Deep learning Final Project

Loris

Abstract

This report presents an exploration of local feature matching techniques for reconstructing 3D models from 2D images, focusing on the 2022 Image Matching Challenge. We begin with a foundational approach using the Scale-Invariant Feature Transform (SIFT), a classic method for feature extraction and matching, and then explore more recent, deep learning based approaches such as LoFTR and LightGlue. The goal is to evaluate the effectiveness of these techniques in handling complex image sets with varying viewpoints, lighting, and occlusions. Our experiments reveal that while SIFT provides a basic solution to the problem, deep learning models like LoFTR significantly outperform traditional methods, offering higher accuracy in matching and 3D reconstruction. The report also briefly discusses the challenges faced during the Kaggle competition, and provides results of described methods.

1 Introduction

The rapid proliferation of digital images, primarily captured through smartphones, has opened new possibilities for advanced computer vision tasks, including the reconstruction of 3D models from 2D images. The 2022 Image Matching Challenge embodies this potential by encouraging the development of methods that leverage datasets to create 3D representations from ordinary photos. Traditional techniques like Scale-Invariant feature transform (SIFT) have been instrumental in this field, but the challenge posed by diverse viewpoints, lighting conditions, and occlusions demands more sophisticated solutions. This report delves into various methods applied to the challenge, starting with the well-established SIFT algorithm and progressing to cutting-edge deep learning approaches like LoFTR and LightGlue. We will briefly explain those models and then provide their results on particular problem.

2 Image Matching Challenge 2022

There is a vast amount of pictures circulating through our lives. Most of them are captured with our phones. There may be some snaps of famous landmarks like Colosseum or State of Liberty or something completely else. Those photos are two-dimensional. But there is an idea that we could do more with those photos. We can gather them and create three-dimensional view. This is the core idea of the Image Matching Challenge [**imagechallenge**]. We want to use Maching Learning to capture the richness of the world using vast amount of data.

2.1 Structure from motion

SfM is a technique for estimating three-dimensional structures from two-dimensional images. When one tries to make such estimation he ensures it that the images captured or the data collection of images is of good quality. This Challenge tries to do much more difficult task which is to build 3D models from many difficulties. For example the wide variety of viewpoints, lighting and weather conditions, occlusions from various objects.

2.2 Approach to problem

It is impossible to compare every pixel in one image with every pixel in the second image because this would be very costly. So the idea is to identify on picture some prominent points which could be matched with other picture. Typically these are corners of a window or edges of a concrete. To recap we are searching for some locations in an image that can be identified across different views or difficulties. We find those points, perform some local feature extraction and then try to match the descriptors to other ones from another image.

2.3 Scale-invariant feature transform

SIFT is an algorithm from 1999 described in the paper with title Object recognition from local scale-invariant features [**lowe1999object**] which combines all the ingredients for solving the problem. Nowadays it is obsolete but never the less a good baseline for us to tackle the problem. First the keypoints are extracted and described from a set of images. These keypoints are then matched based on Euclidean distance of their feature vectors.

2.4 Fundamental Matrix

The final result of our solution will be fundamental matrix which encapsulates the projective geometry between two views of the same scene. This would be our output to the

evaluation. It has dimensions of 3x3 and relates corresponding points of images. Fx describes an epipolar line on which the corresponding point x' on the other image must be. What is true is x'Fx=0. There must be at least 7 corresponding points to estimate that matrix, but we will lift up the limitation to 8 because estimation method of fundamental matrix uses different algorithm for 7 points.

2.5 Kaggle competition rules

There is a rule that there must be no internet for the scoring phase on the test set of competition. That caused a lot of troubles as it was our first time doing a competition in Kaggle.

The score of the implementation is computed with mean Average Accuracy (maAA). Provided the fundamental matrix by the submission they calculate the error in rotation and translation. By thresholding they classify a prediction as accurate if it meets both thresholds. At last they calculate the percentage of pairs that pass thresholds and average those results. For each scene the metrics are separately computed and averaged afterwards.

2.6 Kaggle competition problems

We must state that we had a great deal of troubles with Kaggle competition framework. Debbuging is extremly hard to do. There was one problem at the beggining and the Kaggle report stated that only that there has occurred an error (Notebook throwed exception). How to debug that? There were no errors with a few test samples that are available probably just for that particular cause to debug. Also no errors with the training data. So there must be some instance that has a peculiar formation of input that disturbs some code. At the end it turned out that function find Fundamental matrix from opencv causes this error. And then we imposed all the limitations, to the input of that function, that we could think of. Sadly error persisted.

Next major problem was offline installation of LightGlue because there is a limitation that there must be no internet connection. Everything should be done offline. You need to download the requirements on that machine. Download it to local computer and then upload it so it can be posed as dataset which you can use offline. And at last, every time that you initialize a model there occurs a download of that model.

We know that those problems are not written in the papers, but this is a report and it should be noted that there was a lot of time lost due to this problems or impositions.

3 Methods - Deep learning approach

We tried two different deep learning techniques to solve a problem. They are described in the next two subsections.

3.1 LoFTR

In classical matching of two images, there is first feature detection then feature description and at last feature matching. In feature detection the prominent points are detected for each image. Then local descriptors are formed around neighborhood regions of those selected points. Finding correspondences consists of searching matching points with nearest neighbor search or something more sophisticated. We must keep in mind though that the feature detector stage reduces the search space of matching. It may well be that the feature detector fails to extract enough matching points and then it is impossible to perform task correctly. In this subsection we will briefly describe LoFTR which was proposed in the article named LoFTR: Detector-free local feature matching with transformers [**sun2021loftr**].

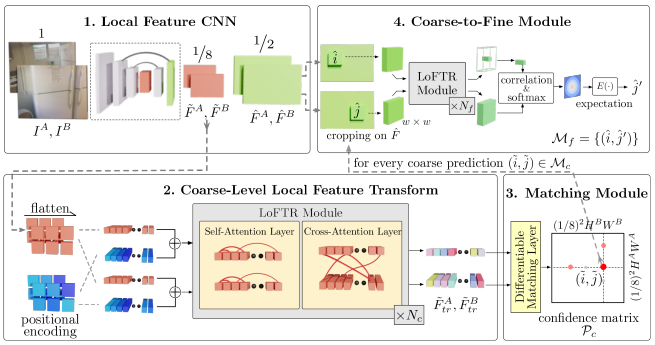

Figure 1: LoFTR architecture.

LoFTR is composed of four stages.

1. Local Feature Extraction.

   It uses a standard convolutional network with Feature Pyramid Network to extract features from both images. It outputs coarse-level features that are at 1/8 or the original dimension and fine-level features that are at 1/2 of the original dimension. Both for each of two images. The Convolutional Neural Networks which are used for this first stage have some useful properties for local feature extraction such as translation equivariance, locality and also downsampling comes in handy for the smaller input to the LoFTR module.

2. Local Feature Transformer (LoFTR) Module.

   The coarse level features from both images are passed through the LoFTR module to extract position and context dependent local features. What LoFTR module do from these coarse level features if that it turns them to representations that are easy to match.

   It is passed through transformer. But the computational cost grows quadratically with the input so vanilla transformer is not suitable for the task. Therefore it uses Linear Transformer to reduce the computation complexity. It is done by using alternative kernel function in attention layer instead of using exponential kernel used in the original.

   It uses standard positional encoding with 2D extension and it is added at the input to LoFTR module. So the features become position dependent. That position dependence helps LoFTR in indistinctive regions to make matches. That is how it makes it possible to describe homogenous white walls for example.

3. Matching module.

   The output of the LoFTR module goes through differentiable matching layer which produces the confidence matrix. It does so with finding score matrix by making

inner product of the two input matrices. Then this score matrix is passed through dual-softmax to obtain matching probability. Based on that matrix the matches are selected. Those that are below threshold are discarded and those above are kept.

4. Coarse-to-Fine Module.

   Coarse matches are combined with fine matches in Coarse-to-Fine module. First the position of coarse matches is found in fine matches. This is combined and cropped to go to LoFTR module. The module yields two transformed local feature maps. The vectors are correlated and this produces a heatmap that represent the matching probability of each pixel. The expectation is computed over the probability distribution to get final position.

3.2 LightGlue

Lightglue is a neural network that matches two sets of local features from two images. It rejects non-matchable points and jointly finds correspondences.

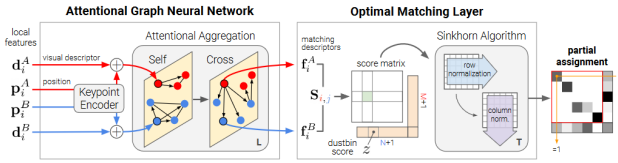

Figure 2: SuperGlue architecture.

Traces of LightGlue go to SuperGlue where they proposed to look at the feature matching problem in another way. SuperGlue was proposed in a paper called Superglue:Learning feature matching with graph neural networks [**sarlin2020superglue**]. Clearly LightGlue and SuperGlue differ from LoFTR because LoFTR does everything. It detects and describes features and then does also the matching process. LightGlue is only about matching phase. The feature detection and description should be done beforehand and serve as input to the LightGlue. Instead of focusing on learning better task local features the SuperGlue proposed to learn matching process from the local features that already exist. That means that we can use SIFT for local feature description and then pass those features ahead to SuperGlue to match them. We will first briefly describe SuperGlue as it is the predecessor of LightGlue.

The motivation of the SuperGlue idea is that they eliminate the need for domain expertise and learn relevant priors directly from the data. Super glue works in two steps. One is Attentional Graph Neural Network and other is Optimal Matching Layer. The first one is a keypoint encoder that combines together the position and descriptor of keypoint. This serves as the input to Transformer. Transformer utilizes self and cross attention to enrich information through multiple layers. This creates more powerful representations of keypoints. Then there is Optimal Matching Layer which produces partial assignment matrix. Partial assignment means that only a subset can actually be assigned successfully. This arises when the data is not exact. In this case we must assume that not all of the keypoints would be correctly determined and matched. It starts off by producing score matrix which is just inner product of the two outputs from Attentional Graph Neural Network. To obtain partial assignment matrix from score matrix we must somehow incorporate the idea of inexactness. This is done via dustbin so the unmatched keypoints are explicitly assigned to it. This is now optimization problem and it is closely connected to optimal transport between discrete distributions. This is solved with Sinkhorn Algorithm.

LightGlue was proposed in paper called Lightglue: Local feature matching at light speed [**lindenberger2023lightglue**]. LightGlue in its basic interpretation revisits SuperGlue and makes simple but effective improvements. These modification make many positive contributions for example they make it more efficient, more accurate, and easier to train. LightGlue considers difficulty level of each image pair. This largely varies and it is dependent of visual overlap, appearance changes and other information. What that means is that the easy cases are computed much faster than the hard ones. This is done in a manner that at every layer it decides if further computation is required. LightGlue is a plug and play replacement for SuperGlue and that is why we first discussed SuperGlue.

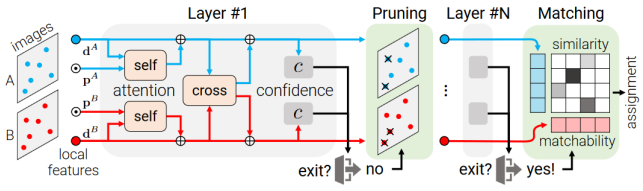

Figure 3: LightGlue architecture.

For Lightglue each local feature is composed of two things. First one is position of feature and second one is visual descriptor. Transformer works with this information. Each layer is composed of self and cross attention units. These enrich representation of features. At each layer there is classifier. This classifier decides if it should halt the inference of some features that are likely not to yield any good result. With this there is much unnecessary computation avoided. The lightweight head for correspondence prediction that lies at the end of the architecture similarly to SuperGlue it computes a pairwise score matrix between points of both images. And for each point also matchability score is computed. This signifies how likely is point to have a corresponding point. If keypoint is not present in the other image it is definitely not matchable. This two scores are combined and produce soft partial assignment matrix. This yields a correspondence. The points over a threshold are selected if they are predicted as matchable and their similarity score is greater than all the rest.

There are two more things to consider in LightGlue architecture. First one is confidence classifier. While the backbone of the LightGlue enhances the inputs of visual descriptors these become reliable if the case is easy. The predictions become confident and always more identical to previous layers. This is a sign that the output of model is good enough

and the model can stop inference for those points. This confidence of points is predicted at each layer. It adds a bit of computation time but it often saves much more. Second one is exit criterion. This one halts the inference if a sufficient ration of all points is confident. There is also point pruning. If points are predicted as confident and also unmatchable this signifies that they will be hard to match. This points are discarded at each layer and only prosperous points are passed for further inference.

4 Dataset

There is provided training set for experiment. The hidden set is not available to see. We can not assume anything particular about hidden test set. But the methods that we applied got pretty much similar score on training and on test data. We can say that they can not be drastically different because of the results.

In the training set there are 16 different scene locations. But each of that scenes has different amount of pictures. For example, scene of British museum has 176 pictures whereas scene of Buckingham palace has 446 images. This will not be so much important to us because we won't be training anything. We will try to use existing models and attempt to find correspondences.

In figure 4 there are examples of images from scene Buckingham palace. We can see that the images are really diverse. There are many problems with them. Occlusions are everywhere. Let us describe a few of them to get a feeling for dataset. Look at the example (b). It has a frame, there is occlusion of the statue and there is a vivid shiny tone to the colors. In example (c) there is a man posing in front of everything. Example (f) is taken almost at night in strange lightning conditions. Example (h) is challenging even for human to locate on the building where does it fit. Example (g) has sun directly shining to a camera. And to conclude the examples (e) and (d) have almost more focus on cars or bikes than the object of our interest. We can conclude that with that data achieving the goal is not easy at all.

5 Results

We started off with basic model of our choice that was SIFT. It dates back in time but we got some baseline results written in 1. We expected better results with this model, but as the dataset goes it is not that surprising. In figure 5 we can see the result of the SIFT algorithm and brute force matching algorithm. It can be seen that some matches are completely out of order but otherwise it produced result that is not so terrible. Figure 6 shows how those matches obtained are filtered with RANSAC. Suddenly there are a lot less matching points.

Next we tried to produce some better results with LoFTR which is based on deep network as described earlier. The score as it is written in table 1 is much better than with SIFT. It was expected to be that way because the approach was developed in 2021 whereas SIFT is from 1999. But bear the attention because even SIFT result will be improved. The figure 7 shows a mess of matches that produced LoFTR. It gives an impression that there are more matches and those lines connecting them are more synchronized than in the figure 5. But

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 4: Sample images from dataset of scene Buckingham palace.

one can not make a conclusion based on that image. Figure 8 shows only 200 random matches. And now we can also that the results are better than previous method shown in figure 6.

Best results gave LightGlue which is most recent approach. But it differs from previous two as it is only takes care of local feature matching. Whereas LoFTR takes all process to itself. We have used five different feature detection and description algorithms that are written in table 1. Best result was yield by the combination of ALIKED and LightGlue. It scored on the test set (Public Score) 0.76 mean average accuracy. The best result of competition was 0.86. Our score is not bad but the best one is much better. It was composed of much tweaking and combination of many different approaches.

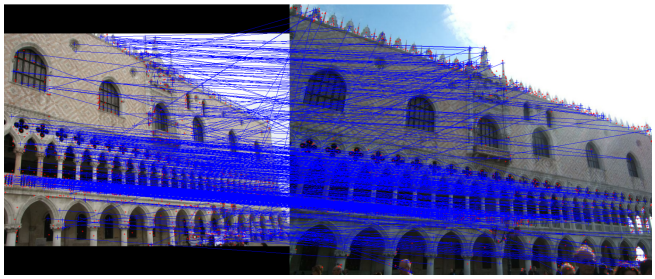

Figure 5: Matches before RANSAC.

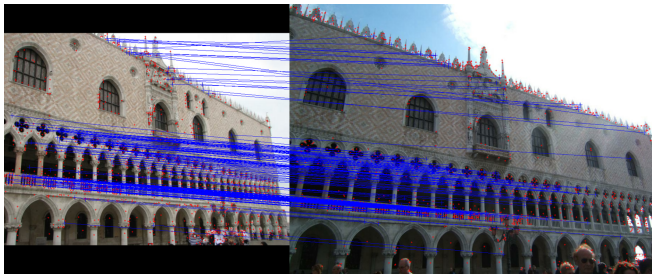

Figure 6: Matches after RANSAC.

Experiment	Type	Private Score	Public Score
1	SIFT & BF matcher	0.21	0.22
2	LOFTR	0.59	0.57
3	SIFT + LightGlue	0.61	0.60
4	DogHardNet + LightGlue	0.66	0.64
5	SuperPoint + LightGlue	0.69	0.69
6	DISK + LightGlue	0.73	0.70
7	ALIKED + LightGlue	0.77	0.76

Table 1: Table of experiments.

6 Discussion

The results of our experiments highlight the significant advancements in image matching techniques over the past few decades, particularly when comparing traditional methods like SIFT with modern deep learning approaches such as LoFTR and LightGlue. The initial experiment using SIFT provided a useful baseline, but its performance was limited by the challenges inherent in the dataset, such as variations in lighting, occlusions, and viewpoint changes. These limitations are evident in the relatively low accuracy scores achieved with SIFT.

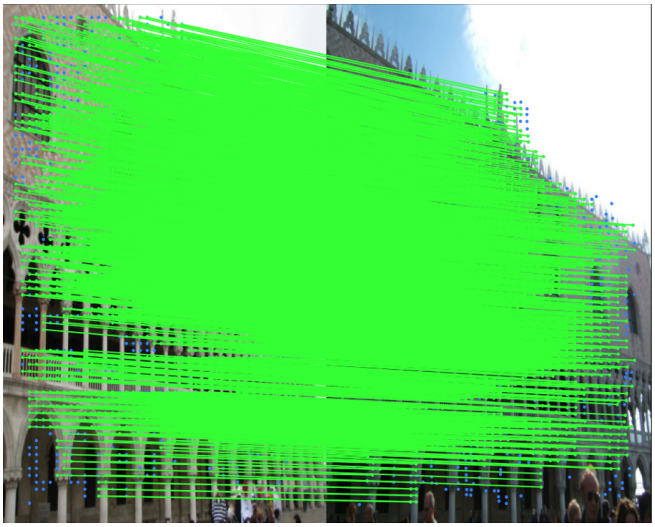

Figure 7: LoFTR result.

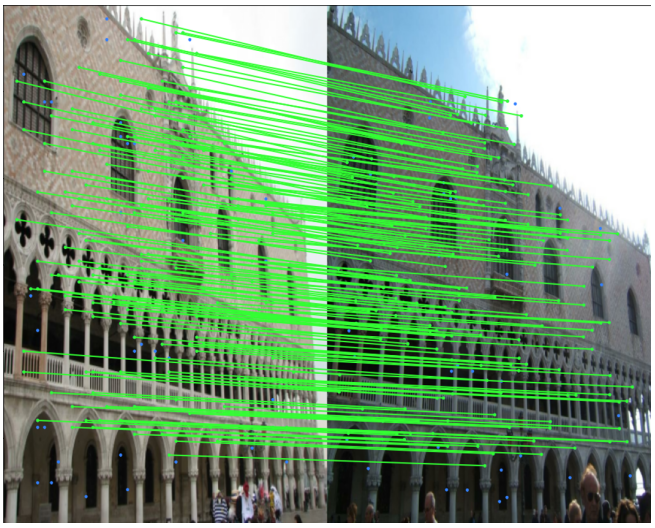

Figure 8: LoFTR partial result for easier viewing.

The shift to deep learning-based methods marked a substantial improvement in performance. The most notable improvement was observed with the implementation of LightGlue, particularly when combined with advanced feature detectors and descriptors like ALIKED. LightGlue's architecture, which focuses on the feature matching process while leveraging feature detectors, allowed it to achieve the highest accuracy scores in our experiments. LightGlue, which can be paired different feature detectors and extractors, provided a flexible and effective solution that outperformed the approach of LoFTR which does the whole process by itself.