Learning (ML) originated in the 19th century, they have only become popular and widespread in the last decade. This diffusion is particularly driven by the success of Deep Learning (DL), which has emerged as the core technology behind AI and is effectively incorporated into

numerous applications.

Deep learning is a subset of machine learning (ML) and artificial intelligence (AI) [17]. AI encompasses a broad research domain focused on integrating human behaviour and intelligence into machines or autonomous systems. ML is a more specialized field that utilizes algorithms and methods designed to learn directly from data. Deep learning is characterized by data-driven models that are based on the concept of **Artificial Neural Networks (ANNs)**. Within the deep learning framework, computations and processing are carried out through Deep Neural Networks, which consist of multiple layers that perform specific operations.

The model of Artificial Neural Networks has led to the development of a wide range of techniques and architectures within DL. At the base is the Multilayer Perceptron (MLP), also called the Feedforward Neural Network (FNN). Later, Convolutional Neural Networks (CNNs) became the standard and most adopted approach for numerous computer vision tasks. In recent years, Transformer-based architectures have emerged as the new state-of-the-art.

2.2.1 MultiLayer Perceptron

The **MultiLayer Perceptron (MLP)** is an architecture within ANNs. It takes a fixed-size input and produces a corresponding fixed-size output [18].
The MLP consists of multiple layers: The input layer, where the input variables $X = \{x_1, ..., x_n\}$ are introduced; the hidden layers that process and transform the data; and the output layer, which generate the final output of the model $Y = \{y_1, ..., y_m\}$. Figure 2.2 illustrates the structure of an MLP.

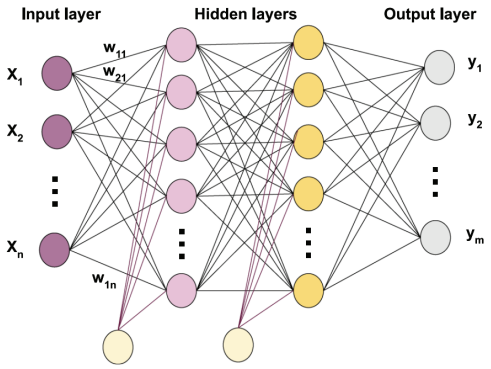

Figure 2.2: Topology of MLP [19]

Each layer contains a set of units, *neurons*. Each neuron computes a weighted sum of its input from the previous layers and applies a non-linear activation function. For a single neuron j, the corresponding scalar output y_j can be formulated as:

$$y_j = \sigma(\sum_i w_{ij} x_i + b_j) \tag{2.1}$$

where x_i are the input of the neuron, w_{ij} are the internal weights and b_j is the bias (called *parameters*). σ represents a non-linear activation function. Commonly used functions are ReLU (Rectified Linear Unit), ELU (Exponential Linear Unit), or Leaky ReLU and SiLU.

The strength of MLPs lies in their ability to introduce non-linearity into data processing. Research has demonstrated that an Artificial Neural Network with at least one hidden layer can approximate any continuous function with arbitrary precision [20]. This capability allows the MLP to learn complex patterns and effectively capture the underlying relationship between inputs and outputs.

ANNs can learn in a **supervised** manner, which means they need to know the desired output, referred to as the *target* or *ground truth*, associated with each corresponding input. The learning process involves minimizing a specific function, known as the *loss function*, which measures the error between the network's predictions and the actual targets. This is accomplished through the **Backpropagation** algorithm [21], which calculates the gradient of the loss function with respect to the internal weights and biases of the network. The model's parameters are then updated to reduce the error. In this way, the model learns to perform the task at its best.

2.2.2 Convolutional Neural Network

The **Convolutional Neural Network (CNN)** is a popular deep learning architecture that has become highly relevant due to its ability to learn directly from the input without the need for human feature extraction [22]. CNNs are specialized ANNs for processing data that can be represented through a grid-like topology, such as time-series data or images, represented as grids of pixels [23]. CNNs have become state-of-the-art approaches in many computer vision tasks, such as image classification and segmentation, object detection and many others.

The mathematical operation of **convolution** is the foundation of the convolutional network. This operation enables the detection of patterns, textures, and spatial relationships within the data, focusing on local connections.
The convolution operation involves sliding a small window, known as a kernel

or filter, across the input data. At each position, it computes the dot product between the values in the window and the corresponding input values. The kernel's parameters (weights) are learned during the training process to minimize the loss function.
For 2D input data I and a 2D filter K, the discrete convolution operation can be expressed as:

$$S(i,j) = \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} K(m,n) \cdot I(i+m, j+n). \tag{2.2}$$

where $S(i,j)$ is the output at position (i,j), referred to as *feature map*. The dimensions k_h and k_w represent the height and width of the kernel, respectively [24]. This formulation is given for a single input and single output channel. Convolutional layers, however, can operate over multiple channels.
For an input with C_{in} channels and a convolutional layer with D output channels, each feature map S_d is computed by summing over all input channels, each with its own kernel K_d:

$$S_d(i,j) = \sum_{m=0}^{k_h-1} \sum_{n=0}^{k_w-1} \sum_{c=0}^{C_{\text{in}}-1} K_d(c,m,n) \cdot I(c, i+m, j+n) \tag{2.3}$$

Figure 2.3 shows an example of the application of 2D convolution operation.

CNNs may differ in their architectures and sizes. However, the typical architecture, as illustrated in figure 2.4, comprises several distinct types of layers:

- **Convolutional layers**: They are responsible for performing the convolution operations mentioned earlier. Each of these layers has various hyperparameters that can be optimized according to the specific requirements of the task. The hyperparameters include the number of filters used, the dimensions of the filter window, the stride (which is the step size of the window), and the amount of padding applied.

- **Pooling layers**: They reduce the spatial dimensions of feature maps by performing a downsampling operation. The common pooling methods are *Max* and *Average* pooling, which calculate the maximum or average value in a local window.

- **Fully-Connected layers**: They utilize a flattened feature map, connecting each input to all neurons in the layer. These layers aggregate features learned from previous layers and generate predictions.

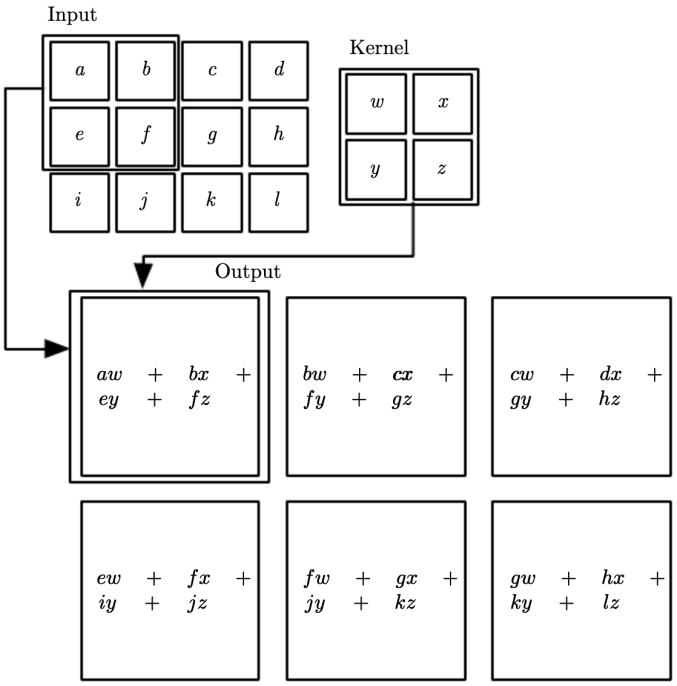

Figure 2.3: An example of 2-D convolution operation [23]

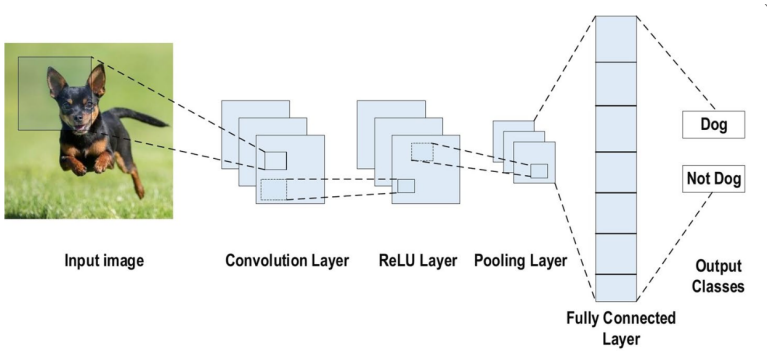

Figure 2.4: Example of CNN architecture [25]

2.2.3 Transformer

A **Transformer** model is a key deep-learning architecture that exploits the **Attention** mechanism to understand contextual relationships within sequential data. First introduced in 2017 [26], it has been widely adopted in Natural Language Processing, Computer Vision and speech processing.
Transformers generally work on sequence of elements, for example, *tokens* in NLP or image patches in computer vision, which are first transformed into a vector representation.

The core of the architecture is the attention module with the Query-Key-Value (QKV) representation [27]. The matrices query Q, keys K and values V are linear projections of the same input sequence, in the case of *Self-Attention*, or from different inputs, for the *Cross-Attention*.
Given the matrix representation of queries $Q \in \mathcal{R}^{N \times D_k}$, keys $K \in \mathcal{R}^{M \times D_k}$ and values $V \in \mathcal{R}^{M \times D_v}$, the scaled dot-product attention used by Transformer is

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_k}}\right)\mathbf{V} = \mathbf{A}\mathbf{V} \tag{2.4}$$

where N and M represent the lengths of the queries and key inputs, respectively. D_k and D_v denote the dimensions of the key (and query) and value vectors. Meanwhile, A is referred to as the *Attention matrix*, which captures the attention weights correlating the queries with the values through the key matrix.

Transformers use a Multi-Head Attention (MHA) scheme. The original input matrices Q, K, V of dimension D_m are linearly projected into subspaces of dimensions D_k, D_k, D_v with H different sets of learnable weights (W_i^Q, W_i^K, W_i^V). Each head independently computes the attention function. The outputs are concatenated and projected back to the original embedding dimension D_m with the weight matrix W^O. Figure 2.5 illustrates the computation graphs of the MHA. This is the mathematical expression:

$$\begin{aligned} \text{MHA}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}\left(\text{head}_1, \cdots, \text{head}_H\right)\mathbf{W}^O \\ \text{where} \;\; \text{head}_{\,i} &= \text{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V\right). \end{aligned} \tag{2.5}$$

The standard Transformer architecture consists of a series of encoder-decoder blocks, as illustrated in figure 2.6. Each encoder block processes a sequence of input tokens using a Self-Multi-Head Attention (Self-MHA) module and a Feed-Forward Network. In the Self-MHA module, each token attends to all other tokens, allowing it to learn dependencies between different positions in the sequence. In contrast, the decoder block includes an additional Cross-Multi-Head Attention (Cross-MHA)

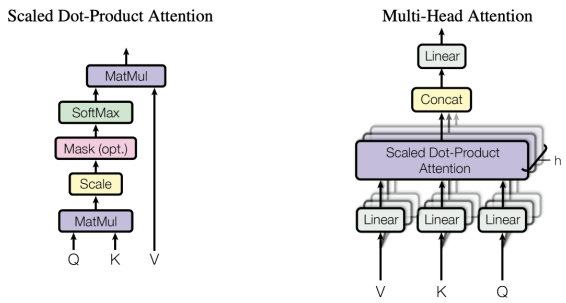

Figure 2.5: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention [26]

module. This module processes the query matrix Q from the decoder's previous layer output, along with the key K and value V matrices derived from the encoder's output.

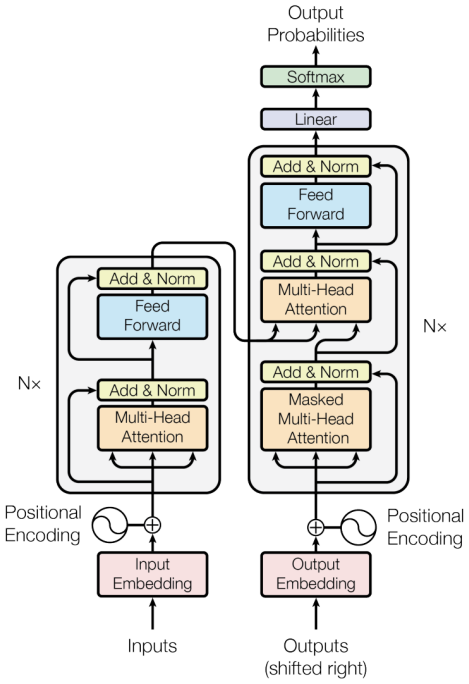

Figure 1: The Transformer - model architecture.

Figure 2.6: Transformer architecture [26]

Vision Transformer

After the success of the Transformer architecture in NLP, this architecture was transferred and adapted for computer vision tasks. One of the first and most important attempts was the **Vision Transformer (ViT)**, introduced in 2020 for image classification [28]. This work marked the relevance of the Attention operation and its use to replace CNNs.

The proposed architecture in ViT is an encoder-only Transformer, meaning it does not include a decoder module typically used in sequence generation. Figure 2.7 is an overview of the structure.

The model is designed to extract an image representation by transforming the image into a sequence of vector representations. In the process of image tokenization, the input image is divided into square patches. Each patch is flattened and then linearly projected into a fixed-dimensional embedding space. These embeddings are combined with a learnable positional encoding, which helps to convey the spatial relationships and the position of each patch within the input.
The transformer encoder processes the sequence of patch embeddings. This comprises multiple modules of Self-MHA and MLP. As made for token sequence, the Attention learns how different regions are correlated and the global context of the image. At the end, an MLP classification head predicts the image class based on the Transformer output.
From the ViT, many variants have emerged, including hybrid models that combine CNNs and Transformers.

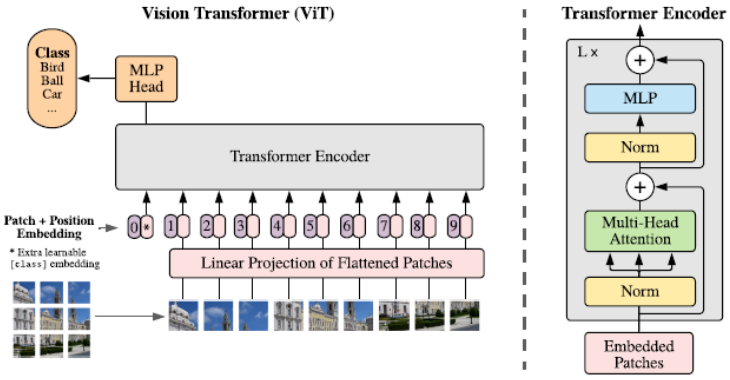

Figure 2.7: Overview of the Vision Transformer [28]

2.3 Knowledge Distillation

In recent years, deep learning neural networks have expanded in capability, size, and computational demands. Modern models often consist of billions of parameters and require substantial computing resources. While these large-scale models have achieved success across a wide range of applications, their complexity and storage requirements pose serious challenges for deployment in real-time scenarios and resource-constrained environments.
In this context, it becomes essential to explore efficient machine learning and deep learning techniques aimed at reducing model complexity and optimizing memory and energy usage. Various optimization strategies can be employed, including model compression methods such as quantization and pruning [29, 30], the adoption of efficient neural network layers and computational techniques [31], and the Knowledge Distillation (KD) technique [32].

This work specifically focuses on Knowledge Distillation, a technique that enables the transfer of knowledge between DL models with different architectures and dimensions. This approach is particularly relevant for space applications, where onboard hardware is typically not optimized to run state-of-the-art AI frameworks, often resulting in incompatibility or inefficiency. Furthermore, strict constraints on power consumption, memory, and communication bandwidth in space systems necessitate the use of smaller and more efficient models.

Knowledge distillation is a technique in which a primary deep neural network (*teacher*) distils knowledge to a smaller and optimised network (*student*). The student model can also present a different design and architecture in order to substitute specific layers and operations from the original model. The central idea is that, with the teacher's supervision during the training, the student can replicate the teacher's model performance despite the differences. This method was popularised by the paper [33].
The classical KD framework comprises the components: **Knowledge**, **Distillation Algorithms**, and **Teacher-Student architectures**. The following sections review the main aspects of this technique according to [32] [34].

2.3.1 Knowledge

Knowledge is essentially how the teacher model guides the learning of the student model. Different forms of knowledge can be used: response-based, feature-based, and relation-based. Figure 2.8 shows a scheme of the different categories.

In the **response-based** approach, the knowledge used derives from the *logits*,

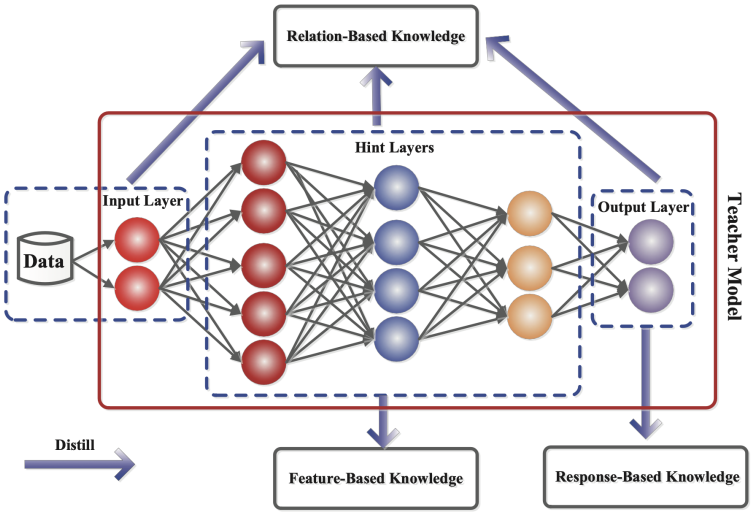

Figure 2.8: Scheme of the different sources of knowledge [32]

which are the output of the last layer of the teacher model. The student looks directly at the final prediction and tries to replicate it. The objective is to minimize the distance between the logits of the teacher and the student models.
Considered a vector of logits z as the output of the last Fully-Connected (FC) layer, the distillation loss can be formulated as:

$$L_{ResD}(z_t, z_s) = \mathcal{L}_R(z_t, z_s)$$

where $\mathcal{L}_R(.)$ indicates the divergence loss of teacher z_t and student z_s logits. The most popular approach for classification tasks uses *soft targets*. Soft targets are estimated from the logits using the softmax function, indicating the probability that the input belongs to a specific class. Soft targets are computed as

$$p\left(z_i, T\right) = \frac{\exp\left(z_i/T\right)}{\sum_j \exp\left(z_j/T\right)} \tag{2.6}$$

where z_i is the logit for the i-th class and T is a hyperparameter to control the importance of each of them.
Typically, the distillation loss is combined with the student loss, which is defined as the loss between the ground truth label and the soft logits of the student model. The Kullback-Leibler divergence loss is often utilised for the distillation loss, while the cross-entropy loss is used for the student loss.

In the **feature-based** approach, the output of the intermediate layer (i.e. feature maps) is used as the knowledge to supervise the student's training. The distillation loss for this case is:

$$L_{FeaD}(f_t(x), f_s(x)) = \mathcal{L}_F(f_t(x), f_s(x)) \tag{2.7}$$

$f_t(x)$ and $f_s(x)$ are the feature maps of the intermediate layers of the teacher and student, respectively, and $\mathcal{L}_F(\cdot)$ indicates the similarity function. Similarity measures can be used such as l_2-norm distance, l_1-norm distance and cross-entropy loss.

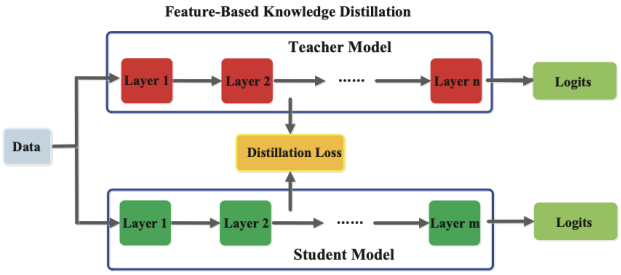

Figure 2.9: Generic feature-based knowledge distillation [32]

Relation-based knowledge explores the relationship between different layers or data samples. For example, it is possible to distil knowledge from the relationship between a pair of feature maps from the teacher, $\hat{f}_t$ and $\tilde{f}_t$, and from the student model, $\hat{f}_s$ and $\tilde{f}_s$. The loss is indicated as

$$L_{RelD}(f_t, f_s) = \mathcal{L}_{R^1}(\Psi_t(\hat{f}_t, \tilde{f}_t), \Psi_s(\hat{f}_s, \tilde{f}_s)) \tag{2.8}$$

where $\Psi_t(\cdot)$ and $\Psi_s(\cdot)$ are the similarity functions and $\mathcal{L}_{R^1}$ indicates the correlation function. This distillation strategy can also be based on the relationship between data samples in the feature space. The student tries to learn the teacher's understanding of data similarities.

$$L_{RelD}(F_t, F_s) = \mathcal{L}_{R^2}(\psi_t(t_i, t_j), \psi_s(s_i, s_j))$$

where (t_i, t_j) and $s_i, s_j)$ represent the data samples in the feature space, $\psi_s(\cdot)$ and $\psi_t(\cdot)$ similarity functions and $\mathcal{L}_{R^2}$ is the correlation function between the feature representations.

2.3.2 Distillation

The distillation scheme is the training procedure for both teacher and student models. There are three main categories:

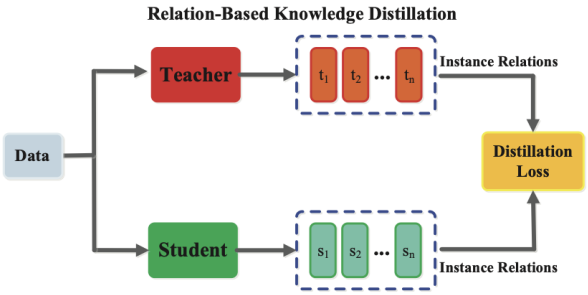

Figure 2.10: Generic instance relation-based KD [32]

- **Offline distillation**: The knowledge is transferred from a pre-trained teacher model into a student model without modifying or updating the teacher. This is the more relevant and established strategy.

- **Online distillation**: both the models are updated simultaneously. It is a one-phase end-to-end, trainable scheme.

- **Self-Distillation**: The same neural networks are used as teacher and student models. An example is [35], in which knowledge from the deeper layers is distilled into shallow layers. This method is less popular than the others.

2.3.3 Teacher-Student architecture

In KD, it is relevant to design proper teacher and student networks to improve the quality of the learning process. The model capacity gap can influence the knowledge transfer and the ability of the student to mimic the teacher. As said, this technique aims to reduce complexity, obtaining a simpler or smaller network than the original. So, the student can be:

1. A simplified version of the teacher network with fewer layers

2. A quantized version that preserves the same structure

3. A different architecture with efficient operations or an optimized global structure.

Many methods have been proposed to control the reduction of the model complexity and construct an adequate student network.

2.4 Feature Detection and Matching

Local feature detection and matching seeks to detect and establish precise correspondences between features in different images. A **feature** (or **interest point**) refers to a distinctive low-level structure within an image, such as specific points, regions, or patterns that remain identifiable across multiple perspectives.
To be useful for matching, interest points should possess specific properties: *uniqueness*, the feature should be distinguishable from others within the image; *repeatability*, since it should be detectable across different images of the same scene and *robustness*, to remain stable under transformations such as rotation, scaling, affine transformations and changes in illumination.
It is possible to classify distinct types of features:

- *Edges*: points that are a boundary between two image regions.
- *Corners*: pixels where the intensity varies rapidly in multiple directions within a local neighbourhood.
- *Blobs*: areas where pixels exhibit similar characteristics (such as brightness or colour) or patterns in relation to nearby regions.

The traditional image-matching pipeline comprises four steps [36]:

- **Feature detection**: Interest points are extracted from the image. These points should have the properties mentioned above to be useful for the matching phase.
- **Feature description**: Once detected, each feature is represented by a descriptor, a high-dimensional vector that encodes relevant information about the local image structure.
- **Feature Matching**: Descriptors enable comparisons between features in various images by assessing similarity measures or correlations.
- **Geometric transformation estimation**: The matches obtained can be used to estimate the geometric relationship between the image pair processed. Relevant geometric information about the structure of the scene can be retrieved.

These steps are not predefined but depend on the methods and approaches employed. With the advent of more advanced techniques, it has become possible to integrate multiple steps into unique frameworks. The following sections will provide an overview of classical approaches and deep learning-based models analysed in this work.

2.4.1 FDM: Classical algorithms

Classical handcrafted FDM algorithms are the foundation of computer vision and have been among the most crucial methods for analysing and extracting information and features from images. These algorithms are explicitly designed to identify and describe distinctive interest points based on mathematical principles, such as filtering techniques, gradient intensity analysis and scale-space representations. Starting with the Harris Corner Detector in 1988 [37], handcrafted approaches have shown substantial success in FDM tasks. In this work, some of the most popular traditional methods, including SIFT, ORB, and AKAZE, have been examined.

SIFT

Scale Invariant Feature Transform (SIFT) was proposed by David Lowe in 1999 [38] and then successively improved in 2004 [39]. It is considered the most renowned feature detection and description algorithm.

The SIFT detector is founded on a filtering technique known as *Difference-of-Gaussians (DoG)* operator, which approximates *Laplacian-of-Gaussian (LoG)* [40], traditionally employed for blob detection.
Given an input image $f(x, y)$, the DoG operator is calculated as the convolution of the difference between two Gaussian kernels at varying scales σ.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * f(x, y)$$

where G is the Gaussian function defined as

$$G(x, y, \sigma) = \frac{1}{2\pi t} e^{-\frac{x^2+y^2}{2\sigma}}$$

The scale space is created by convolving the image with Gaussian blurs at various factors. The resulting convolved images are organized into octaves, where each octave corresponds to a doubling of the σ value.
After forming the image pyramid, as illustrated in figure 2.11, key points are identified by searching for local minima or maxima in the adjacent Difference of Gaussian (DoG) images. Each pixel is compared to its local neighbours, which include 8 pixels at the same scale, 9 pixels at the upper scale, and 9 pixels at the lower scale. Once potential key point locations have been identified, these locations are refined for greater accuracy. The Taylor series expansion is used to determine a more precise position, and a threshold is applied to eliminate extrema with insufficient intensity. To ensure rotation invariance, each feature is assigned one or more orientations based on the gradient magnitudes of pixels in the local region. This technique is known as the *Histogram of Gradient Orientations*.

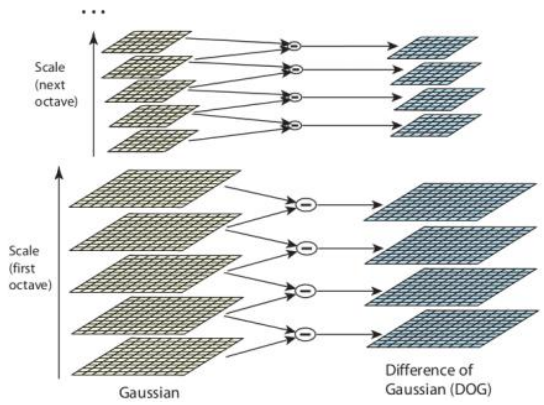

Figure 2.11: Difference of Gaussian

The next step in the SIFT algorithm is feature description. This method extracts information from a 16×16 neighbourhood around each detected feature. The region is further segmented into 16 4×4 sub-blocks. For each sub-block, an 8-bin orientation histogram is created, obtaining a total of 128 values representing the key point descriptor.
SIFT is invariant to image rotations, scaling and limited affine variations but requires a high computational cost.

ORB

The **ORB (Oriented FAST and Rotated BRIEF)** [41] algorithm was introduced in 2011. It derives from the *FAST* features detection algorithm [42] and the *BRIEF* description method [43].

Due to its high speed, FAST was a prominent algorithm in real-time systems. However, unlike SIFT, it does not produce an orientation description of the key point and multi-scale features. So, ORB employed a scale pyramid of the image and produced FAST features at each level in the pyramid, filtering them by the Harris corner measure [44]. Then, through the intensity centroid measure, the algorithm assigns an orientation to the corner extracted.

BRIEF is a binary descriptor method representing each key point with a string of dimensions from 128 to 512 bits. However, it is not invariant to rotation, so ORB utilises a rotation-aware BRIEF based on the orientation of the features computed in the previous step. As SIFT, ORB features are invariant to scaling operations, rotations and limited affine transformations.

AKAZE

Accelerated-KAZE (AKAZE) [45] was presented in 2013 as an improved version of the *KAZE* algorithm [46]. Similar to KAZE, it employs a non-linear diffusion filter, but its non-linear scale-spaces are constructed using a computationally efficient framework called Fast Explicit Diffusion (FED).

The Gaussian scale space used in SIFT or SURF has significant drawbacks, as it does not preserve object boundaries and tends to smooth out important details and noise equally. To address this, employing a non-linear scale-space allows for locally adaptive blurring to the image, preserving object boundaries. Nonlinear diffusion filtering is a technique utilised in image processing that alters the luminance of an image as it is analysed at different scale levels. This process is governed by non-linear partial differential equations (PDEs), which describe how brightness evolves based on a specific flow function. The classic nonlinear diffusion equation is

$$\frac{\partial L}{\partial t} = \mathrm{div}(c(x, y, t) \cdot \nabla L)$$

where L is the image luminance and c is the specific *conductivity* function that allows the diffusion to be adaptive to the local image structure.

The disadvantage of this approach is its intense computational demand, as there are no analytical solutions to the nonlinear diffusion equation. AKAZE, by utilising FED schemes, accelerates this method and outperforms SURF, SIFT, and KAZE, demonstrating high performance in feature detection and description.

K-NN for Matching task

Traditionally, handcrafted algorithms concentrate only on the pipeline's initial two stages: detection and description. Matches between local features can be identified by calculating a distance between the descriptors, which are high-dimensional vectors, using a specific distance function d, such as the Euclidean distance.

$$d(p, q) = \sqrt{(p - q)^2}$$

or the Hamming distance in the case of binary descriptors. The hamming distance between two equal-length vectors of symbols is the number of positions at which the corresponding symbols are different.

A classical approach to determining correspondences when computing distance measures is to use the **k-nearest neighbours algorithm (K-NN)**. This algorithm identifies, for each element, the k closest elements in the set. In local feature

matching, the K-NN algorithm finds, for each keypoint descriptor in the first image, the K nearest descriptors in the second image. K is typically set to 2 to retrieve the two nearest matches.

This enables the application of filtering techniques such as *Lowe's ratio test* to eliminate potential false positives. Lowe's ratio is the ratio of the distances between a point and its two nearest neighbours in the second image. Considering a local feature p_x in image d_x and an image d_y, the point p_y in image d_y that is closest to p_x is selected as a candidate match. Subsequently, the distance ratio $\sigma(p_x, d_y)$ of the closest to the second-closest neighbours of p_x in d_y is computed as:

$$\sigma(p_x, d_y) = \frac{d(p_x, NN_1(p_x, d_y))}{d(p_x, NN_2(p_x, d_y))}$$

where NN_1 and NN_2 represent the most similar and the second most similar key points of p_x in image d_y. Finally, the closest point is considered a correct match if this ratio is below a given threshold c. In [39], $c = 0.8$ was reported as a good threshold to eliminate a relevant number of false matches while discarding a few correct matches.

2.5 FDM: Deep Learning models

In the era of Deep Learning, significant advancements have been made to tackle the challenges of local feature matching, improving both accuracy and efficiency. Figure 2.12 shows some of the most influential AI-based models introduced in recent years. This section provides an overview and classification of these approaches, following the categorization proposed in [36].
Feature matching methods can be divided into two main categories: **Detector-based** and **Detector-free** approaches.

Detector-based These methods identify and describe sparse key points distributed across the image before establishing correspondences between different images. CNNs have demonstrated their capability to extract more robust key points and highly discriminative descriptors, making them adapted for this task. Traditional handcrafted methods, such as SIFT, ORB, and AKAZE, can be considered in this category alongside other algorithms like, for example, the Harris corner detector [44], FAST [42], and SURF [47].
The Detector-based methods can be further classified into four classes [36]:

- *Detect-then-Describe*: In this paradigm, keypoints are first extracted, and descriptors are subsequently generated by analyzing image patches centered

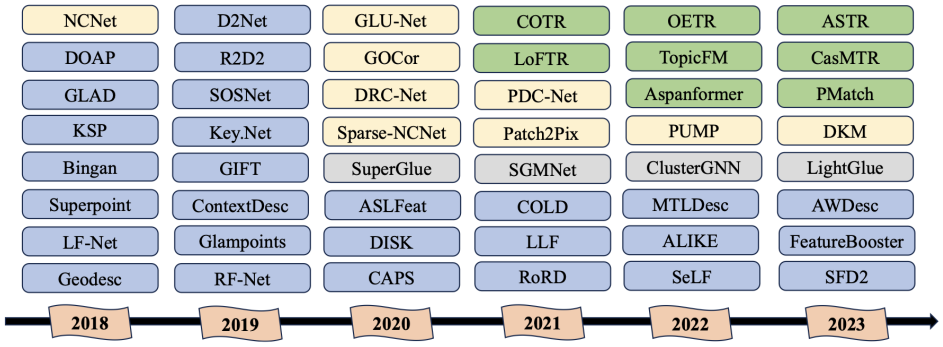

Figure 2.12: Representative local feature matching methods [36]

around each detected key point.

- *Joint Detection and Description*: This approach integrates the detection and description stages within a single model, optimizing both tasks simultaneously. An example is SuperPoint [48].
- *Describe-then-Detect*: In contrast to the previous approaches, this method first extracts local image descriptors, and then keypoints are detected based on the extracted feature representations.
- *Graph-based*: In the conventional feature matching process, correspondences are established through the nearest neighbor (NN) algorithm based on the descriptor similarity. Attention-based graph neural networks (GNNs) treat keypoints as nodes, utilizing self-attention and cross-attention layers from Transformers. This facilitates the exchange of visual and geometric information between nodes within the same image and across others. Examples include SuperGlue [49], GlueStick [50] and LightGlue [51]

Detector-free These approaches eliminate the need for sparse keypoints by exploiting the contextual information across the entire image. They generate dense matches by extracting visual descriptors from a dense grid covering the image. Compared to detector-based methods, they generally produce a higher number of repeatable feature points.
Detector-free strategies also employ different methodologies that can be grouped into:

- *CNN-based*: These methods extract dense feature maps from the image using CNNs. A 4D correlation tensor is then computed, which contains the similarity scores between all possible point pairs across two images. Matches are obtained by analyzing this four-dimensional correlation volume.

- *Transformer-based*: Models integrate the Transformer architecture to consider both local and global information into local feature matching. An example is LoFTR [52], where CNNs are used to extract initial feature maps, followed by the attention layers that refine feature descriptors and generate dense matches.

- *Patch-based*: Instead of detecting individual key points, the images are divided into patches and descriptors are computed for each one. The matching process occurs at the patch level before refining correspondences at the pixel level.

Figure 2.13 summarizes the distinctions between the AI-based models analyzed. This work focuses on SuperPoint, LightGlue, and LoFTR, which are described in more detail in the following sections.

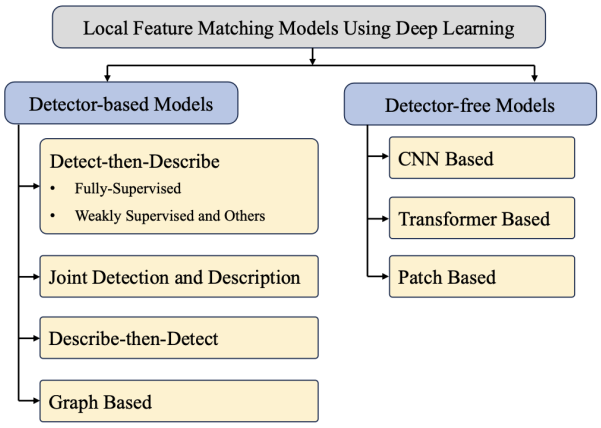

Figure 2.13: Categories of Local Feature Matching models [36]

2.5.1 FDM: Non-Transformer based models

SuperPoint

SuperPoint [48] is a fully convolutional neural network that jointly computes pixel-level interest point locations and descriptors.

The main insight proposed by this study is the generation and use of a specific self-supervised framework for training the neural network, as shown in figure 2.15. A dataset of pseudo-ground truth interest point locations is generated with the interest point detector itself providing supervision.

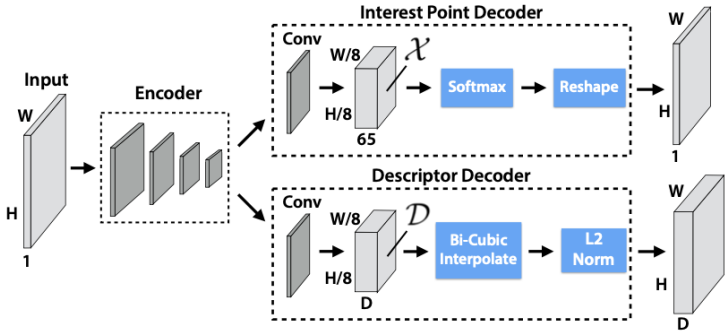

Figure 2.14: SuperPoint architecture

The SuperPoint architecture (fig. 2.14) has a single shared encoder to process and reduce the input image dimensionality. After the encoder, there are two decoder heads, one for interest point detection and the other for interest point description. The encoder consists of convolutional layers and max-pooling layers for spatial downsampling. After the encoder processes the image $I \in \mathcal{R}^{H \times W}$, the obtained feature map is fed into the two decoder blocks.
The detector head produces a tensor of the same dimension as the original image. Each pixel of the output corresponds to a probability of "point-ness" (probability of being a relevant feature point) for that pixel in the input. Instead, the descriptor head computed a tensor of size $H \times W \times D$, where D is the length of the vector descriptors. By combining the outputs of the two heads, the model can extract relevant key point positions with the corresponding descriptors.

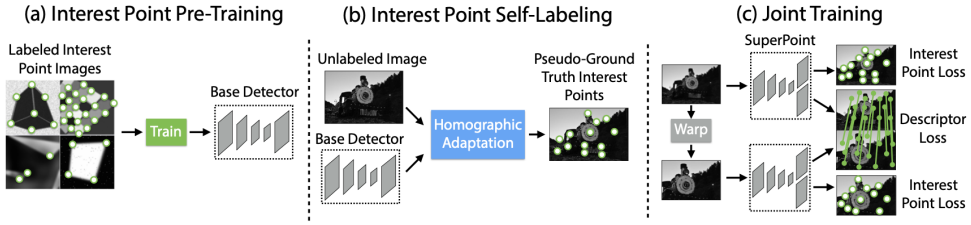

Figure 2.15: SuperPoint Self-Supervised Training overview

As mentioned earlier, they presented a self-supervised solution for training. The first step involves training a CNN detector on a synthetic dataset of simple geometric shapes (*Synthetic Shapes*). The resulting trained model is called *MagicPoint*. It has the same architecture as SuperPoint without the descriptor head. The MagicPoint detector generates pseudo-ground truth annotations on unlabeled real images using the *Homographic Adaptation* technique (figure 2.16).
This technique warps the input image multiple times, enabling an interest point

detector to view the scene from various viewpoints and scales, thereby achieving more consistent detections. The generated labels are ultimately employed to train the SuperPoint model. The performance of SuperPoint is assessed through Homography Estimation on the HPatches dataset. This model surpasses other detector and descriptor methods, such as LIFT, SIFT, and ORB.

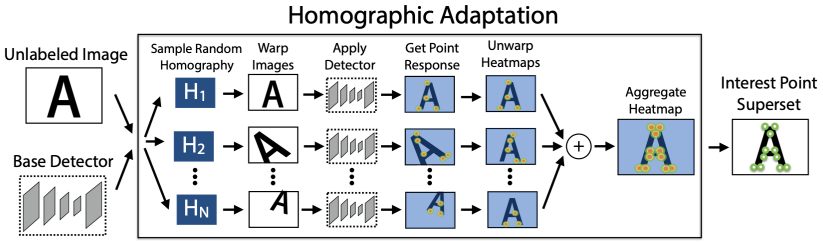

Figure 2.16: Homographic Adaptation

2.5.2 FDM: Transformer-based models

LightGlue

LightGlue [51] is an efficient and lightweight deep neural network for feature matching, proposed in 2023. LightGlue redesigns and optimises *SuperGlue* [49], a transformer-based model that matches sparse key points in image pairs.
By leveraging the power of the Transformer model and the Attention mechanism [53], SuperGlue has surpassed earlier local feature-matching models, demonstrating robustness across various indoor and outdoor scenarios and conditions. LightGlue enhances this by being more accurate, faster and efficient than its predecessor. Both models are dedicated only to the matching phase. They accept two sets of local features and descriptors, extracted from images A and B, as input. They operate alongside other detectors such as SuperPoint, SIFT, and DISK [54].
The architecture consists of a stack of L identical layers. Each layer comprises a self-attention and a cross-attention unit that updates the feature representations. These blocks rely on a message-passing mechanism using attention to aggregate information. Given input states x_i^I, the messages $m_i^{I \leftarrow S}$ are computed, where $I, S \in \{A, B\}$, and the states are updated as follows:

$$\mathbf{x}_i^I \leftarrow \mathbf{x}_i^I + \mathrm{MLP}\left(\left[\mathbf{x}_i^I \mid \mathbf{m}_i^{I \leftarrow S}\right]\right)$$

The message $m_i^{I \leftarrow S}$ is computed via attention-weighted aggregation of states from image S:

$$\mathbf{m}_i^{I \leftarrow S} = \sum_{j \in \mathcal{S}} \mathrm{Softmax}_{k \in \mathcal{S}}\left(a_{ik}^{IS}\right)_j \mathbf{W}\mathbf{x}_j^S$$

where a_{ij}^{IS} represents the attention score between points i and j in images I and S.

- In the Self-attention, each key point attends to all points within the same image ($m^{A \leftarrow A}$ and $m^{B \leftarrow B}$).
- In the Cross-attention, instead, each point interacts with all points of the other image ($m^{A \leftarrow B}$ and $m^{B \leftarrow A}$).

At the final layer, the updated states are used to predict correspondences through a soft partial assignment matrix P. This matrix integrates:

- A Pairwise score matrix $S \in \mathcal{R}^{M \times N}$, which encodes key points' similarity
- two "matchability score" vectors σ^A, σ^B, one for each image, which estimates the likelihood of each point having a correspondence.

$$\mathbf{P}_{ij} = \sigma_i^A \sigma_j^B \underset{k \in \mathcal{A}}{\text{Softmax}} \left(\mathbf{S}_{kj}\right)_i \underset{k \in \mathcal{B}}{\text{Softmax}} \left(\mathbf{S}_{ik}\right)_j$$

A pair of points (i, j) yields a correspondence when both points are predicted to be matchable and when their similarity is higher than any other point in both images.

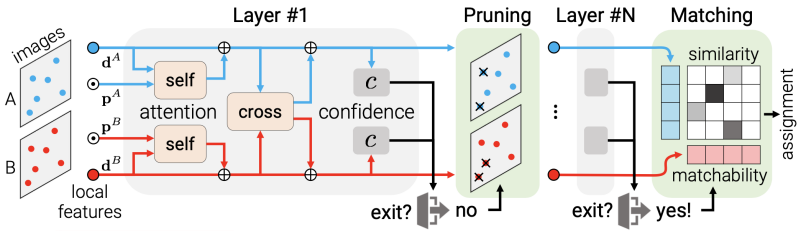

Figure 2.17: The LightGlue architecture

A major insight in the design of LightGlue involves adaptive inference, which minimizes computation and inference time depending on the complexity of the image pair. The model achieves this by dynamically reducing the number of layers for simpler tasks, like pairs of images with significant visual overlap and few alterations. After each layer, a classifier evaluates if the current predictions are confident and likely to stay consistent in later layers. If they are, the model stops the inference.
Furthermore, point pruning is utilized during processing. Points identified as both confident and unmatchable are eliminated, ensuring only pertinent key points proceed to the next layers.
LightGlue experienced training in a supervised framework using a two-stage method. Initially, it was pre-trained with synthetic homographies. In this phase, an image

pair is created by applying a homography transformation to an image, resulting in a warped version. This transformation facilitates the generation of ground truth matches, which are defined as key point pairs exhibiting a low projection error across both images. Following pre-training, the model was fine-tuned using the MegaDepth dataset [55], which offers a variety of real-world image pairs along with depth and pose information.
LightGlue surpasses SuperGlue in homography and relative pose estimations, showing greater accuracy and efficiency in feature-matching tasks.

LoFTR

LoFTR [52] is a detector-free feature-matching model designed to establish dense pixel-wise correspondences directly between images. Unlike other feature-matching approaches —such as SIFT, ORB, and SuperPoint— which rely on feature detectors to extract key points before matching, LoFTR eliminates the detection step, addressing scenarios where sparse key points may be insufficient.

Traditionally, dense correspondences are computed using CNNs. However, CNNs have a limited receptive field, due to the reduced kernel size. They solved this problem by using a Transformer in the architecture. This lead to have a global context of the input, compared to the local context of CNNs.
As shown in figure 2.18, LoFTR has four main components:

1. *Local Feature CNN*: A convolutional backbone that processes the input images I^A, I^B and extracts the coarse-level feature maps $\tilde{F}^A, \tilde{F}^B$ (1/8 of image dimension) and the fine-level feature maps $\hat{F}^A, \hat{F}^B$(1/2 of image dimension)

2. The coarse feature maps $\tilde{F}^A, \tilde{F}^B$ are then fed into the **Local Feature TRansformer (LoFTR)**, which is composed of N_c modules that contains a self-attention and a cross-attention layers

3. The matching module matches the transformed features, after the LoFTR module, and gives the coarse-level match prediction $\mathcal{M}_c$

4. For every selected coarse prediction $(\tilde{i}, \tilde{j}) \in \mathcal{M}_c$, a local window is cropped from the fine-level feature maps $\hat{F}^A, \hat{F}^B$ and the match is refined within this local window with another LoFTR module.

The core of the architecture is the LoFTR module. As for LightGlue, the module interleaves the self and cross attention. For self-attention, the input features f_i and f_j are the same (either $\tilde{F}^A, \tilde{F}^B$), while the cross-attention comes from both images($\tilde{F}^A$ and $\tilde{F}^B$ or $\tilde{F}^B$ and $\tilde{F}^A$), depending on the direction of the cross-attention.

In addition, they reduce the computational cost derived from the Attention layer in the Transformer module, by exploiting the Linear Attention Layer [56]. In fact, the vanilla dot-product attention in the classical encoder layer of the transformer has a complexity of $O(N^2)$, so it grows quadratically with the input length sequence N, while this efficient variant has a liner complexity $O(N)$.
LoFTR is trained on the ScanNet dataset for the indoor model and the MegaDepth dataset for the outdoor model. After training, the model is evaluated on the HPatches dataset for the homography estimation task and on the ScanNet and MegaDepth datasets for the relative pose estimation. The benchmark datasets mentioned are presented in the next section.

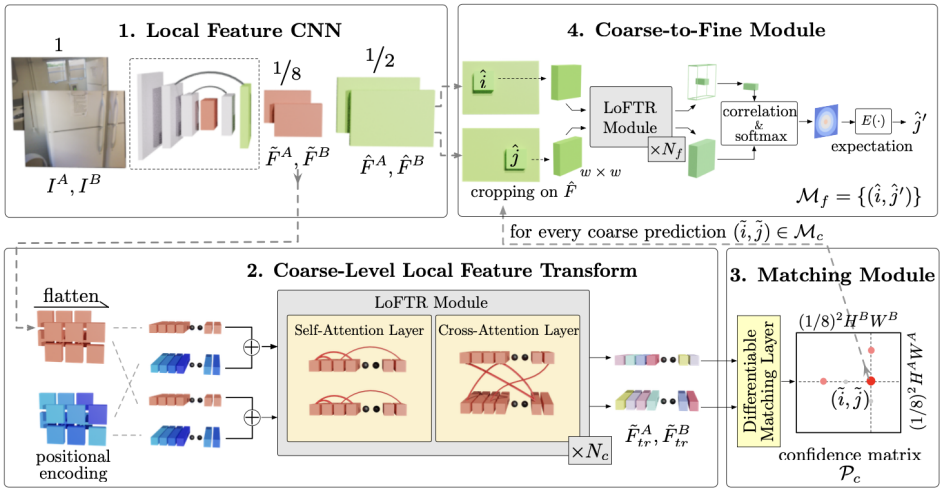

Figure 2.18: Overview of the LoFTR architecture [52]

2.6 Comparison between classical algorithms and Deep Learning models

Handcrafted algorithms and deep learning models vary greatly in how they tackle the FDM task. Deep learning models establish a novel paradigm, transforming the methods for feature extraction and matching, providing distinct advantages. It is possible to identify the following distinctions between the two classes of algorithms.

Feature extraction

Traditional algorithms identify key points using fixed mathematical operations like corner detection and intensity gradients. They focus on small and isolated regions

of the image, making them less effective in scenarios with low-texture surfaces, extreme lighting conditions and uniform backgrounds. This reduces their capability to extract distinctive and repeatable key points.
In contrast, DL models learn to detect relevant features directly from data without relying on predefined rules. They can capture both local and global context, and they better generalize across different imaging conditions. They are also more robust to variations in illumination, texture, and geometric transformations.

Matching accuracy

The Handcrafted methods rely on manually designed feature descriptors, which are fixed and do not adapt to different conditions. Since these descriptors encode information only from the local region where the feature is detected, they struggle to capture log-range dependencies between different features in the image. and encode information only from the local region where the features are extracted. Matching is performed based on descriptor similarity, which can lead to incorrect correspondences and false positive matches, especially in complex or low-texture environments.
On the other hand, DL models can simultaneously detect key points and optimize their corresponding descriptors based on the entire image context. In the matching phase, they can model contextual relationships between all extracted key points, improving the accuracy of feature correspondences. Transformer-based models further enhance matching by learning global dependencies, making them particularly effective in scenarios where traditional methods struggle.

Computational efficiency

Classical algorithms are typically faster in small-scale applications due to their lower computational demands. They do not require a training phase and their computational cost remains fixed.
AI models, by contrast, require more computational resources, particularly during the training phase. However, their architecture and computations can be optimized for efficient inference, leveraging techniques to improve speed and reduce memory consumption and computation.

2.7 Metrics

Various metrics are used to evaluate the performance of FDM methods in image matching. The choice of these metrics depends on the available information and the specific context of the application. When ground-truth correspondences are available or can be derived from the known geometric relationship between two

images, direct metrics can be used to measure the accuracy of individual predicted correspondences. These metrics precisely evaluate how well the detected keypoints match the true positions. However, accurate ground-truth correspondences are not always available. In such cases, indirect metrics can be employed. These metrics assess the algorithm's performance by evaluating higher-level tasks that rely on the extracted and matched features to estimate the geometric transformation between images. The **Homography estimation** and the **Relative Camera Pose** estimation are examined.

2.7.1 Homography Estimation

Images captured from various viewpoints or equipment can exhibit significant geometric and photometric differences due to factors such as environmental variations, camera technology differences, and shooting conditions variations. The homography estimation is a particularly relevant technique to describe the geometric projection relationship between images [57]. This technique has been proposed for various applications such as image registration and fusion.
A homography is a projective transformation between two planes or planar projections of an image [58]. It can encode geometric changes such as rotation, translation, scaling and projection. Figure 2.19 is a schematic representation of the Homography transformation model.

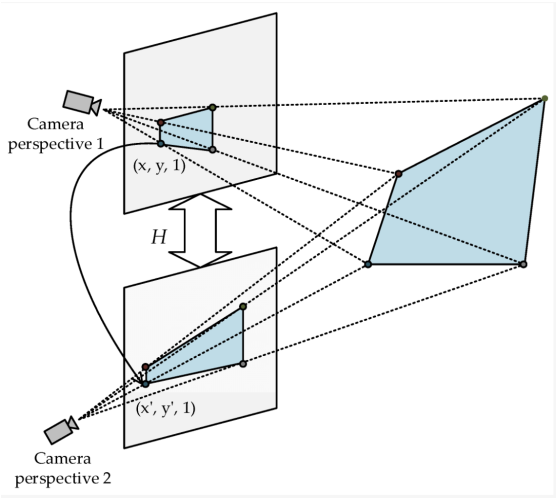

Figure 2.19: Representation of homography transformation [57]

This transformation is represented by the homography matrix H, which is a

3×3 real matrix defined as follows:

$$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{2.9}$$

where $h_{11}, h_{12}, h_{21}, h_{22}$ represent the affine trasformation, h_{13}, h_{23} the translation trasformation and h_{31}, h_{32} the perspective trasformation. The homography matrix relates the coordinates between corresponding points as:

$$\begin{pmatrix} x\prime \\ y\prime \\ 1 \end{pmatrix} = H \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2.10}$$

where $(x, y,1)$ are the homogenous coordinates of an interest point in the first image, and $(x\prime, y\prime, 1)$ denotes the coordinates of the corresponding point in the second image. Expanding the product, we get that

$$\begin{aligned} x' &= h_{11}x + h_{12}y + h_{13} \\ y' &= h_{21}x + h_{22}y + h_{23} \\ 1 &= h_{31}x + h_{32}y + h_{33} \end{aligned} \tag{2.11}$$

To solve the homography matrix between two images, we require two sets of matched coordinate points that can be identified through FDM methods. The element h_{33} is normalized to 1, so the matrix has 8 degrees of freedom and requires at least four pairs of points to compute the estimation. In a real case, more than four pairs are used.

The primary objective in homography estimation is to determine the transformation matrix H that minimizes the total re-projection error across all matched feature point pairs. This leads to an optimization problem, where the goal is to align the re-projected points from the source image with their corresponding observed points in the destination image.
The re-projection error for each matched pair is defined as the squared Euclidean distance between the actual point in the destination image and the point obtained by projecting the corresponding source image point using the estimated homography matrix H. The overall error can be formulated as:

$$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2$$

To obtain a robust initial estimation, methods such as RANSAC or Least-Median-of-Squares (LMedS) can be employed. These techniques operate by randomly sampling subsets of matched point pairs to compute multiple homography

candidates and selecting the one that yields the best consensus, so the highest number of inliers or the lowest residual error under a defined threshold.
Once a reliable initial estimate matrix is found, a non-linear refinement step is applied using the Levenberg-Marquardt algorithm [59]. This iterative optimization further reduces the re-projection error by minimizing the cost function defined above, typically resulting in a more accurate transformation.

For the evaluation of homography estimation accuracy, the angular correctness metric [36] is commonly used. This metric compares the ground truth homography $\hat{H}$ with the estimated homography $\tilde{H}$ by assessing how well they align specific reference points between the two images. Typically, the four corners of the first image are selected and transformed using both $\hat{H}$ and $\tilde{H}$. The projected points are then compared by computing the average Euclidean distance between the ground truth and estimated positions. If this average projection error is below a predefined threshold, commonly set between 1 and 10 pixels, the homography estimate is considered correct. To assess the performance across varying error thresholds, the Area Under the Curve (AUC) metric is employed. The AUC quantifies the proportion of correct estimations by integrating the accuracy over a range of pixel error thresholds.

2.7.2 Epipolar Geometry and Relative Camera Pose Estimation

Epipolar geometry The **Epipolar geometry** is a geometric model to describe the geometry in a stereo vision of the same 3D scene and is at the core of many complex tasks like Visual Odometry, Sfm and SLAM.

The traditional setup involves two cameras observing the same 3D point P, which is projected in each image plane in the locations p and p', respectively. Figure 2.20 illustrates a representation of the epipolar geometry model. In the figure, we can observe the camera centers O_1, O_2 and the **epipolar plane**, which is the plane defined by the camera centers and the 3D point P. The line connecting the two camera centers is called **baseline**. The **epipolar lines** instead are the intersections of the epipolar plane and the two image planes. [58].
The epipolar geometry creates a constraint between the projection of the same 3D point in the two images. In fact, by knowing the camera locations O_1, O_2 and a point in one image p, by definition the corresponding point $p\prime$, which is P's projection on the second image, must be located on the epipolar line of the second image.

The information about the epipolar constraint can be encoded in the **Fundamental matrix F** [58]. It is a 3×3 matrix that relates the points p and p', in the

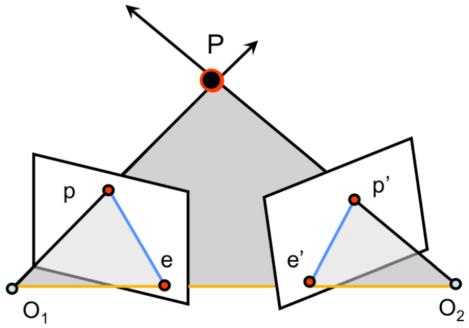

Figure 2.20: Schematic representation of the epipolar geometry model [60]

presence of unknown camera intrinsics. with the equation:

$$(p')^T F p = 0 \tag{2.12}$$

As said p and p' are in homogeneous pixel coordinates.
The matrix F is used to compute the epipolar line associated with the point. So, given the point p in one image, the corresponding epipolar line in the other image is $l' = Fp$ and similarly, $l = F^T p'$.

From the Fundamental matrix, it is possible to derive the **Essential matrix E** [58]. As the matrix F, also the Essential matrix represents the epipolar geometry constraints, but under the assumption that the cameras are calibrated, so with known camera parameters and undistorted images. The camera intrinsics parameters are the *focal lengths* f_x, f_y and the *principal point* coordinates c_x, c_y. They are typically represented with a matrix K.
The matrix E gives us the relationship between normalized camera matrices, so the effect of the known calibration matrix has been removed for each camera. The normalized image coordinates are obtained by transforming the original homogenous pixel coordinates using the inverse of the camera calibration matrix K.

$$\mathbf{p} = K^{-1} p \tag{2.13}$$

where $\mathbf{p}$ is in normalized image coordinates and p in homogenous image coordinates. E gives us the relationship:

$$(\mathbf{p}')^T E \mathbf{p} = 0 \tag{2.14}$$

where $\mathbf{p}, \mathbf{p}'$ are corresponding points in normalized image coordinates. Comparing this with the equation 2.12, it follows the relationship between the Fundamental and Essential matrices

$$E = (K')^T F K \tag{2.15}$$

where K', K are the intrinsic calibration matrices of the two images.

Relative camera pose estimation Now, we consider a pair of normalized camera matrices $C = [\mathbf{I}|\mathbf{0}]$ and $C' = [\mathbf{R}, \mathbf{t}]$. The Essential matrix, previously introduced, contains the information of the relative pose (i.e. position and orientation) between the two camera viewpoints and it has the form

$$E = [t]_x R \tag{2.16}$$

where $\mathbf{R}$ is the rotation matrix and $\mathbf{t}$ is the translation vector. $[t]_x$ denote the matrix form for the cross-product with the vector t

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \tag{2.17}$$

The elements $\mathbf{R}, \mathbf{t}$ represent the relative pose to transform a point from the first camera coordinate system to the second camera coordinate system.

The **Relative camera pose estimation** task involves retrieving the quantity $\mathbf{R}, \mathbf{t}$. The traditional approach is based on detecting and matching two sets of features in the images to establish point-to-point correspondences. Then the sets of corresponding points are used to solve the equation 2.14 and estimate the Essential matrix E, which has 5 degrees of freedom. Both the rotation matrix and the translation vector have three degrees of freedom, but there is an overall scale ambiguity, so the translation vector t can only be estimated up-to-scale without additional information.
Various algorithms can be exploited to solve the problem [61], [62]. Commonly, robust estimator algorithms like RANSAC [63] are embedded in the pipeline to reject outliers and obtain a robust estimation. Once E is estimated, it is decomposed to recover the relative rotation R and the relative translation t. Figure 2.21 is a schematic representation of the camera coordinate system and viewpoints of the scene.

In the relative camera pose estimation task, evaluation metrics serve to compare the ground truth information $\mathbf{R}, \mathbf{t}$ with the ones estimate d $\hat{\mathbf{R}}, \hat{\mathbf{t}}$ from the sets of matched points. Since the vector $\hat{\mathbf{t}}$ is a normalized vector, it indicates only the direction of the relative position without information about the magnitude. For this reason, the angular deviations [36] are used as error metrics. The following formulas are the angular deviations in translation θ_t and orientation θ_r

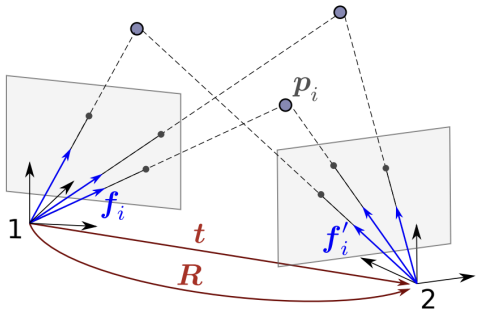

Figure 2.21: Camera coordinate systems [64]

$$\theta_t = arccos(\frac{t \cdot \hat{t}}{||t|| \cdot ||\hat{t}||}) \tag{2.18}$$

$$\theta_r = arccos(\frac{trace(R^T \cdot \hat{R}) - 1}{2}) \tag{2.19}$$

θ_t, θ_r are measured in degrees, and the overall pose error is usually denoted as the maximum among them.

Chapter 3

Space datasets and benchmark datasets

3.1 Space datasets

3.1.1 SPEED

SPEED (Satellite Pose Estimation Dataset) [1] is the first publicly available machine learning dataset for pose estimation of uncooperative satellites, released in 2019.
The dataset contains monocular images of the target spacecraft. By relying only on visual data, computer vision algorithms can estimate the target spacecraft's position and orientation (attitude) relative to the camera. Given the challenges of visual-based pose estimation in space, a standardized dataset is essential for evaluating and comparing the performance of different algorithms. SPEED was introduced to address this need.

The target used in the dataset is the TANGO spacecraft [65] from the PRISMA mission [66] [67]. The images are generated from two different sources:

- *Synthetic*: Created using a camera emulator software and preprocessed to emulate the illumination conditions captured from the actual flight imagery. For half of the images, the Earth background is inserted with non-fixed configurations. The synthetic images replicate the pixel intensity distributions and characteristics of the real images.

- *Realistic-facility (real)*: Captured using the TRON (Testbed for Rendezvous and Optical Navigation) facility at the Stanford Space Rendezvous Laboratory [68]. The facility is a simulated environment with a 1:1 mockup model of the

TANGO spacecraft and a 7 degrees of freedom robotic arm which holds the camera and takes pictures of the mockup model from different viewpoints.

The images have ground truth pose labels consisting of the translation vector and a unit quaternion describing the relative orientation of the Tango spacecraft with respect to the camera frame.
The dataset contains 15000 synthetic images and 300 real images divided into a training and test set. Samples from both real and synthetic sets are in the figures 3.2 and 3.3.
All images are grayscale with a resolution of 1920x1200 pixels. The ground truth pose labels refer to the camera reference frame and report the information relative to the spacecraft reference frame as shown in Fig. 3.1. The labels are encoded with a translation vector $t_{\mathcal{BC}}$ for the translation and a quaternion vector $q_{\mathcal{BC}}$ for the rotation.
The satellite's distance in the synthetic images is between 3 and 40.5 meters, while the distance distribution of real images ranges from 2.8 to 4.7 meters.

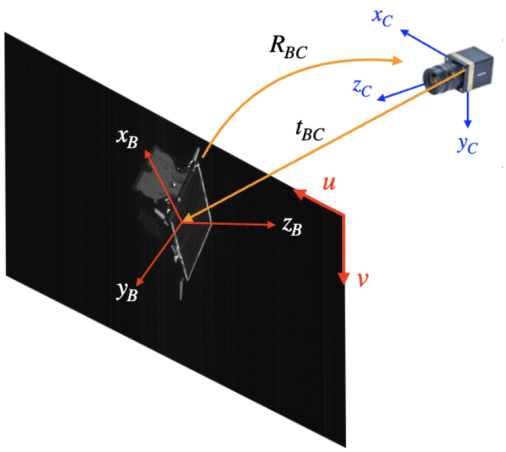

Figure 3.1: Definition of spacecraft body reference frame $\mathcal{B}$, camera reference frame ($\mathcal{C}$), relative position ($t_{\mathcal{BC}}$), and relative orientation ($R_{\mathcal{BC}}$). [1]

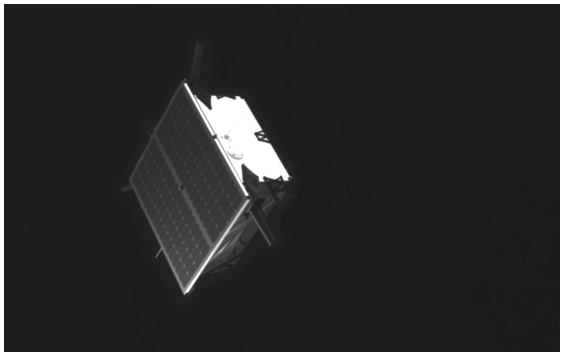

Figure 3.2: Image from the *Real* domain

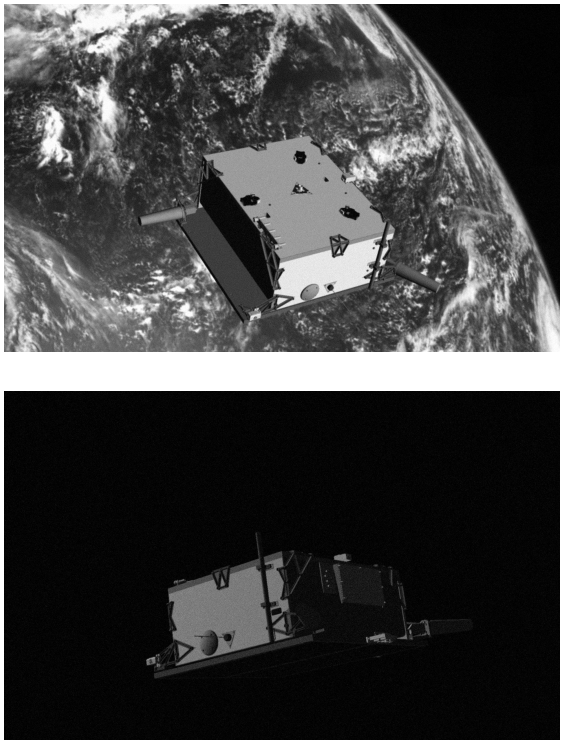

Figure 3.3: Examples of images from the *Synthetic* set of the SPEED dataset

3.1.2 SPEED+

The **SPEED+** dataset [2], released in 2021, is a natural evolution of the SPEED dataset and shares different characteristics with it. It is a dataset for spacecraft pose estimation. In particular, this dataset emphasizes the domain gap between synthetic and real images to improve the evaluation of the robustness of DL models.

SPEED+ includes images of the TANGO spacecraft, consisting of different domains collected from two sources. As in SPEED, the first is a camera emulator software to create 59960 synthetic images. The second source is the TRON facility at SLAB (the same used for SPEED), which is used to generate two domains with different and realistic illumination conditions

- *Lightbox*: They used light boxes with diffuser plates for albedo simulation
- *Sunlamp*: they reproduced direct high-intensity homogeneous light from the Sun.

All the images are in grayscale and have a resolution of 1920x1200 pixels. The ground truth label reports information on the spacecraft's relative pose to the camera. Examples of pictures present in the dataset

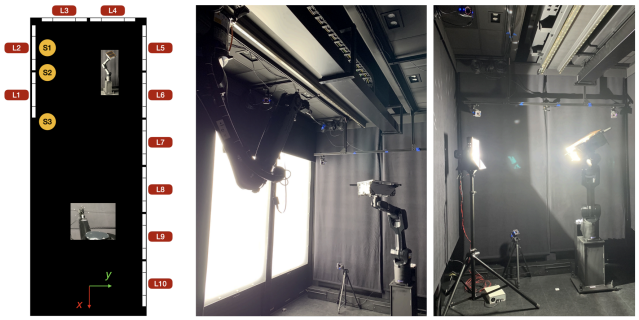

Figure 3.4: Schema and photos of the TRON facility simulation room layout [2]

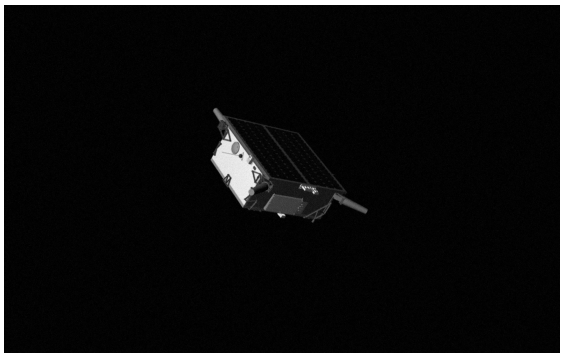

Figure 3.5: Synthetic images from SPEED+

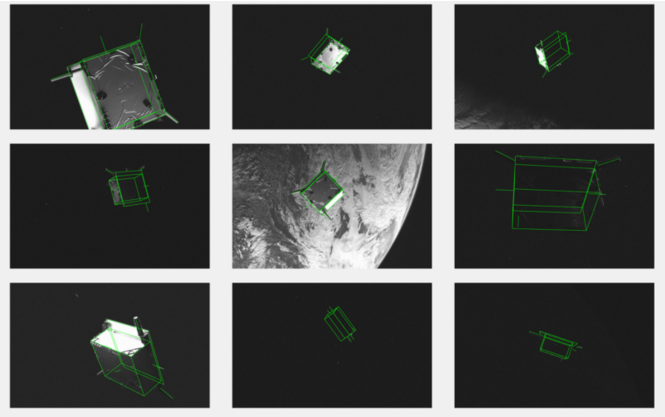

Figure 3.6: Example lightbox images with the wireframe model of the Tango spacecraft [2]

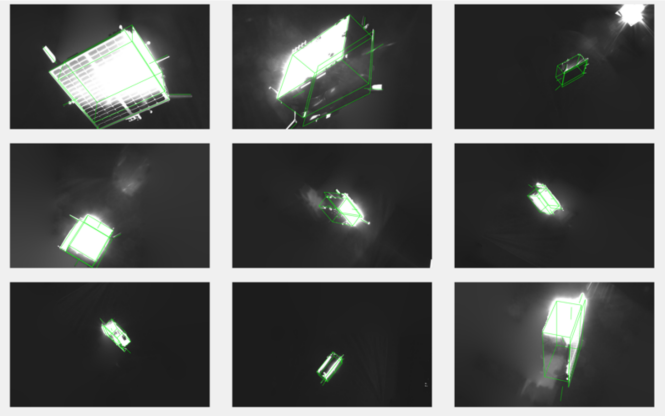

Figure 3.7: Example sunlamp images with the wireframe model of the Tango spacecraft [2]

3.1.3 SPEED-UE-Cube

The last space-related dataset considered is **SPEED-UE-Cube** [3]. It builds on previously released datasets, such as SPEED and SPEED+, with some important modifications. Like its predecessors, it is a machine-learning dataset for monocular pose estimation of a non-cooperative target, focusing on RPOD operations.
The dataset employs a 3U CubeSat model, which differs from the Tango spacecraft used in the other datasets. Figure 3.8 shows the CubeSat's 3D CAD model. In addition, all the images are synthetic and rendered using the software Unreal Engine 5 [69].

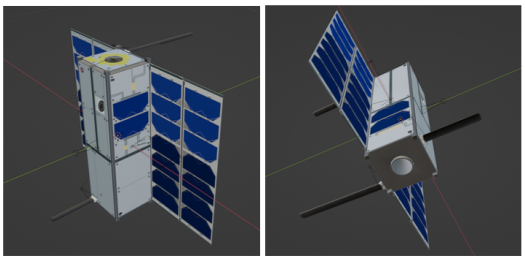

Figure 3.8: A 3D model of the target CubeSat

SPEED-UE-Cube consists of two distinct subsets:

- Training dataset: it is a set of 30,000 images of the target spacecraft with varying conditions of illumination, background and pose.
- Trajectory dataset: it is made of 1,186 sequential images that simulate a rendezvous scenario between the target and a servicer spacecraft.

The trajectory is particularly relevant because it is a sequential test set designed to evaluate pose estimation algorithms on realistic RPOD scenarios with sequential relative information instead of single and isolated cases. Figure 5.16 contains a sequence of images from the datasets.
The images are RGB images with 1920x1200 pixels resolution. The associated ground truth labels represent the relative position and rotation with respect to the camera. In addition, SPEED-UE-Cube limits the distance between the servicer and the target to 15 meters, while SPEED samples distances up to 50 meters. Also, the 3U CubeSat model used in SPEED-UE-Cube is smaller than the TANGO model adopted in SPEED and SPEED+.

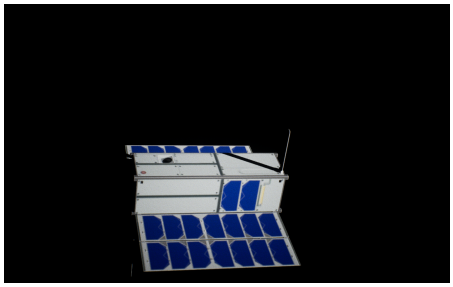

Figure 3.9: Sample image from the training set of the SPEED-UE-Cube

Table 3.1 gives an overview of the three datasets with the sets included in each of them and the number of available images.

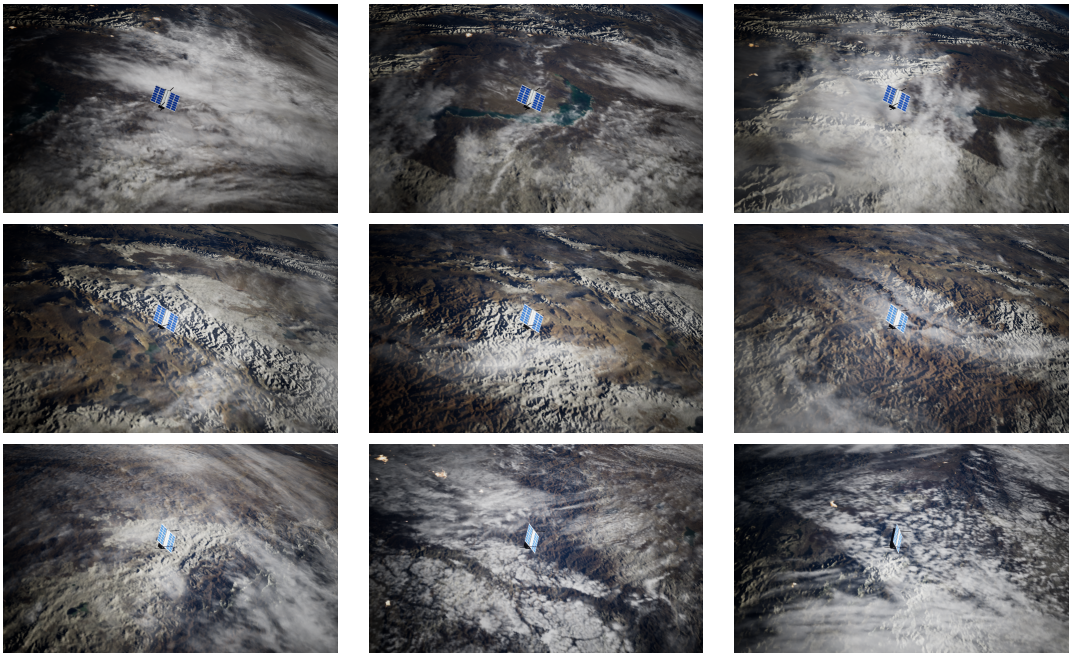

Figure 3.10: Sequence of images SPEED-UE-Cube trajectory set. The sequence is from left to right and from top to bottom

	SPEED		SPEED+			SPEED-UE-Cube	
Splits	**Synthetic**	**Real**	**Synthetic**	**Lightbox**	**Sunlamp**	**Synthetic**	**Trajectory**
Train	12000	5	47966	-	-	24000	-
Validation	-	-	11994	-	-	6000	-
Test	2998	300	-	6740	2791	-	1186

Table 3.1: Number of images in the SPEED, SPEED+, and SPEED-UE-Cube datasets.

3.2 Benchmark datasets

3.2.1 HPatches

The HPatches dataset [4] has been established as a benchmark for training and evaluating local feature extractor and descriptor algorithms. It is used in different tasks, such as image matching, patch retrieval and verification, and homography estimation. The last task is relevant to our scope. The dataset's images are collected from various sources, including other existing datasets. The images are organized in 116 sequences taken from different scenes. Each sequence comprises one reference image and five target images related to it by ground truth homography. The sequences are split into 2 types:

- Viewpoint: 59 sequences with significant viewpoint change
- Illumination: 57 sequences with significant illumination change, both natural and artificial

The camera captures target images by varying the viewpoint and illumination conditions. Some examples of transformations are rotations of the camera, varying the camera zoom and focus, camera positions from a front view to a different angle, light changes varying the camera aperture, or the daily time the image was taken. The sequences are captured so that homographies from the reference image to each target image can approximate the geometric transformations between images. This represents the ground truth label required for the homography estimation task.

Figure 3.11: Example of an image sequence in the HPatches dataset

3.2.2 ScanNet

ScanNet [5] is a large-scale RGB-D dataset for 3D reconstruction and semantic scene understanding. It contains around 2.5 million frames from 1,500 indoor environments, annotations and information about the images. In particular, it offers ground truth information about the camera pose with respect to the scene and camera intrinsic parameters. Thus, it is possible to evaluate FDM algorithms for relative pose estimation on this dataset.

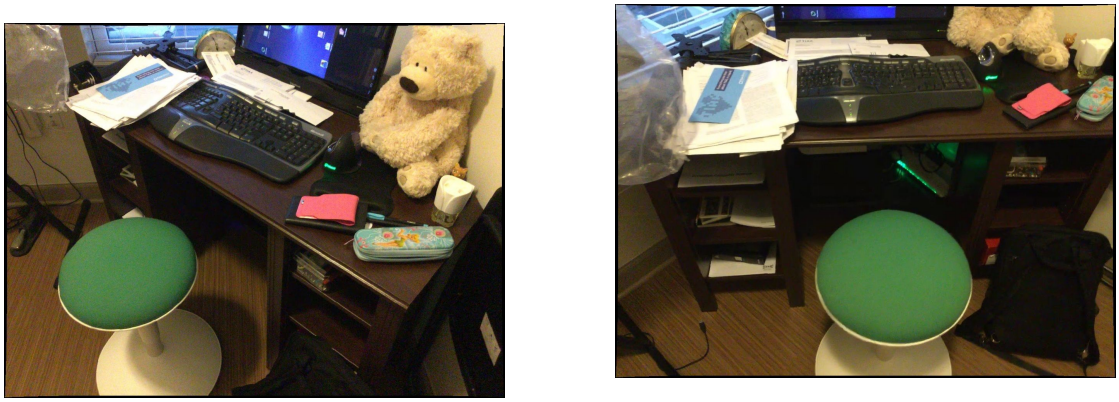

Figure 3.12: Example of images from the ScanNet dataset

Chapter 4

Methods

This chapter describes in detail the methodologies adopted for evaluating Feature Detection and Matching methods and for training the Student model using Knowledge Distillation. The objective of the work is to compare classical algorithms and AI-based methods on images from space-related use cases, highlighting the key differences and limitations of these approaches.

The first section focuses on the preprocessing steps applied to the previously introduced space datasets. Due to the lack of datasets specifically designed for FDM in space contexts, image-processing strategies are required to construct meaningful evaluation scenarios. This is closely related to the indirect evaluation metrics considered in this study: **Homography estimation** and **relative camera pose estimation**. Both metrics rely on knowledge of the geometric relationship between images, such as the homography transformation or the relative pose between camera viewpoints. Therefore, preprocessing is essential to generate image pairs for which this ground truth information is available and can be compared to the output obtained from the FDM process.

The second section explores the application of KD in FDM models. This technique allows a Student model to be trained under the guidance of a pre-trained Teacher model, which can have a different architecture. In this context, it also helps address the lack of appropriate datasets for direct training on FDM tasks by leveraging the supervision provided by the Teacher model. The student model, the distillation strategy, and the different training phases are explained.

4.1 Data preprocessing

The FDM process requires two images taken from different viewpoints of the same scene or object as input. However, the satellite datasets available are designed for assessing spacecraft pose estimation from a single-camera frame. Consequently, these datasets do not include predefined image pairs, making it necessary to generate them.
The first step is to identify strategies to construct sets of coherent image pairs that allow for a correct evaluation. Randomly pairing images without constraints is not an optimal strategy. For instance, some images may capture completely different perspectives of the spacecraft, significantly reducing the shared visual content. Without sufficient overlap or excessive transformations, it becomes hard to identify and match consistent and repeatable interest points, thereby compromising the feature-matching process and the consequent estimation of the geometric transformation.

In a real-world scenario, this aspect is combined with the frame rate of image acquisition and an analysis of the motion between the camera and the target. Balancing these factors allows consecutive frames to capture overlapping areas of the target, which can improve the effective application of FDM methods.
To address this challenge, two distinct ways of generating meaningful image pairs are explored:

- **Homography-based**: It involves picking an image from the original dataset and applying a planar transformation to generate a second image. This approach is applied to perform the Homography estimation.

- **Relative camera pose-based**: This leverages the ground truth annotations available in the datasets. Each dataset provides information about the spacecraft's pose in the camera coordinate system. By comparing the poses of the spacecraft in two different camera coordinate systems, we can compute the relative camera pose transformation. This method is used in the relative camera pose estimation.

The following sections provide a detailed explanation of both strategies, with visual examples illustrating the different scenarios.

4.1.1 Homography generations

The homography estimation is particularly useful because it does not require 3D information about the scene and can be applied to any 2D image pair. Its estimation is relevant in scenarios where the camera experiences pure rotational motion around its optical centre or when objects are situated at considerable distances from the camera. In these situations, the homography offers a valid geometric transformation model.

The approach adopted for generating image pairs via homography involves sampling a random homography transformation and applying it to a reference image, creating a warped version. As a result, the two images are related by a known homography matrix $\hat{H}$. To ensure effective and meaningful transformations, homographies are generated through a combination of fundamental planar transformations such as translation, scaling, and rotation. The procedure follows the steps:

1. The four corners of a square patch centred in the original image are identified, defining the region of interest for the transformation.
2. A concatenation of elementary transformations - translation, rotation and scaling - is applied to the corners.
3. The original and transformed coordinates of the four points are used to identify the homography matrix $\hat{H}$.
4. The computed homography is applied to the original image, generating a pair consisting of the reference image and its transformed version.

Homographies are randomly sampled within predefined hyperparameter constraints to regulate the transformation's complexity. These parameters control aspects such as the patch's size, maximum rotation angle, range of scaling factors, allowed translation values, and the number of discrete angles sampled. Fine-tuning these parameters ensures that the generated transformations remain sufficiently challenging without introducing excessive distortion. The intervals chosen for the homography generations are reported in the table 4.1. Figure 4.2 illustrates some examples of pairs generated.
The homography estimation metric is highly dependent on the degree of transformation applied. In the presence of excessive perspective distortion, the estimation can be inaccurate due to the difficulties in the FDM process and the numerical estimation of the correct homography matrix. For this reason, careful selection of hyperparameters is important to maintaining a balance between expressiveness and image integrity.

Hyperparamete	range
Path size	600×960
Translation factor	(0, 0.3)
Rotation angle	$(0, 90)^\circ$
Scale factor	(0.7, 1.3)

Table 4.1: Hyperparameters for the homography generation

To evaluate the homography estimation, the FDM process is applied to the generated image pairs, extracting feature correspondences between the original and transformed images. The estimated homography matrix $\tilde{H}$ is then compared to the ground truth $\hat{H}$, providing an indirect metric to assess the accuracy of detected correspondences.

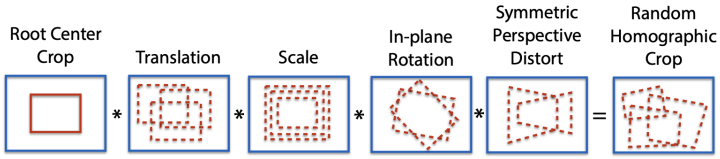

Figure 4.1: Random Homography generation [48]

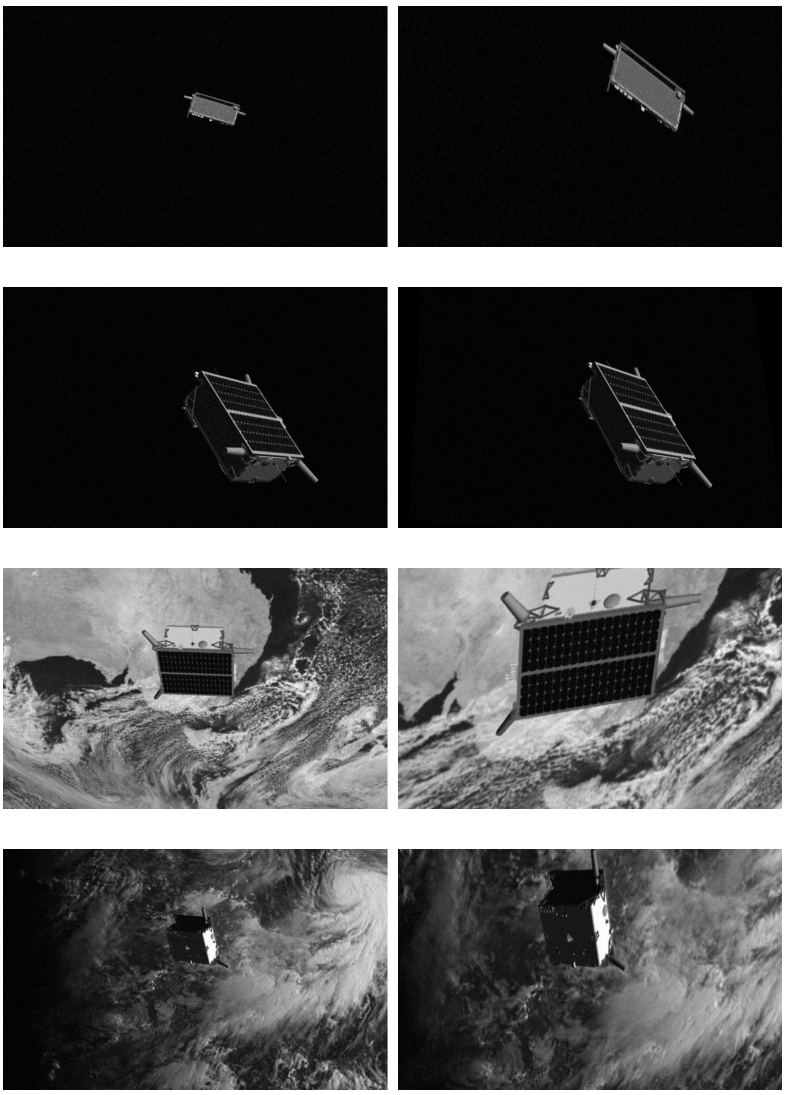

Figure 4.2: Images from SPEED dataset with homography transformation applied. *(left)* Original image. *(right)* Modified image

4.1.2 Image pair generation from relative camera pose

In this approach, the objective is to generate image pairs directly from the dataset rather than creating pairs through synthetic transformations. A key challenge in this process is ensuring that the selected image pairs exhibit sufficient visual overlap to extract relevant and matchable features. To address this, images are paired based on similarity criteria that ensure a partial view overlap of the same object, in this case, the spacecraft.
To achieve this, the ground truth pose information provided in the dataset is employed. The label consists of a quaternion vector for the orientation and a translation vector for the position, as detailed in section 3.1.1. The three datasets share the same notation for label representation.
For each image, the *relative camera pose* with respect to another image is computed, representing the transformation between the two camera viewpoints.
The matrix $T_{1\to2}$ encodes the change from the first camera coordinate system to the second coordinate system and is computed as:

$$T_{1\to2} = T_2 T_1^{-1} \tag{4.1}$$

where T_1, T_2 are 4×4 transformation matrices from the respective camera to the spacecraft coordinate system and are constructed as:

$$T_i = \begin{bmatrix} R_i & t_i \\ 0 & 1 \end{bmatrix} \tag{4.2}$$

Here, R_i is the 3×3 rotation matrix derived from the quaternion q_i in the ground truth annotations, and t_i is the 3-dimensional translation vector.
From the relative transformation matrix $T_{1\to2}$, the relative rotation R_{rel} and the relative translation t_{rel} are retrieved.

To quantify the difference between the two camera viewpoints, the relative orientation angle and the relative translation magnitude are computed.
The orientation angle is derived from the trace of the relative rotation matrix R_{rel} using the formula:

$$\cos\theta = \frac{\mathrm{tr}(R_{\mathrm{rel}}) - 1}{2} \tag{4.3}$$

And then converted to degrees

$$R_{\mathrm{dist}} = |\arccos(\cos\theta)| \cdot \frac{180}{\pi} \tag{4.4}$$

For the translation, the relative translation distance is normalised by the average camera-object distance in the two images:

$$t_{\text{dist}} = \frac{\|t_{\text{rel}}\|}{\frac{1}{2}(\|t_1\| + \|t_2\|)} \tag{4.5}$$

This normalisation accounts for variations in absolute scale, ensuring that the computed distance remains independent of the overall scene scale.

Once the relative rotation angle and relative translation distance have been computed, image pairs are selected based on values that fall within a specified interval. These ranges are arbitrarily defined and guided by an analysis of changes in visual perspective. To ensure a more comprehensive evaluation, the analysis should consider values and conditions that closely reflect real use cases.
For the tests, three relative translation distance and rotation angle intervals, each progressively increasing in range, are chosen. This approach allows for generating image pairs with varying difficulty levels, simulating scenarios with increasing perspective changes. Larger relative translations or rotations result in more challenging conditions, where the spacecraft appears with significantly different orientations or positions in the two frames. By controlling these intervals, the dataset effectively provides image pairs that range from easy perspectives to broader viewpoint changes. The intervals adopted are:

- *Easy*: $t_{dist} \in (0.5, 0.7)$ and $R_{dist} \in (0{,}30)^\circ$
- *Medium*: $t_{dist} \in (0.7, 0.9)$ and $R_{dist} \in (30{,}50)^\circ$
- *Hard*: $t_{dist} \in (0.9, 1.1)$ and $R_{dist} \in (50{,}70)^\circ$

In the translation, we avoid choosing values that are too close to 0. If there is no camera movement between the two frames, the relative camera pose estimation can be unstable and fall in a degenerate case, so we require that there is at least a little movement of the camera between the images.
Some image pairs generated are present in the figure 4.3. In the *easy* scenarios, the pose of the spacecraft relative to the cameras is very similar and the frame captures the same face of the object. In the *hard* case, the camera performs a wider movement and rotation from the first to the second frame.

This method for selecting the images does not depend on other conditions, such as lighting variations or the surrounding scene. The presence of the Earth background can be a relevant factor in the evaluation of the FDM process and metric estimation. In fact, different experiments were conducted to evaluate its impact.
Initially, image pairs are created using only images without backgrounds, ensuring that the visual features extracted primarily derive from the spacecraft. Next, pairs

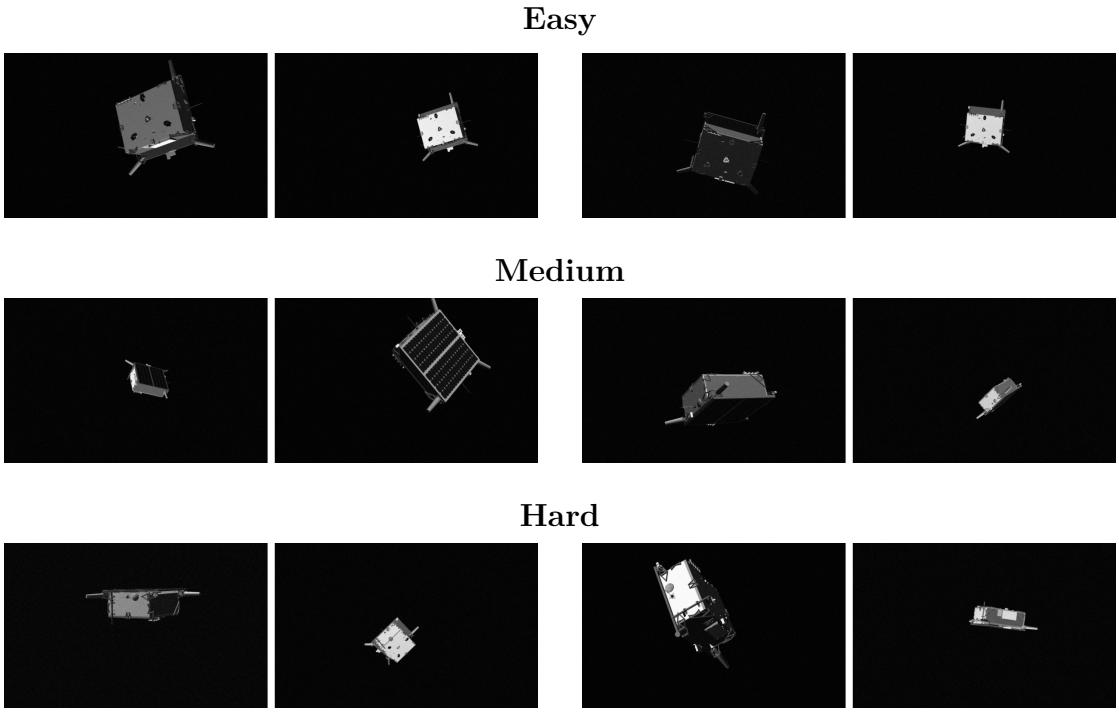

Figure 4.3: Image pairs on SPEED datasets paired based on the relative camera pose. In order: *easy*, *medium* and *hard* cases

are generated using images with an Earth background, allowing for a comparison of how the presence of a complex, variable background affects the matching process. This approach provides insights into the robustness of FDM methods when dealing with background clutter and variations in scene composition, which can be relevant considerations for real space applications. Figure 4.4 shows scenarios with background.

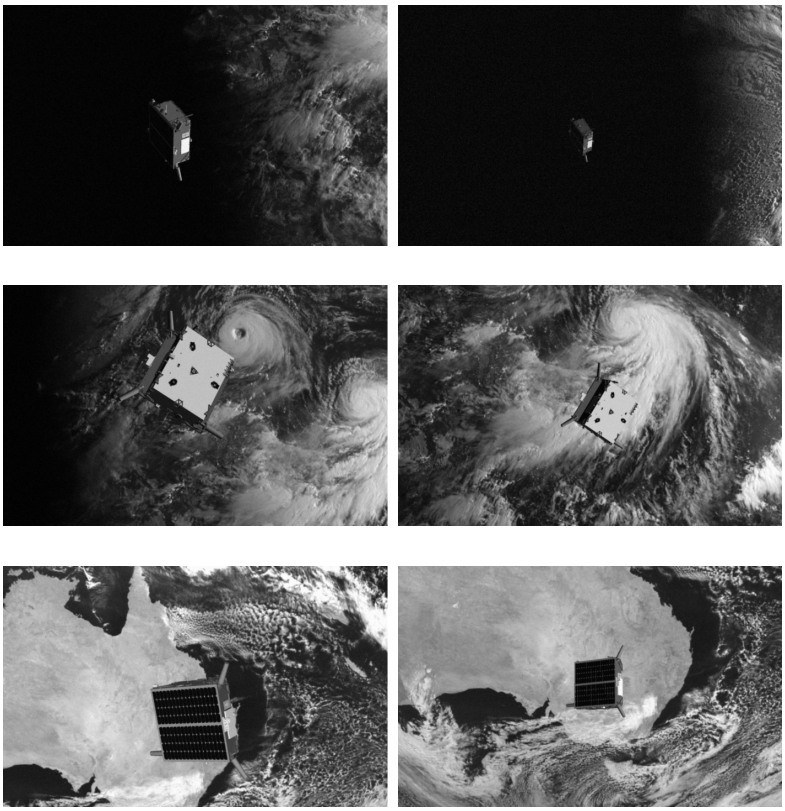

Figure 4.4: Images with background from SPEED dataset paired based on the relative camera pose (*easy* case)

4.1.3 Trajectory set

As introduced in Section 3.1.3, the SPEED-UE-Cube dataset provides a different context by replicating a more realistic RVD operation. The trajectory test set consists of sequential images that simulate the chaser's approach toward the target and its relative motion. When comparing consecutive frames, changes in the spacecraft's appearance are often minimal, with only slight rotations and camera movements. Additionally, the target spacecraft typically appears at a greater distance from the camera, occupying a smaller portion of the scene, unlike some of the previously analyzed cases. This makes it more challenging to extract relevant features from the target. Another aspect is the gradual variation of the background during the trajectory.
This dataset is particularly valuable for evaluating FDM under more realistic conditions, simulating image acquisition and processing at defined time intervals and a specific relative motion between the chaser and the spacecraft.
The dataset's sequential structure eliminates the need for pairing methods, as image pairs are naturally obtained by consecutive frames. The FDM process is thus applied directly between these sequential images.

4.2 Training model with KD

The second phase of the project focuses on Knowledge Distillation to transfer knowledge from a Transformer-based model to a Convolutional-based student model. The attempt seeks to replace the self-attention and cross-attention layers present in **LightGlue** with alternative layers.
The decision to use this model as a teacher is based on the experimental results obtained. LightGlue outperforms LoFTR in various conditions, particularly in the task of relative pose estimation, which is relevant for this study. Additionally, its ability to establish correspondences between sparse keypoints not only enhances matching precision but also allows the use of interest points extracted from different methods and models, such as SuperPoint and SIFT. Furthermore, It has a simpler architecture than LoFTR, which can facilitate the design and distillation process of the Student model. From a computational standpoint, LightGlue, even when combined with SuperPoint as a feature extractor, has a shorter inference time and smaller dimensions.

The student architecture is built on the teacher's design. We aim to maintain the same design but replace the Self-Attention and Cross-Attention modules. Our model is trained under LightGlue's supervision in order to replicate its performance in the matching process despite the architectural differences. The KD is exploited to compare and align the intermediate hidden descriptor representations produced

in each layer of the student with the corresponding representations generated by the teacher.
The training process consists of two steps that follow the same strategies already employed for the pairs generation. This procedure was used to replicate a similar procedure for the LightGlue's training, as described in [51]. The first stage is to train the student on synthetic image pairs generated via homography transformation, as previously detailed in the section 4.1.1. Next, the model is further trained on image paired based on the relative camera position and orientation. The pairs are generated through the criteria described in the section 4.1.2. The source of the images in both phases is the SPEED dataset with the *synthetic* domain.
LightGlue and the student only perform the feature matching phase, so they require an external FD method. In this case, the **SuperPoint** model works as a feature extractor. To simplify and speed up the training, SuperPoint is configured to output a maximum number of key points per image equal to 128. This constraint ensures that the model focuses on a manageable number of relevant features. In challenging situations, the quantity of extracted features is usually limited.
To evaluate the learning performance of the student model, several metrics are used to compare the correspondences predicted by the student with the ground truth correspondences. The adopted evaluation metrics are:

- *Match Recall (R)*: The proportion of ground truth matches correctly identified by the model. It measures how well the student model retrieves relevant correspondences.
- *Precision (P)*: The ratio of correct predictions among the total predicted correspondences. A higher precision indicates that the model is making fewer false-positive matches.
- *Accuracy (A)*: The proportion of all correct predictions, including cases where key points are not matchable. This metric provides an overall measure of the correctness of the model's predictions.
- *Average Precision (AP)*: It evaluates how well the predicted matches are ranked based on their confidence scores. A high AP score means that the model assigns higher confidence scores to correct matches.

In the following sections, we provide more detailed descriptions of the student model architecture, the KD loss and the training stages.

4.2.1 Student model architecture

The student model follows LightGlue's architectural structure, where each layer refines feature descriptors through an iterative update process.
For an input image pair, the SuperPoint model extracts from each image:

- A set of interest point locations $p_i := (x, y) \in [0,1]^2$.
- Corresponding feature descriptors $d_i \in \mathcal{R}^d$ with $d = 256$.

The extracted interest points, alongside their descriptors, serve as input for both the teacher and student models, represented as:

$$P_I \in \mathcal{R}^{N\times 2}, \qquad D_I \in \mathcal{R}^{N\times d}$$

where $I \in \{\mathcal{A}, \mathcal{B}\}$ and N, M are the number of detected points from the images $\mathcal{A}, \mathcal{B}$, respectively.
The student utilizes the interest point descriptors $D_\mathcal{A}, D_\mathcal{B}$ by feeding them into a stack of L modules. Each module features a custom version of the Self-Attention and Cross-Attention layers present in the original architecture. At the end of the modules, the student exploits the same assignment module present in LightGlue. This predicts the pair-wise assignment matrix P from which the estimated correspondences are retrieved. Figure 4.5 is an overview of the student architecture.
The next sections describe how the *Custom Self-Attention* and *Custom Cross-Attention* layers are designed.

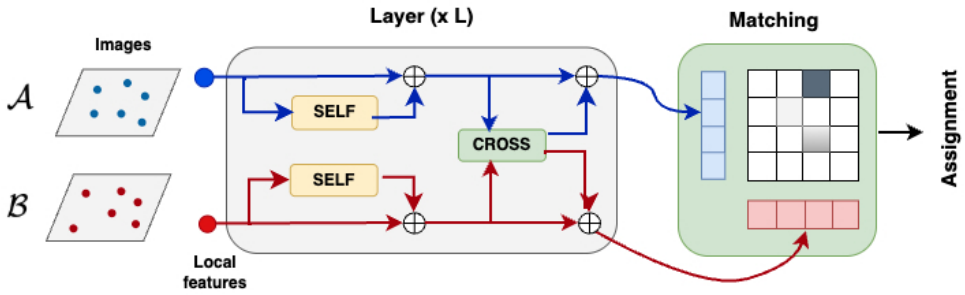

Figure 4.5: Overview of Student Architecture

Replacing self-attention

The Self-Attention mechanism considers as input the interest point's descriptors from a single image, either $D_\mathcal{A}$ or $D_\mathcal{B}$.
The attention operation is replaced with a sequence of 2D convolutional layers. For the design of the convolutional block, we take inspiration from [70], especially in some insights like the larger kernel size of the convolutional operation.
The convolutional block contains an initial PointWise Convolutional layer that expands the channel dimension from 1 to 32 channels. Then, three consecutive

convolutional modules composed of DepthWise and PointWise convolutional operations are performed. At the end, a PointWise layer reprojects the channel dimension to 1. The convolutional layers adopted are present in table 4.2

Operation	Kernel Size	Input Channels	Output Channels
Pointwise Convolution	1×1	1	32
Convolutional Blocks (3x)			
Depthwise Convolution	7×7	32	32
Layer Normalization	-	-	-
Pointwise Convolution	1×1	32	128
ReLU Activation	-	-	-
Pointwise Convolution	1×1	128	32
Pointwise Convolution	1×1	32	1

Table 4.2: Convolutional operations in the module to replace the Self-Attention layer

After the convolutional module, the output descriptors $D' \in \mathcal{R}^{N \times d}$ are combined with the module's original input D. The merged descriptors with a dimension $2d$ are integrated through a Multi-Layer Perceptron (MLP) and reprojected to a dimension d, as made in LightGlue. The final outputs $D_{\mathcal{A}}$ and $D_{\mathcal{B}}$ are then passed to the *Custom Cross-Attention* module

Replacing cross-attention

To replace the Cross-Attention layer, a Cross-Graph convolutional learning module is adopted with the same approach described in [71]. This module originates from Graph Convolutional Networks (GCNs) and is designed to learn node embeddings from two different graphs in graph matching problems. In the context of feature matching, an analogy is drawn between node embeddings in graph matching and feature descriptors from different images. The model learns the correlations between feature descriptors from two images, providing an alternative to the Cross-Attention. Let $X^{(k)}$ and $Y^{(K)}$ denote the feature descriptors of image $\mathcal{A}$ and $\mathcal{B}$, respectively (rewritten from $D_{\mathcal{A}}^{(k)}$ and $D_{\mathcal{B}}^{(k)}$ for clarity).
The co-affinity matrix $C_{xy}^{(k)}$, which encodes the correlations between them, is computed as:

$$C_{xy}^{(k)}(i,j) = \exp\left(\frac{\left(X^{(k)^T} W^{(k)} Y^{(k)}\right)_{ij}}{\delta}\right) \in \mathbb{R}^{m \times n} \tag{4.6}$$

where $W \in \mathcal{R}^{d \times d}$ (d is the vector descriptor dimension equal to 256) is a trainable weight matrix. δ is a scale factor. In the same way, the co-affinity matrix $C_{yx}^{(k)}$ is computed.

Once the co-affinity matrices are obtained, the feature representations are updated as:

$$\begin{aligned} X^{(k+1)} &= \left[C_{xy}^{(k)} Y^{(k)} \| X^{(k)}\right] \Theta_{xy}^{(k)} \\ Y^{(k+1)} &= \left[C_{yx}^{(k)} X^{(k)} \| Y^{(k)}\right] \Theta_{yx}^{(k)} \end{aligned} \tag{4.7}$$

where $||$ is the concatenation operation and parameters Θ_{xy} an Θ_{yx} are trainable weight matrices. $X^{(k+1)}$ and $Y^{(k+1)}$ represent the update feature descriptors from the respective images $\mathcal{A}$ and $\mathcal{B}$.

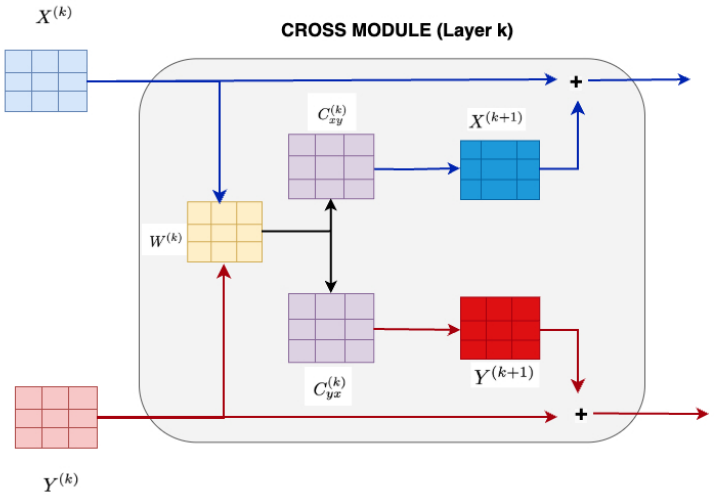

Figure 4.6: Schematic representation of Cross-module

4.2.2 Distillation

The Knowledge Distillation loss employed a feature-based approach and is crafted to evaluate and align the feature descriptors generated by each module of the student model with those in the teacher. More specifically, the loss function compares the feature descriptors following the self-attention module and the final output from each module, which occurs after the cross-attention layers. Supervision is applied at these intermediate levels because the student model substitutes both the Self-Attention and Cross-Attention layers.
The KD loss is expressed as a weighted sum of two components: the loss $\mathcal{L}_{SA}$ and the loss $\mathcal{L}_{CA}$

$$\mathcal{L}_{KD} = \alpha \mathcal{L}_{SA} + \beta \mathcal{L}_{CA} \tag{4.8}$$

where α, β are weight hyperparameter.The loss $\mathcal{L}_{SA}$ considers the descriptor of both images $(\mathcal{A}, \mathcal{B})$ from the teacher and student model after the Self-Attention layer, respectively $\hat{D}^{SA}$ and $\tilde{D}^{SA}$. It can be defined as:

$$\mathcal{L}_{SA} = \frac{1}{L}\sum_{i=1}^{L}(\mathcal{L}(\hat{D}^{SA}_{A,i}, \tilde{D}^{SA}_{A,i}) + \mathcal{L}(\hat{D}^{SA}_{B,i}, \tilde{D}^{SA}_{B,i})) \tag{4.9}$$

here, L is the number of layers in the architecture.
Instead, the loss $\mathcal{L}_{CA}$ is computed by comparing the descriptor after the Cross-Attention. Denoted as $\hat{D}^{CA}$ for the teacher and $\tilde{D}^{CA}$ for the student

$$\mathcal{L}_{CA} = \frac{1}{L}\sum_{i=1}^{L}(\mathcal{L}(\hat{D}^{CA}_{A,i}, \tilde{D}^{CA}_{A,i}) + \mathcal{L}(\hat{D}^{CA}_{B,i}, \tilde{D}^{CA}_{B,i})) \tag{4.10}$$

The loss function $\mathcal{L}$ employed is the Mean Square Error (MSE) function.
The KD loss is combined with the ground truth loss derived from the final correspondences predicted by the student model and the ground-truth correspondences. As for the original training of LightGlue, we aim to minimise the log-likelihood loss constructed as:

$$\begin{aligned}\mathcal{L}_{GT} =& \frac{1}{|\mathcal{M}|}\sum_{(i,j)\in\mathcal{M}} \log {}^{\ell}\mathbf{P}_{ij} \\ &+ \frac{1}{2|\overline{\mathcal{A}}|}\sum_{i\in\overline{\mathcal{A}}} \log\left(1 - {}^{\ell}\sigma_i^A\right) \\ &+ \frac{1}{2|\overline{\mathcal{B}}|}\sum_{j\in\overline{\mathcal{B}}} \log\left(1 - {}^{\ell}\sigma_j^B\right)\end{aligned} \tag{4.11}$$

P represents the assignment matrix predicted by the student following the L layers. $\mathcal{M}$ denotes the set of true matches, whereas $\overline{A}$ and$\overline{B}$ indicate sets of points classified as unmatchable. The loss is balanced between positive and negative labels.
The total loss comprises both the KD loss and the ground-truth loss. Figure 4.7 is a schematic representation of the distillation strategy.

4.2.3 Homography pre-training

In line with the methods applied in LightGlue and earlier in SuperGlue, synthetic homography transformations are used in a supervised pre-training phase.
This training stage allows the student model to learn how to recognise fundamental geometric transformations, including rotations, translations, and scaling. Since these transformations do not alter the intrinsic properties of the images, such as illumination conditions, the detection and matching of repeatable features across

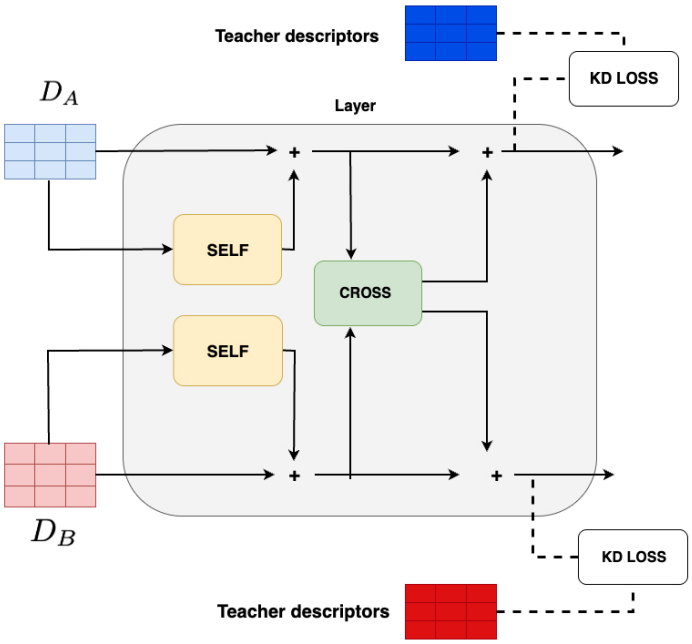

Figure 4.7: Schematic representation of the distillation strategy

image pairs may be simplified.
An important advantage of homography training is that it allows for the computation of ground-truth correspondences between interest points extracted. This is performed by reprojecting the detected features from one image to the other using the known homography $\hat{H}$ and assessing the reprojection distance for each potential pair of key points. Pairs with a reprojection distance smaller than a specified pixel threshold (3 in this case) create the set of ground truth matches $\mathcal{M}$, while other points are classified as unmatchable σ^A, σ^B. These correspondences serve as supervision for the assignment matrix P in equation 4.11.
In figure 4.8, there is an example of an image pair in the training set. The feature points are detected with SuperPoint with a maximum number of detected points equal to 128. The blue lines are ground-truth correspondences estimated with the reprojection distance.

4.2.4 Finetuning

After the homography-based pre-training phase, the student model is further trained on image pairs that are not generated through synthetic homography transformations. Instead, the training set is constructed by pairing images from the SPEED dataset using the strategy detailed in section 4.1.2. This phase exposes the model to more realistic scenarios where the relative motion between two images is not

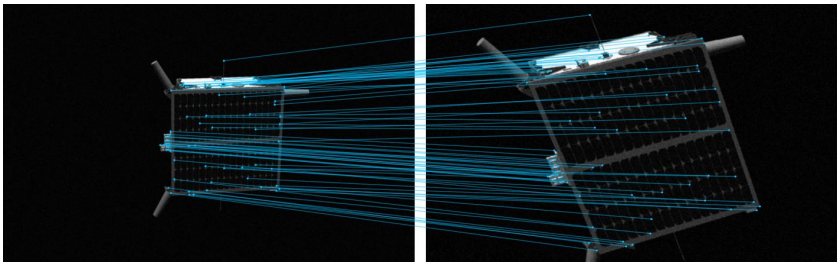

Figure 4.8: Reference image from the SPEED dataset (left) with its transformation (right). The blue lines indicate ground truth matches

represented by a homography transformation, making the task more challenging and closer to real conditions.
The common visual regions of the spacecraft between the two images may differ significantly due to viewpoint changes and illumination variations. This increases the complexity of feature detection and matching, requiring the model to generalize beyond the synthetic transformations used in the pre-training. The images selected do not contain a background since the objective is to correctly match points extracted from the object and not from the background. Figure 4.9 contains sample images used for the training.
Unlike the homography pre-training phase, where the known homography transformation allows for direct computation of ground-truth correspondences, depth information is unavailable in this scenario, making it not possible to explicitly compute ground-truth feature matches for the extracted key points from SuperPoint. To overcome this limitation, the student model is trained using a KD loss combined with a pseudo-ground truth loss, where the matches predicted by LightGlue are treated as correct matches.
Using these predictions as ground truth introduces potential sources of error, as the teacher model itself may produce incorrect matches. However, this setup enables evaluating the student model's ability to replicate the teacher model's matching behavior.

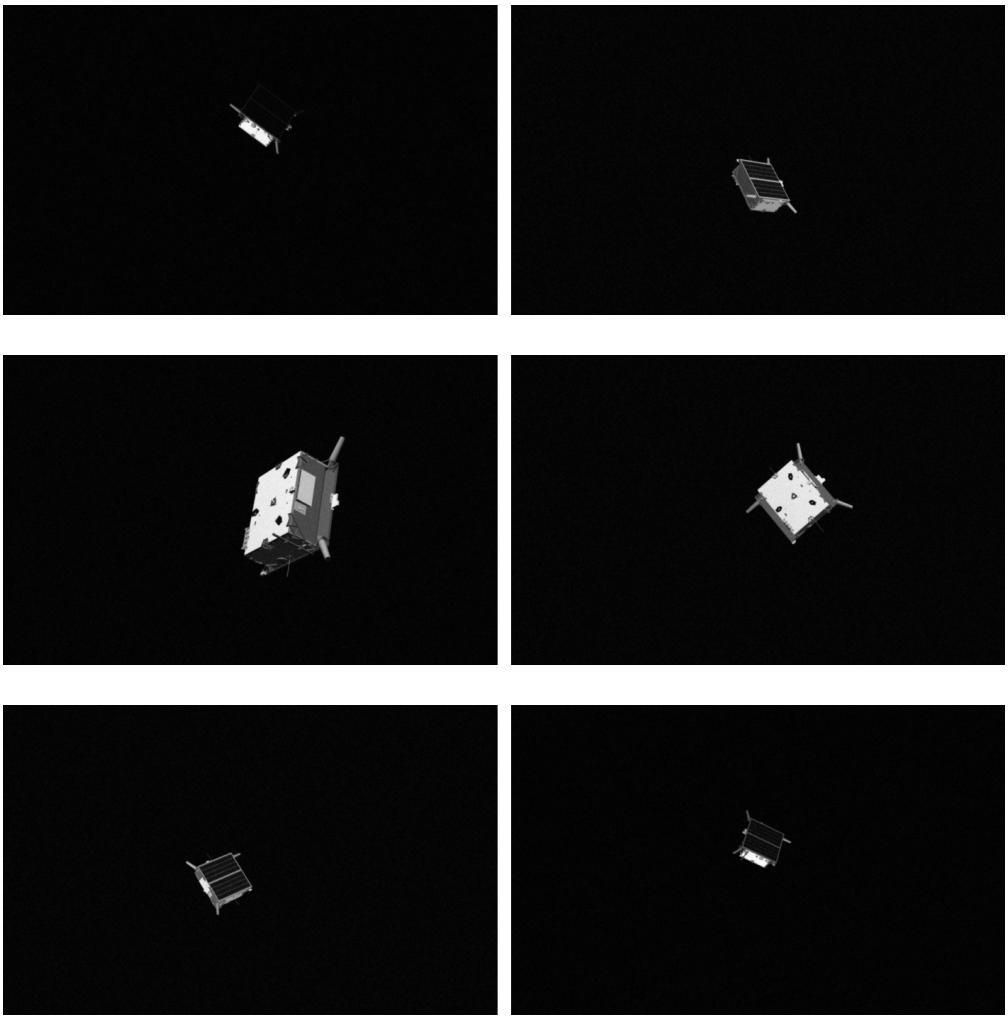

Figure 4.9: Images in the dataset for the student model training

Chapter 5

Experimental results

This chapter presents the experimental results and analysis conducted to evaluate the FDM methods' performance in space-related scenarios.

The first part compares traditional algorithms, SIFT, ORB, and AKAZE, with AI-based models such as SuperPoint combined with LightGlue and LoFTR. The evaluation follows the methodology introduced in the previous chapter, which covers the image pairing strategies and the metrics used. In this section, we explore different settings to give a broader analysis of the advantages that AI models offer over classical methods and their limitations in specific contexts. As discussed in Section 2.1.2, space imagery presents unique characteristics that differ significantly from those found in more generic Earth-based environments.

The evaluation starts with the Homography Estimation task. The tests include the benchmark dataset HPatches, followed by the space datasets SPEED and SPEED+. In particular, on the SPEED+ dataset, we compare results obtained on synthetic images generated via rendering software with real images captured in the laboratory using mock-up models of spacecraft. This comparison is useful to assess the domain gap caused by different image sources and lighting conditions.

The analysis continues with the Relative Camera Pose Estimation task. We begin with the benchmark dataset ScanNet. Then, the results of the space-related datasets are examined. For SPEED and SPEED+, the focus is on how changes in camera perspective, background presence, and varying illumination conditions influence performance. The SPEED-UE-Cube dataset is used to simulate a scenario closer to real space operations. Table 5.1 gives an overview of the metrics, datasets and conditions evaluated.

The second part of the chapter presents the results of the Knowledge Distillation.

It describes the experimental setup and the metrics used during training. Then, it presents the performance of the student model compared to the teacher model in the task adopted. A brief comparison of the computational performance of the different methods is provided.

Task	Dataset	Conditions
Homography	SPEED	Synthetic images
	SPEED+	lightning variations
Pose	SPEED	different levels of visual overlap, background
	SPEED+	lightning variations
	SPEED-UE-Cube	trajectory

Table 5.1: Overview of tasks, datasets, and experimental conditions.

5.1 Feature Detection and Matching Results

In implementing the algorithms and models, we utilized publicly available implementations from external libraries.
Classical feature-matching methods are executed using the *OpenCV* [72] library. The matching is made with the K-NN algorithm with the ratio test, as detailed in section 2.4.1. We used the default hyperparameter configurations specified in the library, modifying only the maximum number of features detectable. Table 5.2 illustrates the setup.

Method	Max Features Extracted	Type of Matching
SIFT	1024	k-nn (k=2, Ratio threshold = 0.8)
ORB	1024	k-nn (k=2, Ratio threshold = 0.8)
AKAZE	1024	k-nn (k=2, Ratio threshold = 0.8)

Table 5.2: Hyperparameters of SIFT, ORB and AKAZE

For LightGlue, we adopt the official implementation and pre-trained weights released by the original authors, following the default configuration specified in Table 5.3. As previously mentioned in the section 2.5.2, this model is designed exclusively for the matching phase and, therefore, requires an external model for feature detection. In this work, we employ SuperPoint as the supporting feature extractor.
LoFTR is implemented with the Kornia library [73] using its default settings. Unlike LightGlue and traditional methods that focus on sparse key points, LoFTR

generates dense matches, resulting in a higher number of correspondences. To ensure a fair comparison among the models, we impose a limit of 1,024 to the number of matches that can be predicted.

Parameter	Value
Extractor	SuperPoint
Max Num Keypoints	1024
Detection Threshold	0.0005
Number of Layers	9
Depth Confidence	0.95
Width Confidence	0.99
Filter Threshold	0.1

Table 5.3: Configuration of the LightGlue model

5.1.1 Homography estimation

In the homography estimation task, the objective is to identify the planar transformation that geometrically relates two distinct images. The correspondences obtained through FDM are used to estimate the homography matrix H, which encodes the transformation.

The homography is estimated with the function *findHomography* [74] of *OpenCV* library. The robust method adopted to improve the accuracy of the estimation is the *Least-Median-of-Squares (LMEDS)* algorithm.
The accuracy is evaluated by comparing the estimated homography $\tilde{H}$ with the ground truth transformation $\hat{H}$. The assessment is based on the following metrics:

- **Mean reprojection error**: The four image corners' coordinates are taken. They are transformed using $\hat{H}$ and $\tilde{H}$. Then the mean Euclidean distance between the two projected sets is computed. A low reprojection error indicates a precise estimation.
- **Area Under the Curve (AUC)**: The AUC curve of the mean reprojection error is used to evaluate homography estimation quality at pixel thresholds of 1, 3, and 5 pixels. The pixel threshold defines the acceptable reprojection error for a point to be considered accurately projected. If the mean reprojection error falls below the given threshold, the predicted homography is correct.
- **precision (P)**: It measures the percentage of correctly matched points relative to the ground truth matches. The target correspondences are computed from the actual homography matrix $\hat{H}$.

HPatches

The HPatches dataset is a well-established benchmark designed to evaluate the performance of feature extraction and matching methods under changing viewpoints and illumination conditions of the same scene.
Due to differing image resolutions, all images were resized such that their smaller dimension equal 480 pixels while preserving their original aspect ratio. The ground truth Homography H is adapted to the resizing.

In this dataset, AI-based models outperform traditional algorithms in homography estimation, which is supported by the results shown in table 5.4. DL models achieve more accurate homography estimations and precision in the correspondences predicted. SIFT has the highest accuracy among traditional algorithms, reducing the performance gap with SuperPoint combined with LightGlue and LoFTR.

Compared to existing literature, it is interesting to note that, as reported in [51], SuperPoint combined with LightGlue achieves a homography estimation accuracy of 79.6% in terms of AUC@5 pixels and a precision of 88.9%. LoFTR, on the other hand, reaches an AUC@5 pixels of 78.8% with a precision of 92.7%. These results are obtained using the RANSAC estimator, which differs from the estimation methods adopted in our evaluation, potentially contributing to the observed performance differences.

Method	**AUC**			**P**(%)	**Mean error**
	@1px	**@3px**	**@5px**		
SIFT	52.6	76.7	81.7	69	45.3
ORB	30.3	60.7	68.6	46	112.8
AKAZE	35.7	69.3	77.6	71	55.5
SuperPoint+LightGlue	**64.3**	**90.7**	**94.3**	**96**	11.7
LoFTR	63.5	89.1	92.4	95	**8.1**

Table 5.4: Results for the HPatches dataset

Table 5.5 summarizes the average number of key points extracted, the average number of predicted matches, and the repeatability score. As expected, the LoFTR model predicts the highest number of correspondences due to its dense matching nature. LightGlue follows, outperforming classical methods in terms of both match quantity and repeatability. SuperPoint produces highly repeatable and easily matchable feature points. Although SIFT and ORB detect a high number of interest points, only a smaller portion of these points are matched.

Method	**Avg Kpts**	**Avg Matches**	**Rep.**(%)
SIFT	939.7	263.3	29.8
ORB	**1009**	214.7	21.3
AKAZE	776.3	217.6	32.7
SuperPoint+LightGlue	994.6	561.1	**57.1**
LoFTR	-	**981.3**	-

Table 5.5: Average number of key points extracted, average number of matches and average Repeatability score for HPatches dataset.

Figure 5.1 shows an example image pair from the HPatches dataset, illustrating the key points extracted and the correspondences predicted by SIFT and LightGlue. In these images, key points that are extracted but not matched are coloured light blue. The green lines represent pairs of key points correctly matched, while the red lines indicate incorrect correspondences. This notation will also be used in subsequent images.
From the image, it is clear that SIFT produces fewer reliable correspondences, and the incorrect matches exhibit larger errors than those generated by LightGlue.

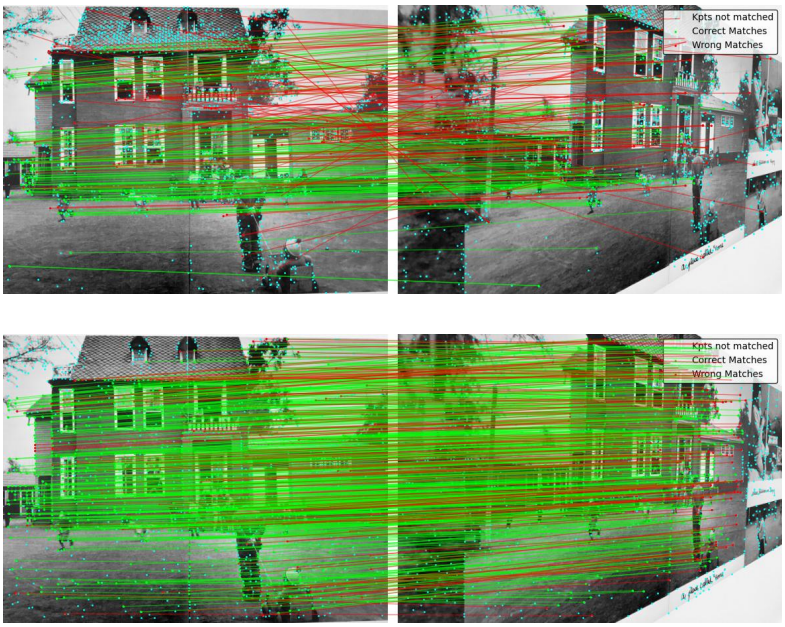

Figure 5.1: Image sequence in the HPatches dataset. The first image contains the output of the SIFT method. The second one refers to LigthGlue combined with SuperPoint.

SPEED

The SPEED dataset is the first space-related dataset analyzed. Due to the limited number of real images available, only the synthetic domain is considered.

Table 5.6 compares the different methods for the homography estimation on SPEED images. Unlike the benchmark case, LightGlue performs comparably to classical methods in the estimation while achieving higher precision and a lower mean reprojection error. The observed performance drop can be due to the decreased number of extracted and matched key points compared to the benchmark dataset, as present in the table 5.7.
This is expected in space environments, where most images have a black background that lacks relevant visual information for effective feature extraction. Additionally, noisy pixel-level details in the background can lead to false positive matches, further complicating the task. The challenge intensifies when the spacecraft is farther from the camera, as this reduces the available visual information.
LoFTR, however, maintains a high number of correspondences despite these conditions. Its average number of matches remains similar to the benchmark case, while other methods experience a significant drop. Homography estimation accuracy appears to be influenced by the high number of inlier points used for estimation. With a reduction in the number of matches, the influence of outliers increases.
Classical methods struggle under these conditions, particularly ORB and AKAZE, which exhibit lower precision and a higher mean reprojection error. This highlights the limitations of traditional handcrafted approaches when applied to challenging space imagery.

Method	**AUC**			**P**(%)	**Mean error**
	@1px	**@3px**	**@5px**		
SIFT	33.7	46.5	53.8	82	65.6
ORB	9.5	28	38	37	266.5
AKAZE	19.9	33.8	41.5	77	2130.4
SuperPoint+LightGlue	25.9	44.5	54	**92**	31.2
LoFTR	**63.9**	**85.3**	**89.1**	91	**29.3**

Table 5.6: Homography estimation results on the SPEED dataset

To illustrate the different behaviours, we provide examples of analyzed image pairs, where extracted key points and resulting correspondences are visualized to highlight the distinct outputs of each method, as in the figure 5.2.

Method	**Avg Kpts**	**Avg Matches**	**Rep.**(%)
SIFT	533.6	196.5	45.6
ORB	**820.9**	220.1	27.8
AKAZE	450.5	157.6	**59.1**
SuperPoint+LightGlue	652.9	276.7	44.15
LoFTR	—	**968.7**	—

Table 5.7: Average number of features extracted and matched, average repeatability value for tests on the SPEED dataset.

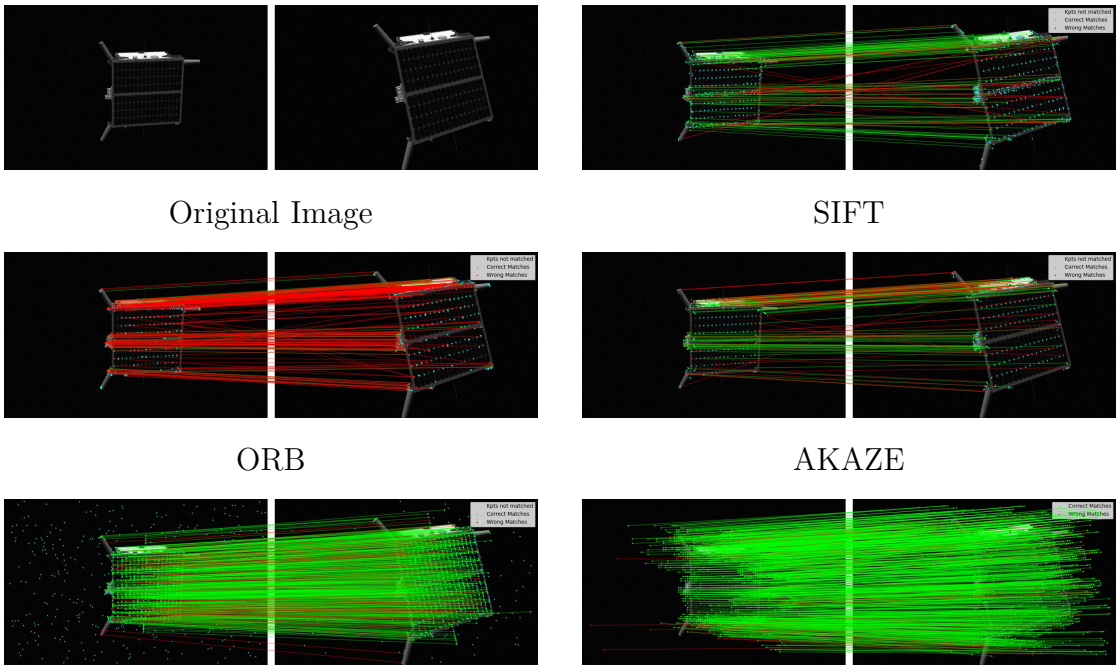

Figure 5.2: Image pair from the SPEED dataset with the predictions made by each method. The blue dots are interest points extracted but not matched, the green lines are correctly predicted correspondences, while the red lines are wrong matches.

SPEED+

The second space-related dataset analyzed is SPEED+. The evaluation of this dataset follows the same methodology used for the SPEED dataset, but it introduces several key differences that expand the analysis. Specifically, SPEED+ provides images from three distinct domains: *Synthetic*, *Lightbox*, and *Sunlamp*, the last two are not present in the SPEED dataset. The purpose of these tests is to evaluate how the FDM methods perform when the characteristics of the images change.

Synthetic This section presents the results obtained using images from the synthetic domain. In space use-cases, synthetic images play an important role in training and evaluating AI models. Due to the limited availability of real space imagery and the challenges associated with reproducing it in the laboratory, synthetic domains remain one of the main sources for developing and testing such systems.
The synthetic images in the SPEED+ dataset are reproduced in the same way that the synthetic images in SPEED but present some differences.
A key difference is the simulated camera-object distance. In SPEED, the target distance ranges from 3 to approximately 40 meters, whereas in SPEED+, the target is positioned between 2 and 10 meters along the z-axis, aligned with the camera's line of sight. Consequently, the spacecraft in SPEED+ synthetic images generally can appear larger and more visible compared to SPEED. This aspect seems to influence the results, showing an overall increase in the accuracy of the estimation, as presented in the table 5.8.

Method	**AUC**			**P**(%)	**Mean error**
	@1px	**@3px**	**@5px**		
SIFT	44.5	64.7	75.8	81	18.9
ORB	11.7	35.4	48.8	38	49.7
AKAZE	23.8	44	55	77	79.8
SuperPoint+LightGlue	32.4	56	67.1	96	**7.9**
LoFTR	**76.7**	**92.6**	**95.7**	**97**	51.9

Table 5.8: Results dataset SPEED+, domain *Synthetic*

Lightbox and Sunlamp The second distinction between the two datasets lies in the real domains: Lightbox and Sunlamp. These images are prepared to simulate lighting effects representative of those encountered in Earth's orbit. Images exhibit extreme illumination conditions, including shadows, high contrast due to overexposure, and regions with insufficient light. This makes it very difficult to extract meaningful key points. Additionally, noise can impact detection and matching accuracy.

Figure 5.3 compares the pixel intensity histogram distributions of images from SPEED and SPEED+. For both synthetic sets, the distributions are quite similar and are concentrated around lower values, which suggests that the images are generally darker. In the real sources, particularly under sunlamp lighting, the distribution tends to shift towards slightly higher values. Notably, the sunlamp set has a distinct peak at the extremely high end of the spectrum, indicating the presence of very bright regions within those images.

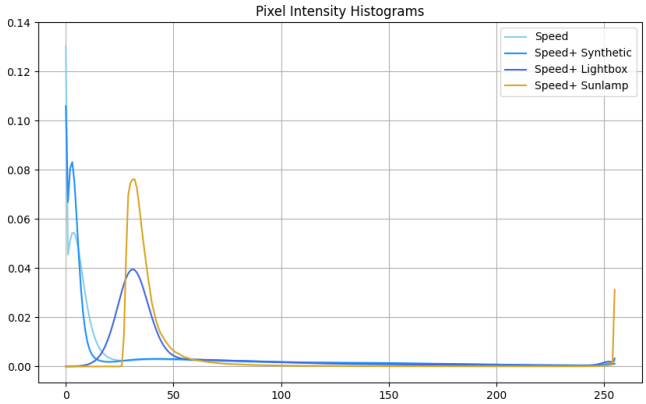

Figure 5.3: Histogram of pixel intensity distributions

The decrease in accuracy is shown in the results presented in Table 5.9. All methods exhibit a drop in homography estimation accuracy when moving from synthetic to real domains. However, deep learning models still demonstrate greater adaptability compared to classical methods, maintaining relatively strong performance despite challenging conditions. It is important to compare the numerical results with the qualitative image examples. From Figures 5.5 and 5.6, it can be observed that, although DL methods produce more accurate estimations, they are highly sensitive to noise and lighting effects. In fact, many of the extracted and matched points do not originate from the spacecraft itself, but rather from the surrounding background. However, in these experiments, even feature points that are not directly relevant, meaning they do not belong to the target body, can still contribute positively to the estimation of the geometric transformation if they are correctly matched.
A comparison of the *AUC@5* pixels values of each method across the different domains is illustrated in the graph 5.4.

Method	AUC - Light			AUC - Sun			P(%)	Mean Error
	@1px	@3px	@5px	@1px	@3px	@5px	Light | Sun	Light | Sun
SIFT	27.2	43.7	50.9	7.9	29.9	43.2	80 | 80	95.36 | 56.7
ORB	6.0	22.1	32.3	2.1	16.3	29.0	36 | 36	218.0 | 46.6
AKAZE	13.7	26.5	33.4	2.6	15.8	25.8	76 | 75	151.6 | 57.5
SuperPoint+LightGlue	18.4	37.7	52.5	**37.1**	**78.7**	**88.0**	**93** | **96**	**32.4** | **8.43**
LoFTR	**44.5**	**73.3**	**80.1**	39.2	77.2	86.5	84 | 95	55.6 | 18.3

Table 5.9: Homography estimation results on SPEED+ dataset for Lightbox and Sunlamp domains.

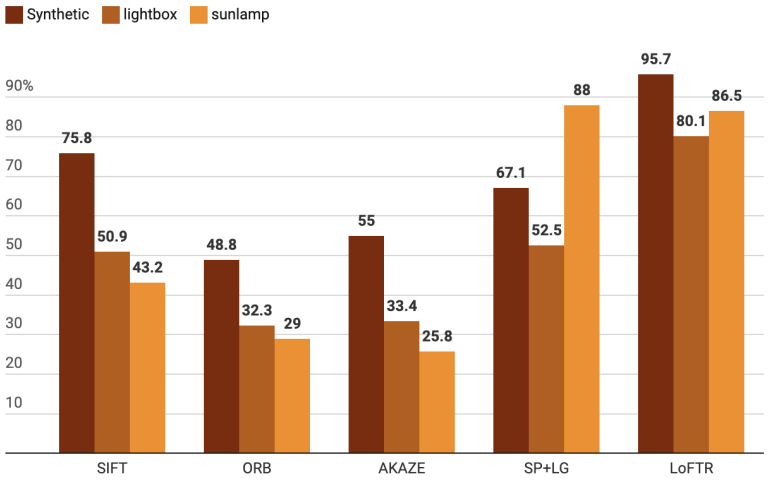

Figure 5.4: *AUC@5* pixels of mean re-projection error on SPEED+ across different domains and methods

Table 5.10 provides further insights into the FDM performance of the evaluated algorithms. From the synthetic domain to the Lightbox and Sunlamp settings, traditional methods like SIFT and AKAZE exhibit a decrease in the number of produced matches, which correlates with their drop in accuracy. Meanwhile, LightGlue increases the number of correspondences in the Sunlamp domain, partly due to noise, which improves the estimation of homographies.

Method	Synthetic		Lightbox		Sunlamp	
	Kpts	**Matches**	**Kpts**	**Matches**	**Kpts**	**Matches**
SIFT	644.6	250.7	414.9	159.9	202.8	93.5
ORB	**935.5**	272.3	**781.3**	223.9	**780.1**	273.7
AKAZE	533.2	186.5	355.8	135.8	256.4	113.4
SuperPoint+LightGlue	655.9	332.3	478.4	230.7	763.4	377.5
LoFTR	-	**1000.2**	-	**919.1**	-	**989.9**

Table 5.10: Average key points extracted and matched for each method across the tree sets.

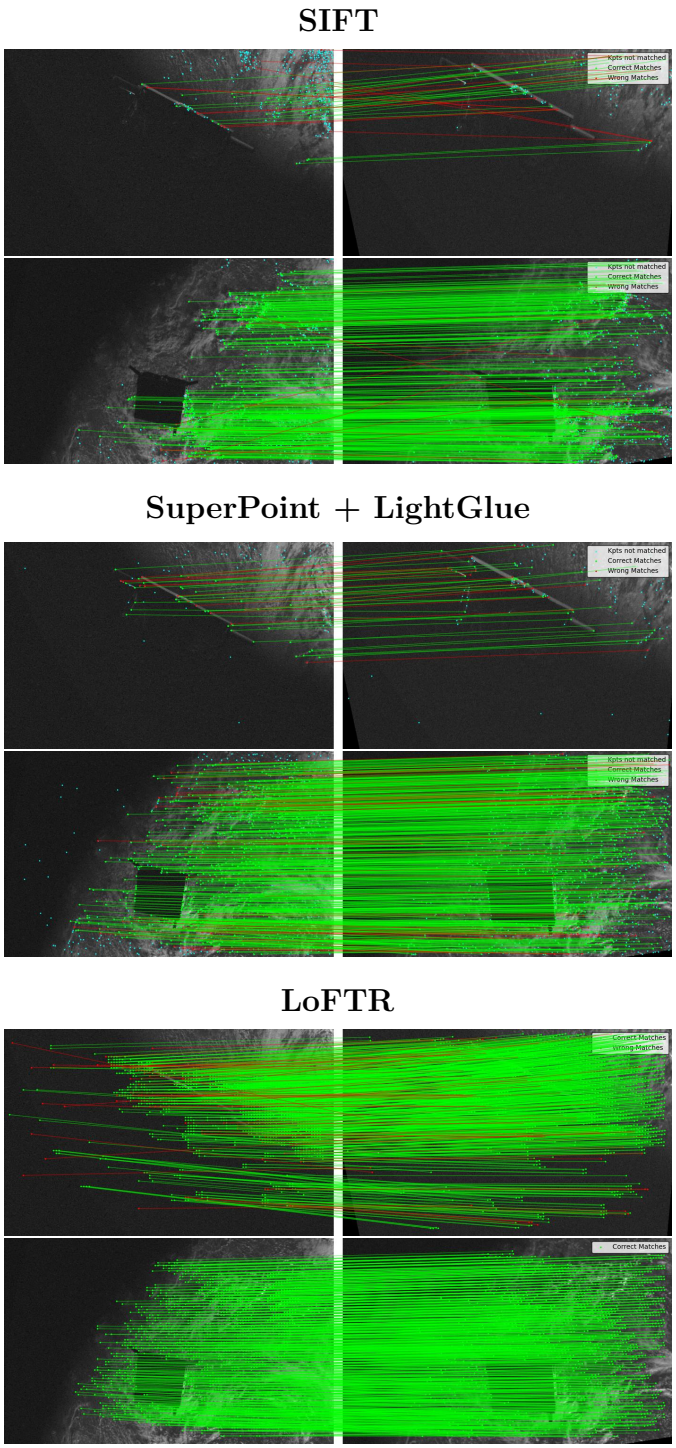

Figure 5.5: Image pairs generated from the Lightbox set. Outputs of SIFT, SuperPoint + LightGlue, and LoFTR are shown.

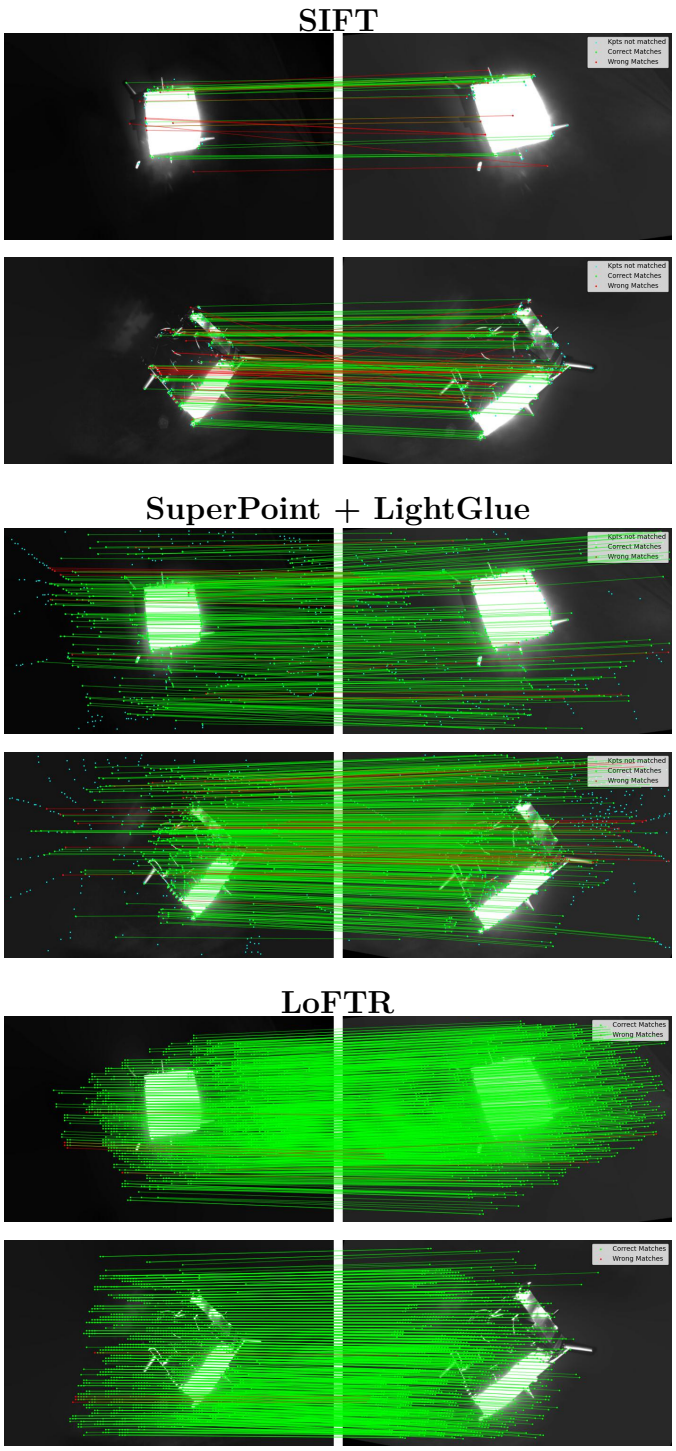

Figure 5.6: Image pairs from the Sunlamp set. From top to bottom: SIFT, LightGlue, and LoFTR outputs.

5.1.2 Relative Camera Pose Estimation

Relative pose estimation is the process of determining the transformation between two different frames. By identifying correspondences between points in two distinct image views of the same scene, it is possible to compute the transformation from the coordinate system of the first camera to that of the second. This consists of a rotation matrix R and a translation vector t, representing the orientation and relative displacement between the two camera centers.

The estimation is made employing the *findEssentialMat* [75] function from the *OpenCV* python library. It solves the essential matrix E from the key points matched in the two images and the camera's intrinsic properties, as explained in the section 2.7.2.
This function admits different robust estimator algorithms. Between those, we used the MAGSAC method, a variant of the widely used RANSAC, to effectively filter out incorrect matches. From our tests, MAGSAC is revealed to be, in general, more accurate and faster than the traditional RANSAC. The parameters of the algorithm for the computation are reported in the table 5.11.

Table 5.11: Hyperparameters for the Essential matrix estimation

Hyperparameter	**Value**
Robust estimator	MAGSAC
pixel threshold	0.5
confidence	0.99999
Maximum number of iterations	5000

The pixel threshold represents the maximum distance from a point to the corresponding epipolar line in pixels. Beyond that, the point is considered an outlier, and so it is not used for computing the final estimation. The confidence parameters specify the desirable level of confidence that the estimated E matrix is correct.
Once the essential matrix is computed, the relative rotation R and translation t are recovered using the *recoverPose* function [76].
The Evaluation metrics employed include the Area Under Curve (AUC) for angular error at angle-thresholds of 5°, 10°, 20°. Additionally, we report the average angular errors for the rotation and translation.

ScanNet

Thanks to its large volume of available data, the ScanNet dataset is widely recognized as a reliable benchmark for various tasks, including relative pose estimation. The tests are conducted on 1500 image pairs selected from the test set of the ScanNet dataset. The image pairs used are the same as indicated in the evaluation phase of the model LoFTR [52] and SuperGlue [49]. Images have a resolution of 640×480.

As confirmed by our experiments, AI models improve over traditional methods. Table 5.12 summarizes the results, clearly demonstrating that DL-based approaches achieve higher accuracy across all angular thresholds and exhibit lower average translation and rotation errors.
These results are consistent with those reported in the literature. In [49], the results obtained using SIFT (NN + ratio test) are reported as 6.7%, 15.7%, and 28.67% for the 5°, 10°, and 20° thresholds, respectively. In [52], the corresponding results for LoFTR are 22.06%, 40.8%, and 57.6%. The same set of images was used in both studies, and the RANSAC algorithm was employed for pose estimation.
For the remaining FDM methods evaluated, ORB, AKAZE, and LightGlue, there are no known benchmarks on this dataset available in previous studies.

Method	**AUC**			**Avg T. Error (°)**	**Avg R. Error (°)**
	@5°	**@10°**	**@20°**		
SIFT	6.10	13.77	23.46	35.11	41.64
ORB	5.67	13.44	23.42	34.17	43.23
AKAZE	4.63	10.81	19.92	35.65	40.03
SuperPoint + LightGlue	19.05	36.99	54.08	15.36	**12.18**
LoFTR	**20.62**	**38.61**	**54.66**	**15.23**	13.23

Table 5.12: Relative pose estimation results on the ScanNet dataset: AUC at 5°, 10°, and 20°, along with average translation and rotation errors.

Table 5.21 presents further statistics about the average number of key points extracted and matched and the repeatability score for each method. LigthGlue and LoFTR are more consistent in retrieving correspondences. We report the prediction made by SIFT, LightGlue and LoFTR in a sample of the ScanNet dataset in figure 5.7.

Method	Avg Kpts Extracted	Avg Matches	Avg Repeatability (%)
SIFT	388.39	58.76	21.68
ORB	**993.28**	101.47	16.92
AKAZE	250.96	42.16	25.31
SuperPoint+LightGlue	661.96	**164.80**	**27.77**
LoFTR	–	818.67	–

Table 5.13: Average key points extracted, matches, and repeatability on ScanNet.

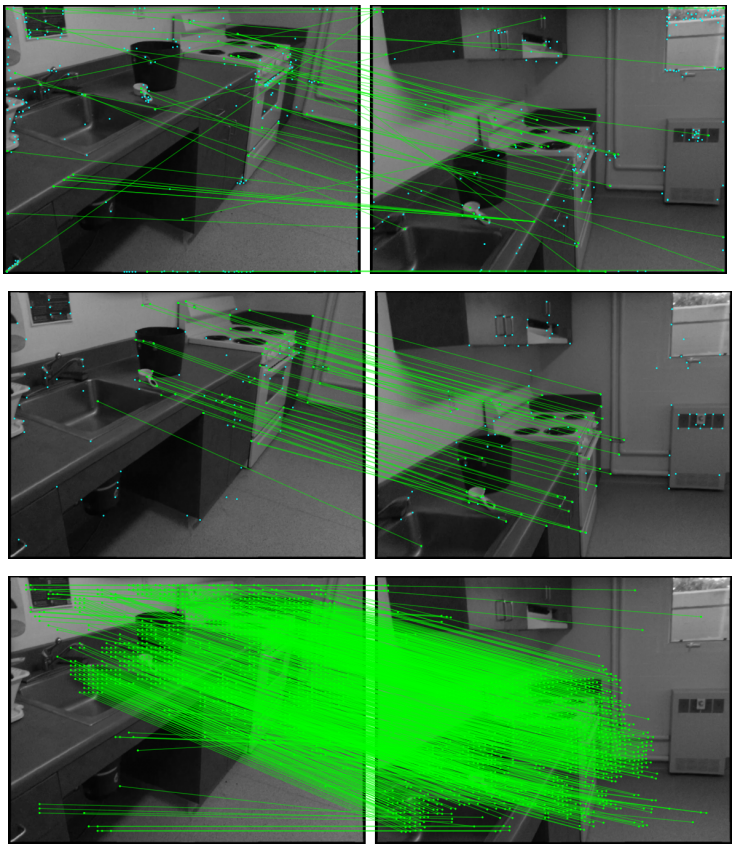

Figure 5.7: Comparison of SIFT, SuperPoint+LightGlue and LoFTR on the same images from ScanNet.

SPEED

In this section, we present the results obtained from tests conducted on the SPEED dataset.
The reported results are categorized based on the difficulty level defined in the section 4.1.2. As explained, this categorization allows for the evaluation of the FDM process on image pairs that either share a similar perspective (*easy* case) or display the object from significantly different viewpoints (*hard* case).

Table 5.14 presents the AUC scores for angular errors in the easy scenario. The combination of LightGlue with SuperPoint achieves the best performance, significantly surpassing both classical methods and LoFTR. This result contrasts with the homography estimation task, where LoFTR has a higher performance.

Method	**AUC**			**Avg T. Error (°)**	**Avg R. Error (°)**
	@5°	**@10°**	**@20°**		
SIFT	2.74	6.01	8.96	32.4	64.9
ORB	2.43	5.83	9.04	32.4	59.5
AKAZE	2.86	6.12	9.04	32.6	60.8
SuperPoint + LightGlue	**10.17**	**18.74**	**30.48**	**26.1**	**25.2**
LoFTR	3.99	9.62	18.93	30.2	22.7

Table 5.14: AUC and average pose estimation errors on the *easy* scenario of the SPEED dataset.

Figure 5.8 shows a qualitative comparison of predicted matches for a sample image pair using SIFT, LightGlue, and LoFTR. This visualization reveals that LoFTR generates the highest number of matches. However, many of these matches stem from background noise rather than valid correspondences. SuperPoint also captures several points from the background, but LightGlue effectively filters these during the matching phase, focusing on accurate correspondences on the spacecraft body. In contrast, SIFT demonstrates greater robustness against background noise, as all extracted feature points are located on the object itself. Nevertheless, it struggles during the matching phase, resulting in some incorrect matches.

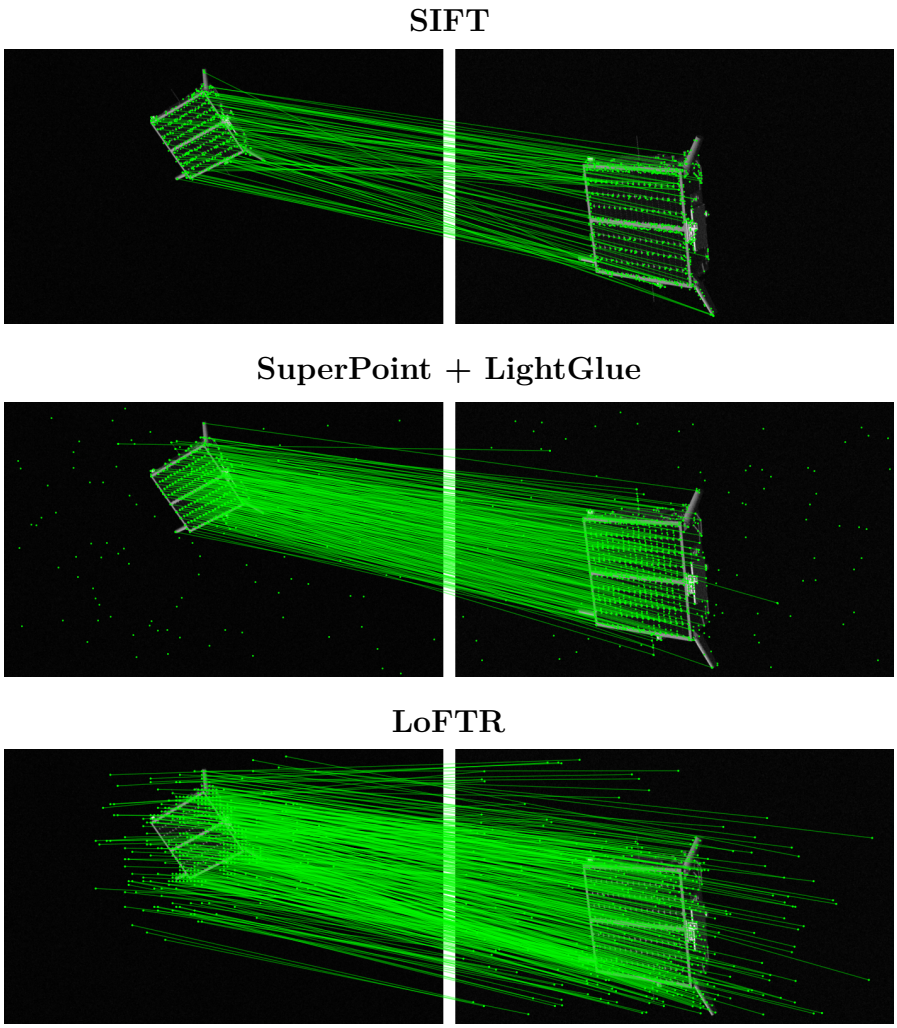

Figure 5.8: Outputs of the method in SPEED-synthetic images (*easy* case)

Table 5.15 presents the results for the medium and hard cases, respectively. As expected, the performance of all evaluated methods declines as the difficulty of the image pairs increases. Among the tested methods, LightGlue still remains the most accurate. Figure 5.9 supports this by showing how the average error in estimating relative translation and rotation increases. Additionally, the histogram in Figure 5.10 presents the AUC@20° for each method across different difficulty levels. The results emphasize the performance gap between classical and deep learning-based approaches.

Method	Medium			Hard		
	AUC@5°	**AUC@10°**	**AUC@20°**	**AUC@5°**	**AUC@10°**	**AUC@20°**
SIFT	0.55	0.98	1.46	0.29	0.51	0.75
ORB	0.52	0.97	1.50	0.00	0.21	0.51
AKAZE	0.34	0.72	1.04	0.10	0.27	0.49
SuperPoint + LightGlue	**3.44**	**6.81**	**12.7**	**0.70**	**1.81**	**4.00**
LoFTR	0.70	1.31	2.56	0.20	0.37	0.80

Table 5.15: AUC results for relative pose estimation on the *medium* and *hard* datasets.

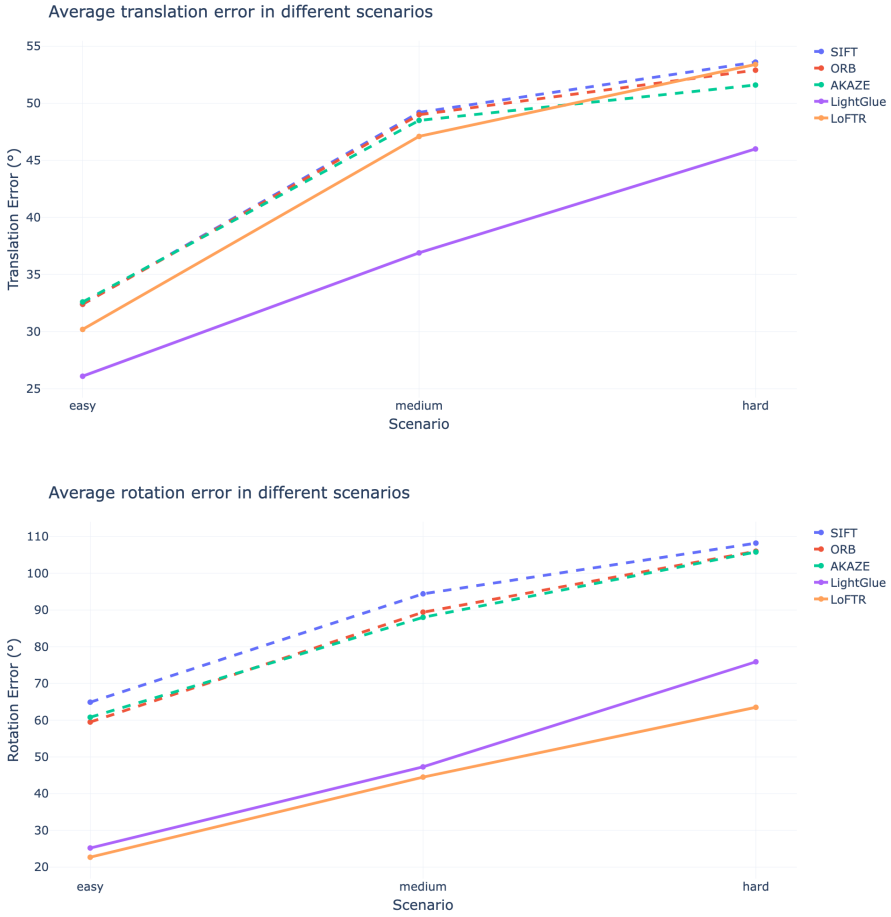

Figure 5.9: Average translation error (top) and average rotation error (bottom) of each method across the different cases.

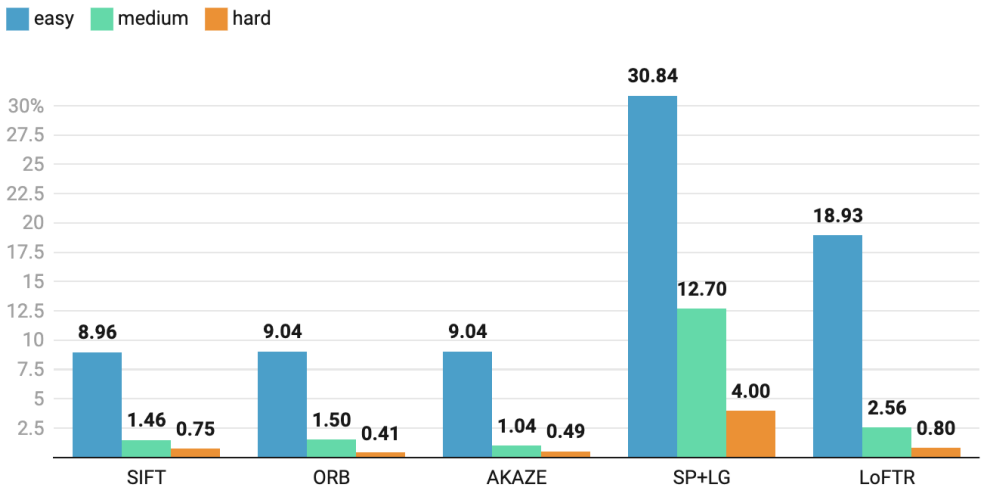

Figure 5.10: The histogram reports the AUC@20° for angular error in the relative pose estimation on the SPEED datasets with different levels of difficulty in the image pairs.

Finally, Table 5.16 reports the average number of key points and correspondences across the three difficulty levels. Compared to the homography transformations, the number of matched key points is lower in all the cases. The number of matches is higher in the easy scenario and decreases in the medium and hard cases, which aligns with expectations. As visual overlap decreases, it becomes more difficult for FDM methods to identify points that are present and matchable in the other image.

Method	Easy		Medium		Hard	
	Avg Kpts	Avg Matches	Avg Kpts	Avg Matches	Avg Kpts	Avg Matches
SIFT	212.29	33.49	207.68	26.02	209.98	24.78
ORB	809.49	39.69	804.38	24.13	810.29	19.38
AKAZE	152.9	19.9	150	12.9	149.2	10.9
SuperPoint+LightGlue	229	63	226.5	44.2	230.4	28.8
LoFTR	–	417.4	–	348.5	–	309.2

Table 5.16: Average key points extracted and matched for each method across easy, medium, and hard cases.

Figure 5.11 shows an image pair from the SPEED dataset along with the outputs of SIFT and LightGlue. These images were used to generate the visualization of the relative camera positions shown in Figure 5.12. The visualization reconstructs the scene's geometry using the first camera as the reference point, with the second camera positioned relative to it. It is important to note that the distance between the cameras in this visualization is arbitrary. The estimated translation vector

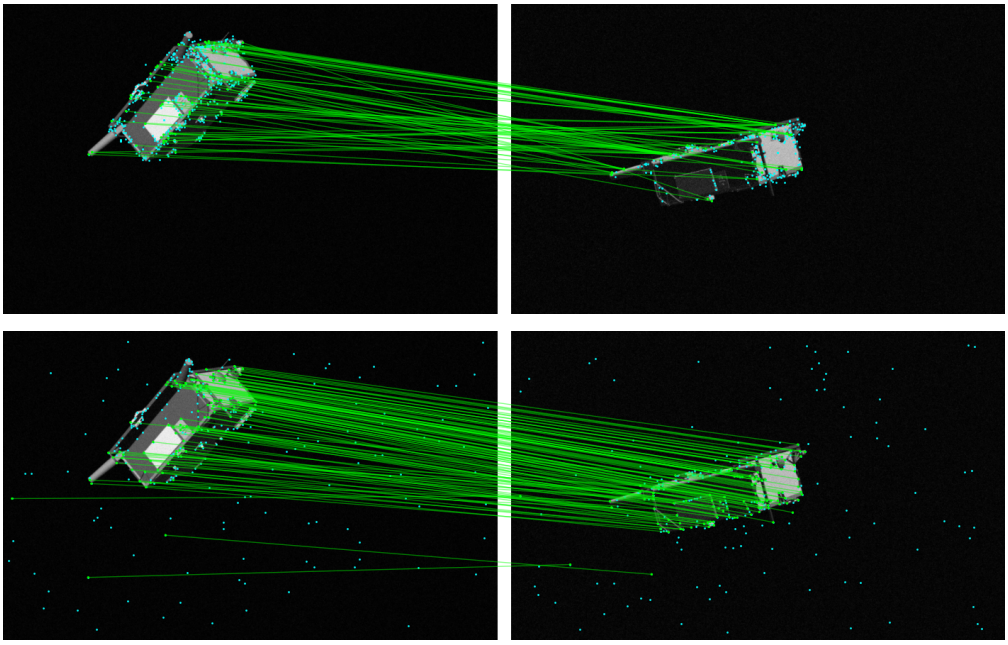

Figure 5.11: Sample of image pair from the SPEED dataset. The figures compare the output of SIFT (top) and LightGlue (bottom).

only provides information about the direction between the two viewpoints, not the actual distance. Without additional data, it is not possible to recover the absolute scale from the images alone.
In the reconstruction of the scene, the reference camera (marked in red) associated with the first image (on the left) is positioned at the origin of the coordinate system. The green camera represents the ground truth position and orientation of the second camera (on the right) in relation to the first. The dashed line indicates the direction of the view from the first camera. The additional cameras (in orange and blue) depict the poses estimated by LightGlue and SIFT, respectively, following the relative pose estimation process.
In this example, LightGlue delivers a significantly more accurate estimation, with its camera pose nearly overlapping the ground truth. Specifically, LightGlue has a translation error of 9.64° and a rotation error of 11.1°. In contrast, SIFT exhibits much higher errors for the same image pair, with a translation error of 89° and a rotation error of 38.6°.

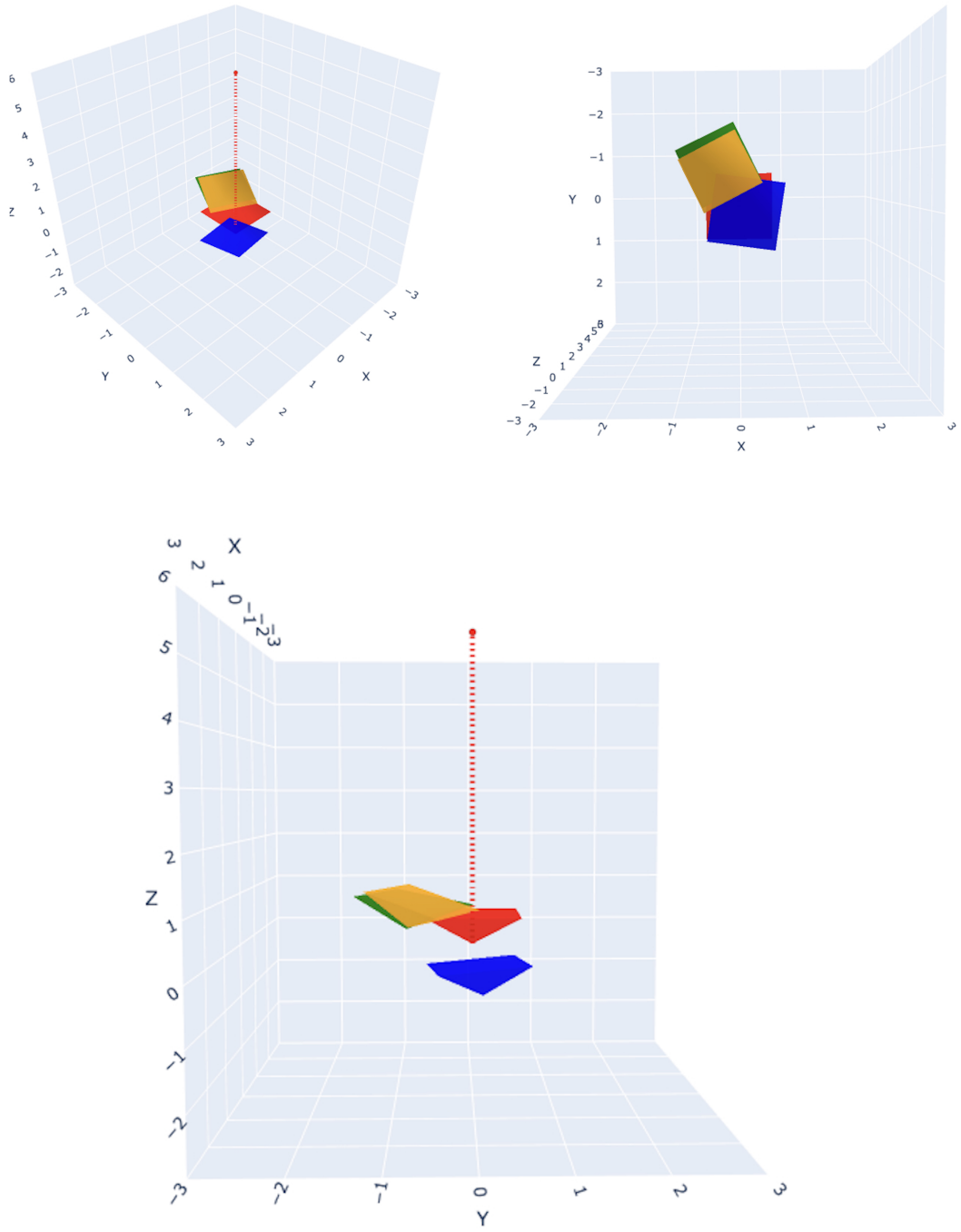

Figure 5.12: The images are visualizations of the camera poses associated with the image pair 5.11 from different views. The red camera is the reference (first image), the green is the ground truth pose of the second camera, the orange and blue pyramids represent the estimated poses obtained with LightGlue and SIFT, respectively.

Impact of Background Another crucial factor considered in these experiments is the presence or absence of background in the images. The presence of background typically negatively influences relative camera pose estimation since it strictly relies on accurately matched features specific to the spacecraft. Points extracted from the background do not represent the satellite's relative pose, and it can significantly increase the number of false positives, leading to inaccurate pose estimations. To assess the impact of its presence in the images, we conducted additional tests by creating image pairs where the background remained visible in both. These pairs maintained the same relative translation and orientation values as in the *easy* case. The results in Table 5.17 indicate that relative pose estimation is generally less accurate in scenes with background elements. Figure 5.13 illustrates the behaviour of SIFT, SuperPoint with LightGlue, and LoFTR in two different scenes containing background information. As observed in these examples, the background distracts the FDM models from focusing on the spacecraft, particularly in the cases of SP+LG and LoFTR. This leads to correspondences that do not necessarily contribute to an accurate estimation of the relative camera pose, potentially increasing estimation errors.

Method	**AUC@5°**	**AUC@10°**	**AUC@20°**	**Avg. T. Err**	**Avg. R. Err**
SIFT	0.8	2	4.2	35.4	98.7
ORB	0.85	1.8	3.5	34.7	105.5
AKAZE	0.24	0.8	2	35.7	105
SuperPoint + LightGlue	**5.6**	**9.9**	**15.7**	32.5	42.4
LoFTR	2	4.1	8.5	39.8	40.2

Table 5.17: AUC results for different feature detection methods on the images with background.

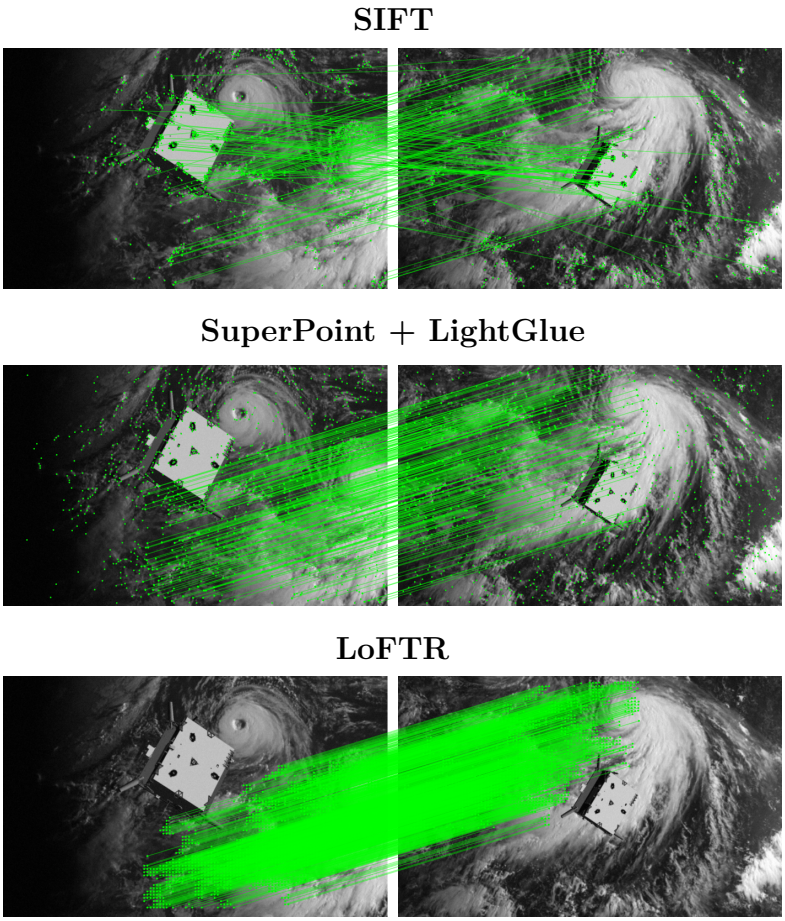

Figure 5.13: Outputs of SIFT, SuperPoint + LightGlue, and LoFTR on SPEED-synthetic images with background.

SPEED+

We evaluated the studied methods on the SPEED+ dataset, considering both the synthetic domain and the real-image domains (lightbox and sunlamp). Image pairs are generated using the same relative translation and rotation intervals as in the SPEED dataset.
For the following tests, we focused on the *easy* scenario to analyze the impact of the domain gap on performance.

Synthetic Table 5.18 presents the AUC values for the *synthetic* domain. In this case, only images without background are considered. Compared to the synthetic domain of the SPEED dataset under the same conditions, the results show a significant improvement in the SPEED+ dataset, particularly for the LightGlue model. This performance gain is likely attributed to the characteristics of the SPEED+ dataset. As previously mentioned, the camera-spacecraft distance in SPEED+ is generally smaller than in the SPEED dataset, which may positively impact the feature detection and matching process and lead to more accurate relative pose estimation.

Method	AUC@5°	AUC@10°	AUC@20°	Avg. T Err	Avg. R Err	Matches
SIFT	4.62	8.72	16.01	33.9	49.9	58.9
ORB	3.19	7.05	14.13	33.8	43	63.7
AKAZE	2.87	6.25	12.14	36.6	47	35.2
SuperPoint + LightGlue	**20.12**	**34.57**	**50.47**	15.0	12.1	115.7
LoFTR	8.72	16.7	28.21	25.5	19.7	429.4

Table 5.18: AUC results for different feature detection methods on the Synthetic domain with no background (SPEED+ dataset)

Lightbox and Sunlamp FDM methods struggle with the challenging characteristics of real-image domains, leading to lower accuracy in these conditions. The *sunlamp* domain, despite its extreme lighting conditions, obtains similar results to the *lightbox* domain for deep learning-based models. This suggests that although the intense illumination variations in the *sunlamp* domain pose a significant challenge, the presence of strong contrasts may still allow AI-based models to extract and match meaningful features. Deep learning models remain significantly more robust than traditional algorithms in these extreme cases. The results for the *lightbox* and *sunlamp* domains are reported in Table 5.19.

Method	Lightbox			Sunlamp		
	AUC@10°	**AUC@20°**	**Matches**	**AUC@10°**	**AUC@20°**	**Matches**
SIFT	0.44	1.19	19.2	0.6	1.72	19.5
ORB	0.70	1.48	15.3	0.36	1.3	21.3
AKAZE	0.59	1.49	15.9	0.44	1.63	16.1
SuperPoint + LightGlue	**5.75**	**9.94**	38.3	**5.47**	**11.19**	68.0
LoFTR	2.63	6.07	219.4	2.97	6.12	212.8

Table 5.19: AUC results for different feature detection methods on Lightbox and Sunlamp datasets.

SPEED-UE-Cube

In the final analysis, we evaluate the results for the trajectory set in the SPEED-UE-Cube dataset. The images in this dataset pose a significant challenge for relative camera pose estimation due to the target's distance from the camera. As shown in Section 4.1.3, the spacecraft appears smaller and less prominent compared to other datasets, making keypoint detection and matching more difficult.
To improve the estimation accuracy, we cropped a patch of dimension 360×576 from the image around the spacecraft and applied upsampling to the resolution of 600×960, improving the visibility of spacecraft details compared to the original resolution. To refine essential matrix estimation, we fine-tuned the MAGSAC algorithm's pixel threshold, obtaining an optimal value of 0.15, in contrast to the 0.5 threshold used in previous tests. The results are reported in table 5.20

Method	**AUC@5°**	**AUC@10°**	**AUC@20°**	**Avg T Error (°)**	**Avg R Err (°)**
SIFT	1.33	2.7	5.33	72.7	38.9
ORB	0.93	2.7	6.29	67.33	29.32
AKAZE	1.49	3.78	7.67	63.2	31.2
SuperPoint + LightGlue	1.23	3.18	7.15	58.3	18,8
LoFTR	**3.17**	**8.2**	**17.3**	**38.3**	**11.6**

Table 5.20: Results on SPEED-UE-Cube dataset

From the tests and the figure 5.14, it emerges that the FDM methods perform well in estimating the relative orientation, with most errors concentrated at low angles. However, translation estimation remains challenging. The high translation error is primarily attributed to the scene composition rather than to inaccuracy in the FDM process. During the trajectory, the spacecraft is often far from the camera and the relative motion between consecutive frames can appear minimal. This results in near-degenerate baselines, where the camera motion is insufficient to provide strong geometric information for translation. Even when correspondences are correctly matched, such configurations make it difficult to recover an accurate

Method	Avg Kpts Extracted	Avg Matches	Avg Repeatability (%)
AKAZE	214.3	77.5	50.1
ORB	**675.1**	177.3	29.5
SIFT	203	57.9	38.3
SuperPoint+LightGlue	200.8	76.4	**66.4**
LoFTR	–	442.5	–

Table 5.21: Average key points extracted, matches, and repeatability on SPEED-UE-Cube.

relative translation.

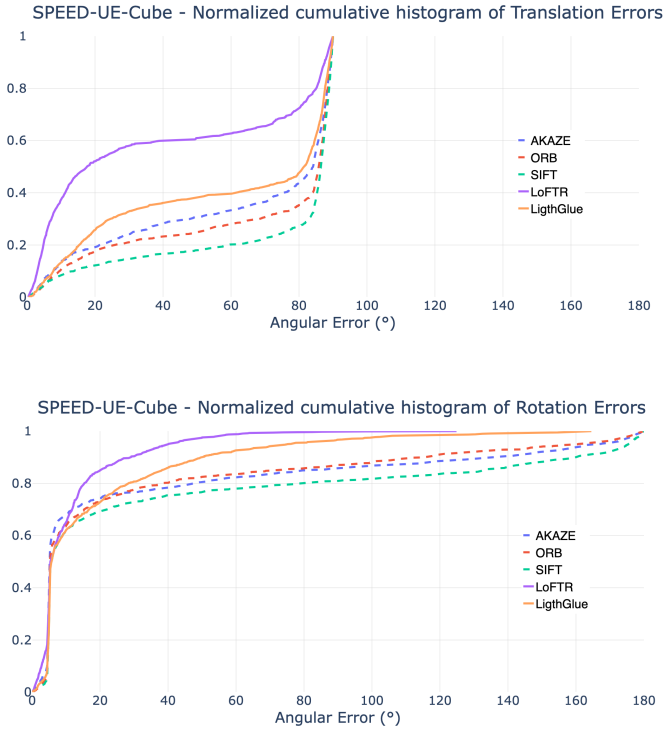

Figure 5.14: Cumulative histograms of translation errors (top) and rotation errors (bottom) on the SPEED-UE-Cube trajectory set.

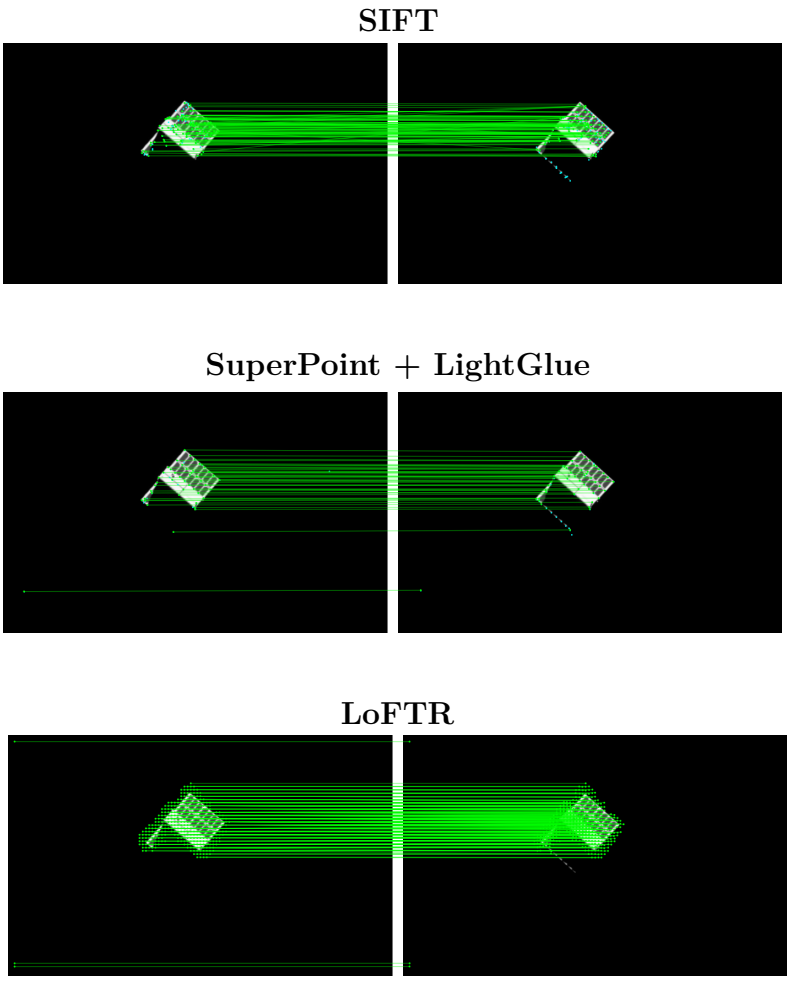

Figure 5.15: Comparison on the SPEED-UE-Cube dataset. From top to bottom: SIFT, SuperPoint + LightGlue, and LoFTR.

In Figure 5.16, we have included a visualisation of the trajectory completed by the camera around the object. The trajectory has been plotted by accumulating the relative movement of the camera between each pair of subsequent frames. In Figure (a), the ground truth trajectory is shown, starting from the labels of relative poses; however, the relative distances have been normalised to facilitate comparison with the estimated trajectory. Figure (b) contains the trajectory obtained by using the relative poses estimated through the output of LoFTR. It is important to emphasise that the estimated camera position in each frame i is obtained by multiplying the estimated relative pose with the previous ground truth position at frame $i-1$ and not with the previously estimated position at frame $i-1$. This is done to avoid accumulating errors during the trajectory and diverging excessively from the actual

one.

In a real application of Visual Odometry, the process of estimating the camera's motion is more complex and involves additional techniques to reduce the estimate error and better refine the motion. For example, several methods are based on the Kalman Filter algorithm [77], which has proven its efficacy in spacecraft GCN systems [78]. The Kalman Filter is a recursive algorithm that fuses information, like relative positions and velocities, from a measurement system over time, with predictions of the expected system behaviour made by a dynamical model. This approach allows us to estimate the system state more accurately than either the measurements or dynamics alone.

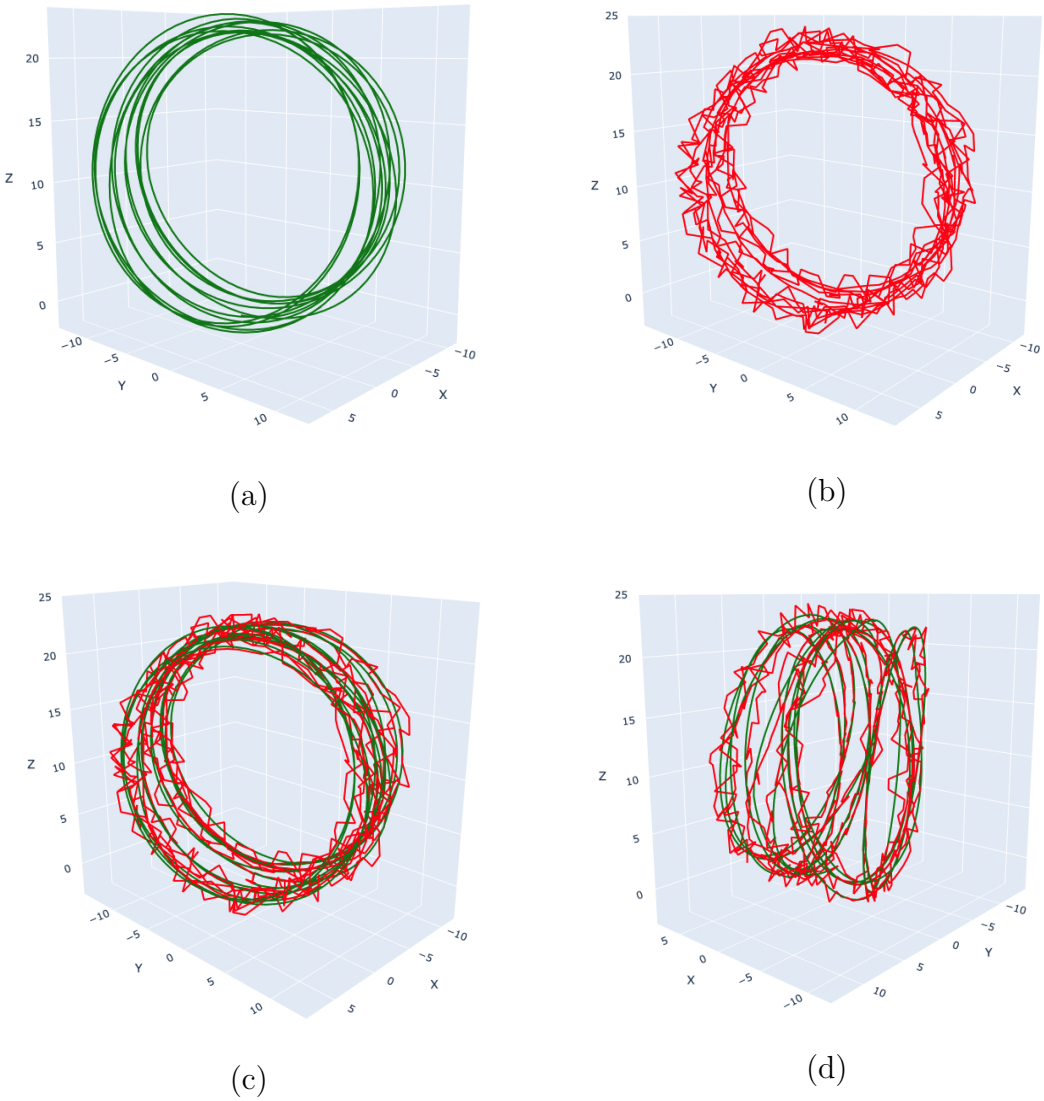

Figure 5.16: Trajectory visualizations from the SPEED-UE-Cube dataset. (a) Ground truth trajectory; (b) Estimated trajectory with LoFTR; (c-d) Comparisons of estimated and ground truth trajectories from different views

5.2 Knowledge Distillation results

5.2.1 Training results

The training procedure for the student model, as described in the previous chapter, consists of two sequential phases. The first phase involves training on synthetically generated homographies. The second phase involves fine-tuning the student model on image pairs generated based on relative pose information.
The LightGlue configuration used during training is present in Table 5.22.

Table 5.22: Configuration of the LightGlue model

Parameter	**Value**
Extractor	SuperPoint
Max Num Keypoints	128
Detection Threshold	0
Number of Layers	9
Depth Confidence	-1
Width Confidence	-1
Filter Threshold	0.1

Compared to the configuration used for testing, we disabled *points pruning* and *early exit*, which are present in the original LightGlue architecture. This decision ensures that the number of processed layers and key points remains constant, simplifying the training process for the student model, as these dynamic strategies are not replicated in the student architecture.
The following sections provide a detailed breakdown of the different training phases and their results.

Homography training

During synthetic homography training, images are transformed using randomly generated geometric transformations, controlled by hyperparameters that regulate intensity and variability. The training process consists of two phases, each using a different set of hyperparameters for homography generation:

- Training 1: It uses the same hyperparameters as the original LightGlue training.
- Training 2: It modifies transformation parameters to better align with the SPEED dataset characteristics and enhance the model's robustness to geometric distortions.

In the second training, the adjustments include reducing the translation factor to prevent the spacecraft from moving too far out of the camera's field of view, increasing the maximum rotation angle to improve adaptability to orientation changes and introducing image scaling to simulate variations in camera distance. The complete set of homography parameters and the number of image pairs used for training are detailed in Table 5.23.

Parameter	**Training 1**	**Training 2**
Image Resolution	600×960	600×960
Translation Factor	(0, 1)	(0, 0.3)
Max Rotation Angle	45°	90°
Scale Factor	-	(0.7, 1.3)
Training Samples	10800	20000
Validation Samples	1200	2000
Test Samples	1000	3000

Table 5.23: Training Configuration and Image Resolution Parameters

For the homography training, the following setup was adopted:

Hyperparameter	**Value**
Number of epochs	10
Optimizer	AdamW
Learning rate	10^{-4}
Batch size	8 (gradient accumulation steps = 8)
Loss balancing factors	$\alpha = \beta = 0.25$, $\gamma = 0.5$
ground truth balancing factor	0.5

Table 5.24: Training Hyperparameters

The ground-truth balancing factor is a hyperparameter that balances the weight of the positive and negative matches. At the end of this training phase, the student model is evaluated on the test set, and the obtained metrics are summarized in Table 5.25. The results indicate that the student model demonstrates a strong ability to predict correct correspondences when compared to both the ground truth and the predictions made by LightGlue. Additionally, LightGlue itself proves to be highly effective in retrieving correct matches when evaluated against the ground truth correspondences.

Metric	Student vs GT	Student vs Teacher	Teacher vs GT
Recall	0.85	0.81	0.95
Precision	0.78	0.78	0.91
Accuracy	0.89	0.88	0.97
Average Precision	0.66	0.64	0.84

Table 5.25: Metrics after the first homography training stage

In the second phase of training, we utilize the dataset with modified transformation parameters. The training setup remains unchanged, and the model undergoes an additional 20 epochs to adapt to more complex transformations. After this, the student model is evaluated on the test set. The results are presented in Table 5.26. As shown, the student model continues to demonstrate the ability to retrieve correct matches.

Metric	Student vs GT	Student vs Teacher	Teacher vs GT
Recall	0.87	0.82	0.92
Precision	0.82	0.80	0.88
Accuracy	0.91	0.89	0.96
Average Precision	0.71	0.67	0.82

Table 5.26: Metrics after the second homography training stage

Finetuning

The finetuning involves training the model on image pairs that are not obtained through geometric transformations but instead paired based on their relative camera poses. This approach aims to simulate more realistic conditions. The relative distance and orientation intervals are:

- $t_{dist} = (0,1)$
- $R_{dist} = (0,70)^\circ$

This introduces variability in viewpoint change, from simple change of perspective to more complex scenarios. The training set is composed of 20000 pairs, while the validation and test sets of 2000 and 3000 image pairs, respectively.
The training is made with the following setup

Hyperparameter	Value
Number of epochs	50
Optimizer	AdamW
Learning rate	10^{-4}
Batch size	8 (gradient accumulation steps = 8)
Loss balancing factors	$\alpha = \beta = 0.25$, $\gamma = 0.5$
ground truth balancing factor	0.7

Table 5.27: Training Hyperparameters

In this stage, the supervision is entirely provided by the teacher model. So, we report the metric obtained in the test set compared to the LightGlue predictions only. In these scenarios, as we can see in table 5.28, the student model has more difficulty copying the teacher's output.

Metric	Value
Match Recall	0.48
Match Precision	0.36
Accuracy	0.74
Average Precision	0.19

Table 5.28: Evaluation Metrics Compared to Teacher Model

5.2.2 Student vs Teacher model

After training the student model, we evaluate its performance using the same tests conducted for both classical and deep learning-based methods. The analysis focuses on comparing the student model with its teacher model, LightGlue, identifying cases where their performance is similar and where there are gaps.
Both models use SuperPoint as the feature extractor, meaning they receive the same input features and descriptor vectors. The student model is trained with a maximum of 128 interest points to simplify and accelerate training, but it can process a higher and variable number of key points. To ensure a fair comparison, we report results using a maximum (but not fixed) limit of 1024 key points, consistent with previous tests. LightGlue's results are obtained under the same conditions.
By evaluating the student model on homography estimation and relative camera pose estimation, we assess its performance against LightGlue at a global level. Instead of directly analyzing pixel-level matching correspondences, we focus on indirect metrics related to final transformation estimations, providing a more

comprehensive performance comparison.

Homography estimation

We evaluated homography estimation performance on the SPEED and SPEED+ datasets to assess the student model's effectiveness compared to its teacher model, LightGlue. These evaluations provide insight into the model's robustness across different domains and its ability to generalize under varying conditions.
Table 5.29 presents the homography estimation results on the SPEED synthetic dataset.

Method	**AUC**			**P (%)**	**Mean error**	**Matches**
	@1px	**@3px**	**@5px**			
Student	24.2	40.3	46.9	89	80.4	239.4
LightGlue	25.9	44.5	54.0	92	31.2	276.7

Table 5.29: Homography estimation results on the SPEED dataset.

The results for the SPEED+ dataset are shown in table 5.30 and in table 5.31 for the *synthetic*, *lightbox* and *sunlamp* domains, respectively.
Across all datasets, LightGlue consistently achieves higher AUC scores, better precision, and a lower mean reprojection error. However, the student model achieves good results, especially in the synthetic images. When evaluating performance on real image datasets from SPEED+, such as Lightbox and Sunlamp, the effect of domain shift becomes evident, especially in the Sunlamp scenario where the gap between the two models is larger. In figure 5.17, there are images from the different datasets to have a visualization of the results of the student model.

Method	**AUC**			**P (%)**	**Mean error**	**Matches**
	@1px	**@3px**	**@5px**			
Student	29.2	49.1	60.6	89	24.82	225.5
LightGlue	32.4	56.0	67.1	96	8.0	377.5

Table 5.30: Homography estimation results on the SPEED+ synthetic dataset.

Method	Lightbox				Sunlamp			
	AUC@5px	**P (%)**	**Mean error**	**Matches**	**AUC@5px**	**P (%)**	**Mean error**	**Matches**
Student	49.5	85	26.6	200.6	72.2	90	14.2	225.5
LightGlue	52.1	93	32.4	230.7	88.0	96	8.4	377.5

Table 5.31: Homography estimation results on the SPEED+ dataset (Lightbox and Sunlamp domains).

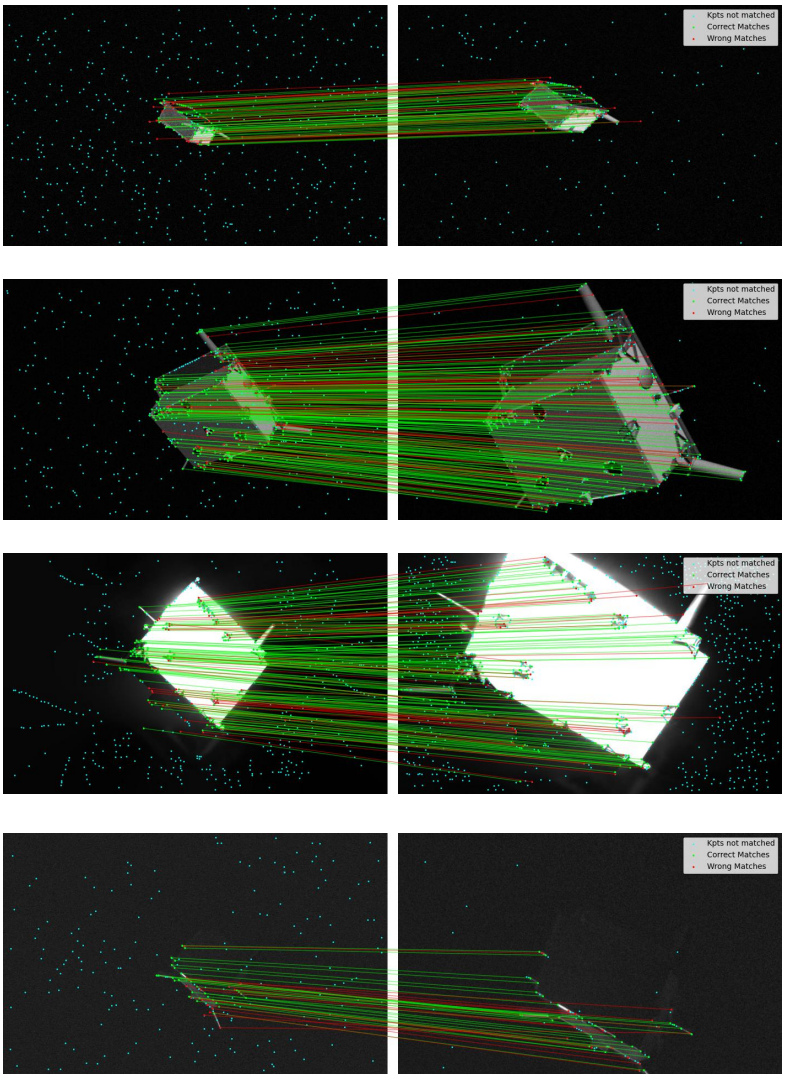

Figure 5.17: Homography estimation results using the student model across different datasets. In order: SPEED, SPEED+ synthetic, Sunlamp and Lightbox domains

Relative camera pose estimation

To evaluate the the student model on the relative camera pose estimation task, we tested it on the same image pairs used in the previous experiments.
The first evaluation scenario focuses on the SPEED dataset, specifically on the *easy* case without background elements. From results in Table 5.32, It is possible to observe how the results of the student model align with the teacher model, particularly in terms of the number of predicted matches and the average errors, while there is a larger gap regarding the accuracy of the estimation.

Method	AUC			Avg Matches	Avg T. Error (°)	Avg R. Error (°)
	@5°	@10°	@20°			
Student	6.91	14.99	27.23	58.5	26.6	25.9
LightGlue	10.50	18.59	30.02	59.8	26.3	25.7

Table 5.32: Relative camera pose estimation results for the student and LightGlue on the SPEED dataset.

The synthetic domain of the SPEED+ dataset confirms this, Table 5.33. This set offers more favourable conditions, such as higher spacecraft visibility, and even the student model benefits from achieving better performance, especially when compared with the results of other methods in previous tests. In this situation, the reduced accuracy in pose estimation is influenced by the number of matches, which is lower than that of the teacher. Even in the real domains, Table 5.34, there is a difference compared to LightGlue. Moreover, the Student model, having been trained solely on synthetic images, faces greater challenges with images under different conditions. Figure 5.18 contains sample pairs from SPEED+ with the student model output.

Method	AUC			Avg Matches	Avg T. Error (°)	Avg R. Error (°)
	@5°	@10°	@20°			
Student	17.23	30.96	46.60	82.3	16.4	11.7
LightGlue	20.12	34.57	50.47	115.7	15.0	12.1

Table 5.33: Relative camera pose estimation results for the student and LightGlue in the synthetic domain of the SPEED+ dataset.

The last scenario is the trajectory set, 5.35. In this case, the differences between the two models are limited, they achieve the same performance, and both exhibit a greater error in the estimation of the correct translation. As mentioned, this is conditioned by the spacecraft's position relative to the camera.

Method	Lightbox				Sunlamp			
	@5°	**@10°**	**@20°**	**Avg Matches**	**@5°**	**@10°**	**@20°**	**Avg Matches**
Student	1.58	2.88	5.67	36.5	0.98	2.85	7.71	11.6
LightGlue	2.11	4.45	8.59	59.8	2.32	5.15	11.29	65.7

Table 5.34: Relative camera pose estimation results for Lightbox and Sunlamp domains of the SPEED+ dataset.

Method	Max Kpts	AUC			Avg Matches	Avg t. Error (°)	Avg R. Error (°)
		@5°	**@10°**	**@20°**			
Student	1024	1.38	3.31	7.12	44.56	59.3	20.1
LightGlue	1024	1.23	3.18	7.15	68.2	58.2	18.8

Table 5.35: Relative camera pose estimation results for the student model in the SPEEDcube dataset.

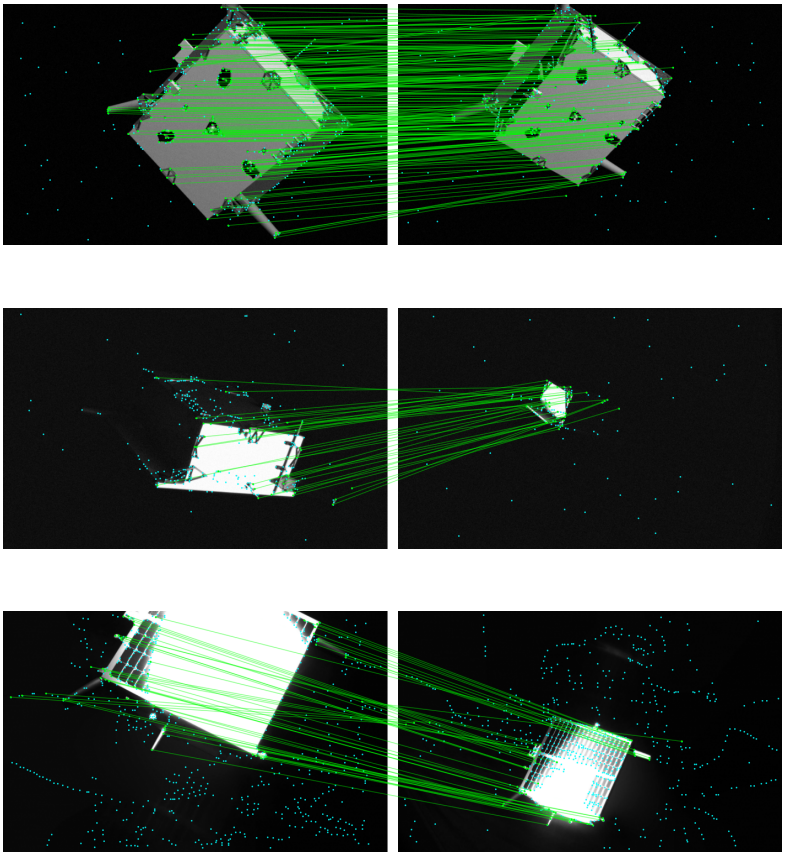

Figure 5.18: Examples of input image pairs from the SPEED+ dataset: synthetic (top), lightbox (middle) and sunlamp(bottom).

5.2.3 Performance and model size comparison

The analysis of the computational performance of AI models is important for space applications due to the specific operational constraints of onboard systems. Compared to ground-based systems, space-grade hardware typically has limited processing power, memory, and energy resources. Additionally, inference times must be evaluated to ensure that models can operate within the timing requirements of real operations, where the process from the image acquisition to the final decision of the system must occur within strict time intervals.
A more complete evaluation would involve testing on specific space-qualified hardware. In this work, we provide a general overview and comparison of computational performance without using specific space-grade boards.

We report the model sizes for LoFTR, SuperPoint, LightGlue, and the Student model. For classical methods, model size is not provided, as these are algorithmic approaches without learnable parameters.
Table 5.36 shows the number of parameters and the model size in megabytes (MB). For both LightGlue and the Student model, the number of layers is set to $L = 9$, which corresponds to LightGlue's standard configuration. To ensure consistency, the same number of layers was adopted for the Student model.

Model	**Parameters (M)**	**Model Size (MB)**
SuperPoint	1.30	4.96
LightGlue	11.85	45.21
Student	11.56	44.12
LoFTR	28.35	108.13
SuperPoint + LightGlue	13.15	50.17
SuperPoint + Student	12.86	49.08

Table 5.36: Comparison of models by parameter count and model size.

Table 5.37 presents the inference times, computed as the average number of image pairs processed per second (pairs/sec). The results are based on inference over 1,000 image pairs with a resolution of 600×960. For SuperPoint combined with LightGlue or the Student model, the reported time includes the whole pipeline (extraction and matching). It's important to note that AI-based models are executed on a CUDA-enabled GPU, whereas classical algorithms are run on the CPU.
Table 5.38 shows the inference speed for classical methods, where the processing time includes both the feature extraction and the K-NN-based feature matching phases.

Model	**Average Inference Speed (pairs/sec)**
SuperPoint + LightGlue	30.43
SuperPoint + Student	7.10
LoFTR	10.23

Table 5.37: Average inference time of AI-based methods, measured in image pairs per second.

Algorithm	**Average Inference Speed (pairs/sec)**
SIFT	8.85
ORB	20.77
AKAZE	11.45

Table 5.38: Average inference speed of classical methods, measured in image pairs per second.

Chapter 6

Discussions and Conclusions

6.1 Discussions

The indirect metrics considered in this study, homography estimation and relative camera pose estimation, serve as proxies for assessing the accuracy of FDM methods. Due to the lack of datasets that provide direct ground truth correspondences, it becomes necessary to rely on these metrics to evaluate models. It is important to note, however, that these metrics can be influenced by the nature of the image transformations and by instabilities or inaccuracies in the estimation process itself. Despite these limitations, they offer a valuable, systematic, and quantitative way to compare the effectiveness of different FDM approaches.

Homography Estimation

Starting with the homography estimation task, some interesting insights emerge. In this setup, the image pairs differ only in terms of geometric transformation, while lighting conditions and background remain constant. This simplifies the FDM process by analysing only the geometric variation. In the SPEED dataset, LoFTR achieves the best performance in estimating the homography, with an *AUC@5 px* of **89.1%**. LightGlue and the Student model, on the other hand, show slightly lower performance, comparable to classical methods, despite their high matching precision scores (**92%** and **89%**, respectively).
This can be likely due to the fundamental differences in their architectures. LoFTR directly predicts dense correspondences between the two images, leading to a significantly higher average number of matches (e.g., 968 in SPEED). In contrast, LightGlue and the Student model focus on sparse keypoint matching, yielding a generally lower number of correspondences (e.g. 276.7 and 239.4, respectively), similar to traditional methods. This directly affects the accuracy of the estimated

homography since even non-relevant matches (e.g., from background or noise), if correctly matched, can improve the precision of the estimated transformation.
The same observation holds for the SPEED+ dataset across its different subsets. When transitioning from the synthetic to real images, *Lightbox* and *Sunlamp*, traditional methods show limited robustness, as illustrated in Figure 5.4. Their performance drops significantly under extreme lighting conditions, such as poor illumination or overexposure caused by direct sunlight. In contrast, AI-based models tend to extract and match a higher number of interest points. For instance, in the *Sunlamp* set, LightGlue reaches an *AUC@5px* of **88%**, compared to **43.5%** for SIFT. It also predicts a considerably higher average number of matches, **377.5** versus **93.5**. The Student model follows closely with an *AUC@5px* of **75.5%** and an average of **225.5** matches.
This difference highlights an important trade-off. On one hand, traditional methods may fail to find a sufficient number of correspondences in visually challenging scenarios, leading to poor estimation accuracy. On the other, AI models might be overly sensitive to noise or irrelevant features. Therefore, it may be necessary to fine-tune internal hyperparameters to filter out non-relevant matches and improve robustness between different conditions.

Relative pose estimation

The second task expands the analysis by considering other scenarios and conditions. In these tests, it is important to note that the image pairs may differ not only in terms of perspective due to varying viewpoints between the cameras but also in lighting conditions and background content. Unlike the previous task, in this case, it becomes important for the key points to be extracted and matched from the body of the target object rather than the surrounding environment to have an accurate geometry estimation.
In the SPEED dataset, we evaluated the methods across increasing difficulty levels, based on the extent of viewpoint differences between the frames. As expected, the accuracy of all methods decreases with higher difficulty and the average translation and rotation errors increase. Overall, the most effective models remain LightGlue and the Student. For instance, in the *easy* scenario, they achieve angular AUC scores that are approximately **3–5**$\times$ higher than those of traditional algorithms and even outperform LoFTR. Although LoFTR produces a much higher number of correspondences, these are not necessarily relevant and can even hinder an accurate estimation.
Another key factor is the presence of background. In the previous results, the background is absent, so images with varying surrounding environments are considered to assess its impact. While LightGlue remains the best model, its results degrade significantly in these settings, largely because many of the correspondences come

from the background rather than the spacecraft. For example, LightGlue has a drop of $-$**14.8%** in *AUC@20°* when the background is introduced. Additionally, the average rotation error is much higher in traditional methods, with a difference of **+56.3°** between SIFT and LightGlue. This is also evident from the visualizations, where classical methods generate many false positive matches in the background. The Student model performs worse in this case compared to the teacher, which is expected since it is trained only on images without background. Consequently, it predicts a much lower average number of matches, approximately 11.2, resulting in less reliable estimations.
In the SPEED+ dataset, we analyzed the domain gap. As anticipated, all methods perform best in the synthetic domain, where lighting is consistent, backgrounds are removed, and the spacecraft appears closer to the camera. In this context, both LightGlue and the Student model show strong performance. Specifically, LightGlue achieves an *AUC@20°* gain of approximately **+22%** over LoFTR and **+34.6%** over SIFT, while the Student model improves by **+14.3%** and **+26.5%**, respectively. In real image domains, performance drops for all methods, but DL models maintain slightly greater robustness, with lower average errors compared to classical algorithms. For example, the average error gap between LightGlue and SIFT reaches **+54.4°** in the Lightbox domain and **67.5°** in Sunlamp.
Finally, the results obtained on the trajectory test set are analyzed. Performance across all models is generally more uniform in this scenario due to the inherent difficulty of accurate estimation, as explained earlier. Nonetheless, LoFTR achieves translation and rotation errors of **38.3°** and **11.6°**, which are significantly lower than those of SIFT, **72.7°** and **38.9°**, respectively. LightGlue and the Student model produce very similar results, with only minimal differences, **1.1°** in translation error and **1.3°** in rotation error, in favour of the teacher.

To summarize, DL models have significant advantages that stem from their ability to capture global contextual information in images. Unlike the algorithmic methods, which rely on local image regions, DL-based have a broader understanding of the scene. This enables them to match key points not only based on descriptor similarity but also by considering their spatial relationships and the overall image structure. As a result, they produce more accurate correspondences and a higher number of inliers, leading to improved geometric estimations. Both DL and classical methods tend to perform better on synthetic images, which offer more favourable visual characteristics. In contrast, real domains can introduce a domain gap due to extreme lighting conditions. These factors compromise robustness, increase noise, and often lead to the extraction of irrelevant features and incorrect matches. To reduce this gap, DL models can be trained or fine-tuned on datasets that more accurately reflect the appearance and variability of real space imagery.

Knowledge Distillation

KD proved to be an effective strategy for training the Student model in these experiments. Despite the lack of annotated data needed for supervised training, the Student benefits from the supervision provided by the Teacher model. Despite the use of different operations within the architecture, the results obtained are close to those achieved by LightGlue and the differences in many tests are reduced. The Student typically predicts fewer correspondences than the Teacher, which can negatively affect performance. This aspect can be improved through adjustments in the training process. For instance, our model is trained using a maximum of 128 key points to speed up the training, whereas evaluation was performed with up to 1024 detectable key points. This mismatch may influence the number of matches and, consequently, the estimation accuracy. Also, there is room for improvement in the distillation strategy, such as trying additional loss functions and fine-tuning training hyperparameters.

6.2 Conclusions

In conclusion, this preliminary analysis demonstrates how state-of-the-art deep learning models can perform better than classical algorithms in the FDM task and how their capabilities, acquired in general contexts, transfer to space environments characterized by challenging image conditions.
Future works can explore the integration of FDM within visual navigation space systems and the deployment on specific space-grade hardware. Additional optimization techniques can be explored to refine the model design, reduce its size, and lower the number of operations, making it more suitable for systems with resource constraints.

Bibliography

[1] Mark Kisantal, Siddharth Sharma, Tae Ha Park, Dario Izzo, Marcus Märtens, and Simone D'Amico. «Satellite Pose Estimation Challenge: Dataset, Competition Design and Results». In: *ArXiv preprint* (2019). arXiv: 1911.02050. URL: https://arxiv.org/abs/1911.02050 (cit. on pp. 2, 39, 40).

[2] Tae Ha Park, Marcus Martens, Gurvan Lecuyer, Dario Izzo, and Simone D'Amico. «SPEED+: Next-Generation Dataset for Spacecraft Pose Estimation across Domain Gap». In: *2022 IEEE Aerospace Conference (AERO)*. IEEE, Mar. 2022, pp. 1–15. DOI: 10.1109/aero53065.2022.9843439. URL: http://dx.doi.org/10.1109/AERO53065.2022.9843439 (cit. on pp. 2, 41–43).

[3] Zahra Ahmed, Tae Ha Park, Aniket Bhattacharjee, Roxana Razel-Rezai, Rhiannon Graves, Oskari Saarela, Ryo Teramoto, Kalyan Vemulapalli, and Simone D'Amico. «SPEED-UE-Cube: A Machine Learning Dataset for Autonomous, Vision-Based Spacecraft Navigation». In: *46th Rocky Mountain AAS Guidance, Navigation and Control Conference*. Breckenridge, Colorado, Feb. 2024 (cit. on pp. 2, 43).

[4] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. *HPatches: A benchmark and evaluation of handcrafted and learned local descriptors*. 2017. arXiv: 1704.05939 [cs.CV]. URL: https://arxiv.org/abs/1704.05939 (cit. on pp. 2, 45).

[5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. *ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes*. 2017. arXiv: 1702.04405 [cs.CV]. URL: https://arxiv.org/abs/1702.04405 (cit. on pp. 2, 46).

[6] W.M. Owen. *Spacecraft Optical Navigation*. JPL Deep-Space Communications and Navigation Series. Wiley, 2024. ISBN: 9781119904458. URL: https://books.google.it/books?id=vL0pEQAAQBAJ (cit. on p. 4).

[7] J. M. Rebordão. «Space optical navigation techniques: an overview». In: *8th Iberoamerican Optics Meeting and 11th Latin American Meeting on Optics, Lasers, and Applications*. Ed. by Manuel Filipe P. C. Martins Costa. Vol. 8785.

International Society for Optics and Photonics. SPIE, 2013, 87850J. DOI: 10.1117/12.2026063. URL: https://doi.org/10.1117/12.2026063 (cit. on p. 4).

[8] V.V. Svotina and M.V. Cherkasova. «Space debris removal – Review of technologies and techniques. Flexible or virtual connection between space debris and service spacecraft». In: *Acta Astronautica* 204 (2023), pp. 840–853 (cit. on p. 4).

[9] Michael Luu and Daniel Hastings. «On-Orbit Servicing System Architectures for Proliferated Low-Earth-Orbit Constellations». In: *Journal of Spacecraft and Rockets* (Aug. 2022). DOI: 10.2514/1.A35393 (cit. on p. 4).

[10] Michael E. Polites. *NASA's Attitude Control Systems for Spacecraft: Advances and Applications*. Tech. rep. NASA/TP-1998-208528. Marshall Space Flight Center, Huntsville, AL: NASA Technical Publication, July 1998 (cit. on p. 5).

[11] Anthea Comellini, Emmanuel Zenou, Christine Espinosa, and Vincent Dubanchet. «Vision-based navigation for autonomous space rendezvous with non-cooperative targets». In: *2020 11th International Conference on Information, Intelligence, Systems and Applications (IISA*. 2020, pp. 1–8. DOI: 10.1109/IISA50023.2020.9284383 (cit. on p. 5).

[12] D. Nister, O. Naroditsky, and J. Bergen. «Visual odometry». In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. 2004, pp. I–I. DOI: 10.1109/CVPR.2004.1315094 (cit. on p. 6).

[13] Wikipedia contributors. *Visual odometry — Wikipedia, The Free Encyclopedia*. Accessed: 2025-03-23. 2024. URL: https://en.wikipedia.org/wiki/Visual_odometry (cit. on p. 6).

[14] Yalan Chen, Yimin Zhou, Qin Lv, and Kranthi Kumar Deveerasetty. «A Review of V-SLAM». In: *2018 IEEE International Conference on Information and Automation (ICIA)*. 2018, pp. 603–608. DOI: 10.1109/ICInfA.2018.8812387 (cit. on p. 6).

[15] Wikipedia contributors. *Bundle adjustment — Wikipedia, The Free Encyclopedia*. Accessed: 2025-03-23. 2024. URL: https://en.wikipedia.org/wiki/Bundle_adjustment (cit. on p. 6).

[16] Kentaro Uno, Takehiro Matsuoka, Akiyoshi Uchida, and Kazuya Yoshida. *Structure from Motion-based Motion Estimation and 3D Reconstruction of Unknown Shaped Space Debris*. 2024. arXiv: 2408.01035 [cs.RO]. URL: https://arxiv.org/abs/2408.01035 (cit. on p. 7).

[17] I.H. Sarker. «Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions». In: *SN Computer Science* 2 (2021), p. 420. DOI: 10.1007/s42979-021-00815-1 (cit. on p. 9).

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep learning». In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539 (cit. on p. 9).

[19] Kit Yan Chan, Bilal Abu-Salih, Raneem Qaddoura, Ala' M. Al-Zoubi, Vasile Palade, Duc-Son Pham, Javier Del Ser, and Khan Muhammad. «Deep neural networks in the cloud: Review, applications, challenges and research directions». In: *Neurocomputing* 545 (2023), p. 126327. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2023.126327. URL: https://www.sciencedirect.com/science/article/pii/S0925231223004502 (cit. on p. 9).

[20] George Cybenko. «Approximation by superpositions of a sigmoidal function». In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314 (cit. on p. 10).

[21] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. «Learning representations by back-propagating errors». In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0 (cit. on p. 10).

[22] I. H. Sarker. «Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions». In: *SN Computer Science* 2 (2021), p. 420. DOI: 10.1007/s42979-021-00815-1 (cit. on p. 10).

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 10, 12).

[24] Wikipedia contributors. *Convolutional layer — Wikipedia, The Free Encyclopedia*. [Online; accessed February 13, 2024]. 2024. URL: https://en.wikipedia.org/wiki/Convolutional_layer (cit. on p. 11).

[25] L. Alzubaidi et al. «Review of deep learning: concepts, CNN architectures, challenges, applications, future directions». In: *Journal of Big Data* 8.1 (2021), p. 53. DOI: 10.1186/s40537-021-00444-8 (cit. on p. 12).

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is All you Need». In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on pp. 13, 14).

[27] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. *A Survey of Transformers*. 2021. arXiv: 2106.04554 [cs.LG]. URL: https://arxiv.org/abs/2106.04554 (cit. on p. 13).

[28] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929 (cit. on p. 15).

[29] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. *Pruning and Quantization for Deep Neural Network Acceleration: A Survey*. 2021. arXiv: 2101.09671 [cs.CV]. URL: https://arxiv.org/abs/2101.09671 (cit. on p. 16).

[30] Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. «A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification». In: *ACM Transactions on Intelligent Systems and Technology* 14.6 (Nov. 2023), pp. 1–50. ISSN: 2157-6912. DOI: 10.1145/3623402. URL: http://dx.doi.org/10.1145/3623402 (cit. on p. 16).

[31] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. «Efficient Processing of Deep Neural Networks: A Tutorial and Survey». In: *Proceedings of the IEEE* 105.12 (2017), pp. 2295–2329. DOI: 10.1109/JPROC.2017.2761740 (cit. on p. 16).

[32] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. «Knowledge Distillation: A Survey». In: *International Journal of Computer Vision* 129.6 (Mar. 2021), pp. 1789–1819. ISSN: 1573-1405. DOI: 10.1007/s11263-021-01453-z. URL: http://dx.doi.org/10.1007/s11263-021-01453-z (cit. on pp. 16–19).

[33] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. «Distilling the Knowledge in a Neural Network». In: *arXiv preprint arXiv:1503.02531* (2015). URL: https://arxiv.org/abs/1503.02531 (cit. on p. 16).

[34] Gousia Habib, Tausifa jan Saleem, Sheikh Musa Kaleem, Tufail Rouf, and Brejesh Lall. *A Comprehensive Review of Knowledge Distillation in Computer Vision*. 2024. arXiv: 2404.00936 [cs.CV]. URL: https://arxiv.org/abs/2404.00936 (cit. on p. 16).

[35] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. *Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation*. 2019. arXiv: 1905.08094 [cs.LG]. URL: https://arxiv.org/abs/1905.08094 (cit. on p. 19).

[36] Shibiao Xu, Shunpeng Chen, Rongtao Xu, Changwei Wang, Peng Lu, and Li Guo. «Local feature matching using deep learning: A survey». In: *Information Fusion* 107 (July 2024), p. 102344. ISSN: 1566-2535. DOI: 10.1016/j.inffus.2024.102344. URL: http://dx.doi.org/10.1016/j.inffus.2024.102344 (cit. on pp. 20, 24–26, 35, 37).

[37] C. Harris and M. Stephens. «A Combined Corner and Edge Detector». In: *Alvey Vision Conference*. Vol. 15. Manchester, UK, 1988, pp. 10–5244 (cit. on p. 21).

[38] David G. Lowe. «Object Recognition from Local Scale-Invariant Features». In: *Proceedings of the Seventh IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 1999, pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410 (cit. on p. 21).

[39] David G. Lowe. «Distinctive Image Features from Scale-Invariant Keypoints». In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110. DOI: 10.1023/B:VISI.0000029664.99615.94 (cit. on pp. 21, 24).

[40] T. Lindeberg. «Image Matching Using Generalized Scale-Space Interest Points». In: *Journal of Mathematical Imaging and Vision* 52.1 (2015), pp. 3–36. DOI: 10.1007/s10851-014-0541-0 (cit. on p. 21).

[41] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. «ORB: An efficient alternative to SIFT or SURF». In: *2011 International Conference on Computer Vision*. 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544 (cit. on p. 22).

[42] E. Rosten and T. Drummond. «Machine Learning for High-Speed Corner Detection». In: *European Conference on Computer Vision (ECCV)*. Graz, Austria, 2006, pp. 430–443. DOI: 10.1007/11744023_34 (cit. on pp. 22, 24).

[43] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. «BRIEF: Binary Robust Independent Elementary Features». In: *European Conference on Computer Vision (ECCV)*. Springer, Berlin, Heidelberg, 2010, pp. 778–792. DOI: 10.1007/978-3-642-15561-1_56 (cit. on p. 22).

[44] C. Harris and M. Stephens. «A Combined Corner and Edge Detector». In: *Alvey Vision Conference*. 1988, pp. 147–151 (cit. on pp. 22, 24).

[45] Pablo F. Alcantarilla, Jesús Nuevo, and Adrien Bartoli. «Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces». In: *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2013, pp. 1–11. DOI: 10.5244/C.27.13 (cit. on p. 23).

[46] Pablo F. Alcantarilla, Adrien Bartoli, and Andrew J. Davison. «KAZE Features». In: *European Conference on Computer Vision (ECCV)*. Springer, 2012, pp. 214–227. DOI: 10.1007/978-3-642-33783-3_16 (cit. on p. 23).

[47] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. «SURF: Speeded Up Robust Features». In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2006, pp. 404–417 (cit. on p. 24).

[48] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. *SuperPoint: Self-Supervised Interest Point Detection and Description*. 2018. arXiv: 1712.07629 [cs.CV]. URL: https://arxiv.org/abs/1712.07629 (cit. on pp. 25, 26, 51).

[49] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. *SuperGlue: Learning Feature Matching with Graph Neural Networks*. 2020. arXiv: 1911.11763 [cs.CV]. URL: https://arxiv.org/abs/1911.11763 (cit. on pp. 25, 28, 80).

[50] Rémi* Pautrat, Iago* Suárez, Yifan Yu, Marc Pollefeys, and Viktor Larsson. «GlueStick: Robust Image Matching by Sticking Points and Lines Together». In: *International Conference on Computer Vision (ICCV)*. 2023 (cit. on p. 25).

[51] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. *LightGlue: Local Feature Matching at Light Speed*. 2023. arXiv: 2306.13643 [cs.CV]. URL: https://arxiv.org/abs/2306.13643 (cit. on pp. 25, 28, 58, 69).

[52] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. *LoFTR: Detector-Free Local Feature Matching with Transformers*. 2021. arXiv: 2104.00680 [cs.CV]. URL: https://arxiv.org/abs/2104.00680 (cit. on pp. 26, 30, 31, 80).

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762 (cit. on p. 28).

[54] Michał J. Tyszkiewicz, Pascal Fua, and Eduard Trulls. *DISK: Learning local features with policy gradient*. 2020. arXiv: 2006.13566 [cs.CV]. URL: https://arxiv.org/abs/2006.13566 (cit. on p. 28).

[55] Zhengqi Li and Noah Snavely. «MegaDepth: Learning Single-View Depth Prediction from Internet Photos». In: *Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 30).

[56] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. «Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention». In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020 (cit. on p. 31).

[57] Yinhui Luo, Xingyi Wang, Yanhao Liao, Qiang Fu, Chang Shu, Yuezhou Wu, and Yuanqing He. «A Review of Homography Estimation: Advances and Challenges». In: *Electronics* 12.24 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12244977. URL: https://www.mdpi.com/2079-9292/12/24/4977 (cit. on p. 33).

[58] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. New York, NY, USA: Cambridge University Press, 2003 (cit. on pp. 33, 35, 36).

[59] Kenneth Levenberg. «A Method for the Solution of Certain Non-Linear Problems in Least Squares». In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168. DOI: 10.1090/qam/10666 (cit. on p. 35).

[60] Silvio Savarese and Juan Carlos Niebles. *CS231A: Computer Vision - From 3D Reconstruction to Recognition, Course Notes*. https://web.stanford.edu/class/cs231a/course_notes.html. Accessed: March 23, 2025. 2024 (cit. on p. 36).

[61] David Nistér. «An Efficient Solution to the Five-Point Relative Pose Problem». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 26.6 (2004), pp. 756–770. DOI: 10.1109/TPAMI.2004.17 (cit. on p. 37).

[62] R.I. Hartley. «In defense of the eight-point algorithm». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.6 (1997), pp. 580–593. DOI: 10.1109/34.601246 (cit. on p. 37).

[63] Martin A. Fischler and Robert C. Bolles. «Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography». In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692 (cit. on p. 37).

[64] Mercedes Garcia-Salguero, Jesus Briales, and Javier Gonzalez-Jimenez. *Certifiable Relative Pose Estimation*. 2021. arXiv: 2003.13732 [cs.CV]. URL: https://arxiv.org/abs/2003.13732 (cit. on p. 38).

[65] NanoSats Database. *TANGO - Satellite Information*. Accessed: 2024-02-13. 2024. URL: https://www.nanosats.eu/sat/tango (cit. on p. 39).

[66] Wikipedia contributors. *PRISMA (satellite project) — Wikipedia, The Free Encyclopedia*. Accessed: 2024-02-13. 2024. URL: https://en.wikipedia.org/wiki/Prisma_(satellite_project) (cit. on p. 39).

[67] C. Chasset, R. Noteborn, P. Bodin, et al. «3-Axis magnetic control: flight results of the TANGO satellite in the PRISMA mission». In: *CEAS Space Journal* 5 (2013), pp. 1–17. DOI: 10.1007/s12567-013-0034-9. URL: https://doi.org/10.1007/s12567-013-0034-9 (cit. on p. 39).

[68] Stanford Space Rendezvous Laboratory. *Stanford SLAB — Space Rendezvous Laboratory*. https://slab.stanford.edu/. Accessed: March 26, 2025. 2024 (cit. on p. 39).

[69] Epic Games. *Unreal Engine 5*. https://www.unrealengine.com/en-US/unreal-engine-5. Accessed: March 26, 2025. 2024 (cit. on p. 43).

[70] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. *A ConvNet for the 2020s*. 2022. arXiv: 2201.03545 [cs.CV]. URL: https://arxiv.org/abs/2201.03545 (cit. on p. 59).

[71] Bo Jiang, Pengfei Sun, Jin Tang, and Bin Luo. *GLMNet: Graph Learning-Matching Networks for Feature Matching*. 2019. arXiv: 1911.07681 [cs.CV]. URL: https://arxiv.org/abs/1911.07681 (cit. on p. 60).

[72] G. Bradski. «The OpenCV Library». In: *Dr. Dobb's Journal of Software Tools* (2000) (cit. on p. 67).

[73] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. «Kornia: an Open Source Differentiable Computer Vision Library for PyTorch». In: *Winter Conference on Applications of Computer Vision*. 2020. URL: https://arxiv.org/pdf/1910.02190.pdf (cit. on p. 67).

[74] OpenCV Contributors. *findHomography - OpenCV Documentation*. Accessed: 2025-03-20. 2025. URL: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga8c01c30f50cfa4b07b98d51b37b80b49 (cit. on p. 68).

[75] OpenCV Contributors. *findEssentialMat - OpenCV Documentation*. Accessed: 2025-03-20. 2025. URL: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga465b6a285c40ee047921916127a11417 (cit. on p. 79).

[76] OpenCV Contributors. *recoverPose - OpenCV Documentation*. Accessed: 2025-03-20. 2025. URL: https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga84f5a8093378580b9f52a64bcea0a437 (cit. on p. 79).

[77] R. E. Kalman. «A New Approach to Linear Filtering and Prediction Problems». In: *Transactions of the ASME–Journal of Basic Engineering* 82.1 (1960), pp. 35–45 (cit. on p. 94).

[78] Cory T. Fraser and Steve Ulrich. «Adaptive extended Kalman filtering strategies for spacecraft formation relative navigation». In: *Acta Astronautica* 178 (2021), pp. 700–721. ISSN: 0094-5765. DOI: https://doi.org/10.1016/j.actaastro.2020.10.016. URL: https://www.sciencedirect.com/science/article/pii/S009457652030610X (cit. on p. 94).