Final Project Report (Group-4)

Ayush Kumar Singh
ai20btech11028@iith.ac.in

Digjoy Nandi
ai20btech11007@iith.ac.in

Vojeswitha Gopireddy
ai20btech11024@iith.ac.in

Omkaradithya R Pujari
ai20btech11017@iith.ac.in

Abstract

In our previous report, we discussed different stages of image matching in a learning-based approach. In our final report, we combine these stages to create an end-to-end pipeline for relative pose estimation using fundamental matrix calculation. For testing our pipeline, we used the Image Matching Challenge 2022 test dataset. We started with different state-of-the-art models as our baseline, such as SuperGLUE, DKM, LoFTR, etc. We then tried out their ensemble along with several test time augmentations. We also discussed a commonly used clustering algorithm(DBSCAN) and used it to find common scenes in both images in our final pipeline. We tested all our implementation through Kaggle submissions, and our best submission got placed in the top 50 in the final leaderboard.

1. Introduction

Image matching is a technique used in computer vision to identify and compare the similarity between two or more images. It involves locating the corresponding points or features in the images to be matched and then comparing them using various algorithms.

In our previous report, we discussed different deep-learning and classical learning-based methods for image matching. In this report, we have discussed different state-of-the-art models that significantly improve over traditional models.

Most existing methods use a keypoint detector to find the interest points. However, these models failed to extract an adequate number of interest points between images for various reasons such as poor texture, repetitive patterns, illumination variations, etc. The state-of-the-art model, such as LoFTR, tackles this issue by introducing a detector-free approach. It uses a transformer to find the positional encoding of coarse-level and fine-level features extracted by standard CNN. It then compares these features to find the matching points.

In this report, we created an end-to-end pipeline to estimate the relative pose between two images. We tested our implementation on a publicly available dataset on Image Matching Challenge 2022. We tried out a different ensemble of models and reported its performance on the test dataset. Our final pipeline consists of two stages where we identify matching points and extract the majority cluster. We again perform image matching on these clusters and concatenate it with other matching points obtained from augmented input to get the final set of matching points.

2. Literature Review:

2.1. Super Glue: Feature matching with Graph Neural Networks

SuperGlue estimates the assignments by solving a differential optimal transport problem whose cost function is predicted by a Graph neural network. It applies self and cross-attention to leverage spatial relations and the appearance of the key points. SuperGlue comprises two major components *the attentional graph neural network* and *the optimal matching layer*.

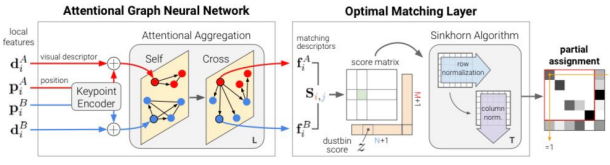

Figure 1. SuperGlue Architecture

2.1.1 Attentional Graph Neural Network

Besides the position and visual appearance, considering the spatial and visual relationship of a key point with other

co-visible key points can increase its distinctiveness. Here comes the attention graph neural network into the picture. The Graph Neural Network(GNN) computes the matching descriptors f_i by letting the features extracted initially (from descriptors like SIFT) communicate with each other through layers of the neural network.

Keypoint Encoder: The intial representation ${}^{(0)}x_i$ for keypoint is combination of the location p_i and visual descriptor d_i of a keypoint i. This representation enables GNN to reason about position and appearance jointly with attention.

$$ {}^{(0)}\mathbf{x}_i = d_i + \text{MLP}_{\text{enc}}(p_i) \tag{1} $$

Multiplex Graph Neural Network: The graph nodes are the key points of the images to be matched. This undirected multiplex graph has two types of edges, Intra-image edges or *self* edges $\mathcal{E}_{\text{self}}$ and Inter-image edges or *cross* edges $\mathcal{E}_{\text{cross}}$. The GNN resulting from the above graph computes the updated representation of the key points by aggregating messages propagated across the edges.

$$ {}^{(l+1)}\mathbf{x}_i^A = {}^{(l)}\mathbf{x}_i^A + \text{MLP}([{}^{(l)}\mathbf{x}_i^A || \text{m}_{\mathcal{E}\to i}]) \tag{2} $$

where ${}^{(l)}x_i^A$ denotes the intermediate representation for the key-point i in image A at layer l , $m_{\mathcal{E}\to i}]$) is the message aggregated from all keypoints $\{j : (i,j) \in \mathcal{E}$ and $[.||.]$ denotes concatenation. A fixed number of layers L with different parameters are chained and alternatively aggregated along self and cross edges.

Attenional Aggregation: The attention mechanism computes the aggregated message $m_{\mathcal{E}\to i}$ as weight-average of the values,

$$ m_{\mathcal{E}\to i} = \sum_{j:\{i,j\}\in\mathcal{E}} \alpha_{ij} v_j \tag{3} $$

Akin to database retrieval, the query q_i retrieves the values v_j based on their attributes, the keys k_j whereas $\alpha_{ij} = \text{Softmax}_j(q_i^T k_j)$.

The key, query and value are computed as the linear projects of deep features of the GNN. Considering that query keypoint i is in the image Q and all source keypoints are in image S, $(Q,S) \in \{A,B\}^2$

$$ q_i = W_1^{(l)} \mathbf{x}_i^Q + b_1 \tag{4} $$

$$ \begin{bmatrix} k_j \\ v_j \end{bmatrix} = \begin{bmatrix} W_2 \\ W_3 \end{bmatrix} {}^{(l)}\mathbf{x}_j^S + \begin{bmatrix} b_2 \\ b_3 \end{bmatrix} \tag{5} $$

Each layer l has its projection parameters, learned and shared for all key points. The final matching descriptors are the linear projections for image A,

$$ f_i^A = W \cdot {}^{(L)}\mathbf{x}_i^A + b \tag{6} $$

2.1.2 Optimal Matching Layer

The second component of SuperGlue, the optimal matching layer, produces a partial assignment matrix by computing the score matrix S for all the possible matches and maximising the total score $\sum_{i,j} S_{i,j} P_{i,j}$ under the constraints $\mathbf{P}1_N \leq 1_M$, $\mathbf{P}^T 1_M \leq 1_N$, where $\mathbf{P}$ is partial soft assignment matrix.

The score for a given pair of key points is expressed as the similarity between their matching descriptors.

$$ S_{i,j} = < f_i^A, f_j^B > \tag{7} $$

Occlusion and Visibility To suppress some key points, the unmatched key points are assigned to the dustbin augmented with its set. Superpoints use the dustbins to account for image cells that might not have a detection. Scores $\mathbf{S}$ are augmented to $\bar{\mathbf{S}}$ by appending a new row and column, the point-to-bin and bin-bin scores, filled with a single learnable character z

Let a, and b denote the number of expected matches for keypoint and dustbin in A and B. The augment $\bar{\mathbf{P}}$ has the following constraints,

$$ \bar{\mathbf{P}} 1_{N+1} = a = \begin{bmatrix} 1_{\text{M}}^T & \text{N} \end{bmatrix}^T \tag{8} $$

$$ \bar{\mathbf{P}}^T 1_{M+1} = b = \begin{bmatrix} 1_{\text{N}}^T & \text{N} \end{bmatrix}^T \tag{9} $$

Sinkhorn Algorithm: The solution to the above score optimization problem corresponds to optimal transport between discrete distributions a and b with scores $\bar{\mathbf{S}}$. The desired soft assignment can be efficiently solved with the Sinkhorn Algorithm. After T iterations, we drop the dustbins and recover the partial soft assignments $\mathbf{P}$ from $\bar{\mathbf{P}}$

2.1.3 Loss

In SuperGlue design, the graph neural network and the optimal matching layer are differentiable. This permits to backpropagate from matches to visual descriptors. To maximise the precision and the recall of the matching, we minimize the negative log-likelihood of the assignment $\bar{\mathbf{P}}$

$$ \text{Loss} = -\sum_{(i,j)\in\mathcal{M}} \log \bar{\mathbf{P}}_{i,j} - \sum_{i\in\mathcal{I}} \log \bar{\mathbf{P}}_{i,N+1} - \sum_{j\in\mathcal{I}} \log \bar{\mathbf{P}}_{M+1,j} $$

2.2. LoFTR: Detector-Free Local Feature Matching with Transformers

Given two images to be matched, most existing matching methods consist of three phases: feature detection, feature description, and feature matching. In the detection phase, salient points like corners are first detected as interest points from each image. Local descriptors are then

extracted around neighbourhood regions of these interest points. The feature detection and description phases produce two sets of interest points and their descriptors, the point-to-point correspondences of which are later found by nearest neighbour search or more sophisticated matching algorithms.
However, a feature detector may fail to extract enough interest points that are repeatable between images due to various factors such as poor texture, repetitive patterns, viewpoint change, illumination variation, and motion blur. Hence a large receptive field in the feature extraction network is crucial.

2.2.1 Method:

LoFTR is a novel detector-free approach to local feature matching. It tackles the repeatability issue of feature detectors with a detector-free design. It uses standard convolution layers to extract local features. Convolution neural networks possess the inductive bias of translation equivariance and locality, which are well suited for local feature extraction.
After the local featured extraction, the features are passed through the LoFTR module to extract position and context-dependent features. In the LoFTR module, 2D extensions of standard positioning encoding in Transformers are used. By adding position encoders to the local features, the transform features will become position-dependent, which is crucial to the ability of LoFTR to produce matches in indistinctive regions.
After Encoding, two types of differential matching layers can be applied in LoFTR, either with an optimal transport layer or with a dual softmax operator. The score matrix S between the transformed features is first calculated by

$$S(i,j) = \frac{1}{\tau}\langle F_{tr}^{A}(i), F_{tr}^{B}(j)\rangle$$

where $F_{tr}^{A}(i)$ and $F_{tr}^{B}(j)$ denotes the transformed fearures. When matching with optimal transport, $-S$ can be used as the cost matrix of the partial assignment problem.
Based on the confidence matrix, we select matches with confidence higher than a threshold and further enforce the mutual nearest neighbour criteria, which filters possible outlier course matches.
After establishing coarse matches, these matches are refined to original image resolutions. For every coarse match, $(\bar{i}, \bar{j})$, we first locate its position $(\hat{i}, \hat{j})$ at fine level features map $\hat{F_{tr}^{A}i}$ and $\hat{F_{tr}^{B}j}$ and then crop two sets of a local window of size wxw. It is then again passed to a smaller LoFTR module which transforms it. The centre vectors of both feature maps are then correlated, and a heat map of matching probability is found. A collection of all the matches is then found by calculating the expectation over these probabilities.

The final loss is a combination of both coarse-level loss and fine-level loss:

$$\mathcal{L} = \mathcal{L}_{\rfloor} + \mathcal{L}_{\{}$$

2.3. DBSCAN:

DBSCAN is a popular clustering algorithm that stands for Density-Based Spatial Clustering of Applications with Noise. It is based on the idea that clusters are dense regions of points that are separated by low-density regions. Unlike other clustering algorithms, such as k-means, DBSCAN does not require specifying the number of clusters in advance. Instead, it can discover clusters of arbitrary shapes and sizes and identify outliers as noise points.

DBSCAN works by finding core points, which are points that have at least a minimum number of other points (minPts) within a given radius (eps). These core points form the basis of clusters, and any point that is within the eps radius of a core point is assigned to the same cluster. Points that are not within the eps radius of any core point are considered noise points and are not assigned to any cluster.

- **Directly density-reachable:** An object q is directly density-reachable from object p if q is within the eps-Neighborhood of p and p is a core object.
- **Density-reachable:** An object p is density-reachable from q w.r.t eps and MinPts if there is a chain of objects $p_1, \ldots, p_n$, with$p_1 = q, p_n = p$ such that $p_{i+1}$is directly density-reachable from p_i w.r.t eps and MinPts for all $1 \leq i \leq n$.
- **Density-connectivity:** Object p is density-connected to object q w.r.t eps and MinPts if there is an object r such that both p and q are density-reachable from r w.r.t eps and MinPts.

The algorithm can be summarized as follows:

- Select a point p
- Retrieve all points density-reachable from p wrt eps and MinPts
- If p is a core point, a cluster is formed.
- If p is a border point, no points are density-reachable from p, and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.

DBSCAN is a powerful clustering algorithm that can find clusters of different shapes and sizes in data sets with noise and outliers. However, it has some limitations that must be considered when applying it to real-world problems. Choosing appropriate values for eps and minPts is crucial for obtaining meaningful results with DBSCAN.

2.4. TTA(Test Time Augmentation):

Similar to what Data Augmentation does to the training set, the purpose of Test Time Augmentation is to perform random modifications to the test images. Thus, instead of showing the regular, "clean" images only once to the trained model, we will show the augmented images several times. We will then average the predictions of each corresponding image and take that as our final guess.
In the case of image matching, we can use several test time augmentation to improve our performance, such as rotating, scaling, cropping, warping, etc.

2.5. Ensembling:

Ensemble learning is a general meta-approach to machine learning that seeks better predictive performance by combining the predictions from multiple models. In ensemble models, we combine the outputs of several models to get the final prediction. There can be several ways to combine the output, such as majority voting, weighted combination, etc.
For the purpose of image matching, we have tried ensembling by concatenating the match points produced by different models.

2.6. Epipolar Geometry:

The epipolar geometry between two views, is essentially the geometry of the intersection of the image planes with the pencil of planes having the baseline as the axis (the baseline is the line joining the camera centres). This geometry is usually motivated by considering the search for corresponding points in stereo matching, and we will start from that objective here.

- The epipole is the point of intersection of the line joining the camera centres (the baseline) with the image plane. Equivalently, the epipole is the image in one view of the camera centre of the other view. It is also the vanishing point of the baseline (translation) direction.
- An epipolar plane is a plane containing the baseline. There is a one-parameter family (a pencil) of epipolar planes.
- An epipolar line is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines and defines the correspondence between the lines.

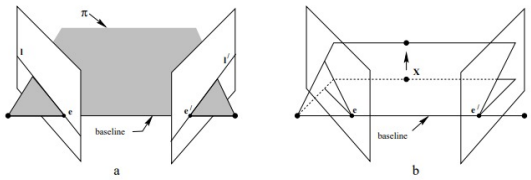

Fig. 9.2. **Epipolar geometry.** *(a) The camera baseline intersects each image plane at the epipoles* $\mathbf{e}$ *and* $\mathbf{e}'$*. Any plane* $\boldsymbol{\pi}$ *containing the baseline is an epipolar plane, and intersects the image planes in corresponding epipolar lines* $\mathbf{l}$ *and* $\mathbf{l}'$*. (b) As the position of the 3D point* $\mathbf{X}$ *varies, the epipolar planes "rotate" about the baseline. This family of planes is known as an epipolar pencil. All epipolar lines intersect at the epipole.*

2.6.1 Fundamental matrix :

The fundamental matrix is the algebraic representation of epipolar geometry. Suppose x and x' are two corresponding points. The fundamental matrix acts as a mapping between them.

$$x'^T F x = 0 \tag{10}$$

3. Image Matching Challenge 2022:

The image matching challenge is a yearly challenge hosted on Kaggle. In our final report, we tried testing our final pipeline on its dataset.

3.1. Goal of the competition:

In this competition, we aim to estimate the relative pose between two images from different viewpoints of the same scene.

3.2. Evaluation:

Submissions are evaluated on the mean Average Accuracy (mAA) of the estimated poses. Given a fundamental matrix and the hidden ground truth, we compute the error regarding rotation and translation. Given one threshold over each, we classify a pose as accurate if it meets both the thresholds. We do this over ten pairs of thresholds, a pair at a time (e.g. at 1 and 20 cm at the finest level and 10 and 5 m at the coarsest level).
We then calculate the percentage of image pairs that meet every pair of thresholds and average the overall results thresholds, which rewards more accurate poses. As the dataset contains multiple scenes, which may have a different number of pairs, we compute this metric separately for each scene and average it afterwards.

3.3. Constraints:

The submission notebook for the competition should follow the following constraints:

- CPU notebook = 9 hours runtime
- GPU notebook $\leq$ 9 hours runtime
- Internet access disabled.

Figure 2. Example training image

4. Pipeline:

For our final pipeline, we use an ensemble of matching points of LoFTR and SuperGlue. We used publicly available pre-trained models for inference. We did not do any fine turning as it did not give any significant improvement. Our final pipeline is a two-stage pipeline.

- In our first stage, we perform TTA(Test Time Augmentation) by scaling images to different resolutions and concatenating the matching points produced by LoFTR and SuperGlue on these resolutions. We took inspiration from the winning solutions of this competition to choose our dimensions(840, 1080, 1280).
- We then use DBSCAN to find the majority cluster and crop our images based on these clusters.
- We again use LoFTR and DBSCAN to find matching key points in the cropped regions with similar dimensions.
- We also perform image warping on our original image and use LoFTR to get key points.
- We finally concatenate the key points produced by cropped region and warped image to find our fundamental matrix. We used the in-built function textitcv2.findfundamentalmatrix with a maximum iteration of 10000 and threshold of 0.25.

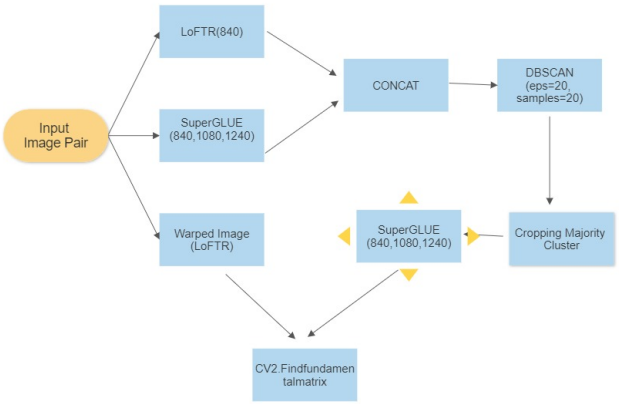

Figure 3. Final Pipeline

5. Implementation Details:

For scaling images, we scaled our longer side to the required dimensions and scaled the other side accordingly. After finding matching points for these scaled images, we rescaled our calculated key points to find key points wrt the original images.

For cropping the image, we find top-left and bottom-right key points and form a box according to them. We store the top-left key points and use them to get key points found in cropped regions wrt the original image.

6. Experimental Results:

We tried different models starting from the baseline model, such as LoFTR, SuperGlue, DKM, etc. Ensembling these models with text time augmentation, such as scaling and warping, improved the performance. In our final pipeline, we use DBSCAN clustering along with ensemble models to get the best results.

Pipeline	Private Score	Public Score
LoFTR	0.733	0.721
SuperGlue	0.675	0.678
LoFTR+Cropping	0.746	0.742
LoFTR+SuperGlue +Cropping	0.791	0.783
LoFTR+SuperGlue +Cropping+Warping	0.803	0.801

Table 1. Experimental results

7. Code References:

- The entire pipeline and helper function was coded from scratch.

Figure 4. Example test images

Figure 5. LoFTR+SuperGlue+Cropping+Warping

- We used publically available codes of LoFTR and SuperGLUE. All these models are pre-trained, and no fine-tuning was done.
- We have used the submission template from previous public submissions.

7.1. Code Guide:

Our final code submission contains two notebooks, LoFTR + SuperGLUE Notebook and Final Inference Notebook. The first notebook contains four versions, in which each version has implementations of the pipelines discussed above. The second notebook contains an implementation of the ensemble model only using SuperGLUE.

8. Conclusion

For our final pipeline, we use an ensemble of matching points of LoFTR and SuperGlue. We further use these matching points to find the majority clusters and again perform image matching on these clusters. We found out that TTA, such as warping, boosted our result, and our best submission got placed in the top 50 in the private leaderboard.

9. References:

1. LoFTR: Detector-Free Local Feature Matching with Transformers, Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, Xiaowei Zhou
2. SuperGlue: Learning Feature Matching with Graph Neural Networks, Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, Andrew Rabinovich
3. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu

10. Links:

Image Matching Challenge 2022 : Kaggle Link
Preliminary Report: PDF
Mid-term Report: PDF
Code Link: LoFTR + SuperGLUE Notebook, Final Inference Notebook