

Image Matching Challenge 2022 - An ensembling approach

Anonymous CVPR submission

Paper ID *****

Abstract

The image-matching task in computer vision presents two images of the same landmark, captured by two different cameras in two different positions and asks for a 3×3 matrix describing the relative change in pose from the first camera to the second. We investigate several avenues for improvement over existing methods on the task of finding this 'fundamental matrix.' Despite the attention this problem has received in recent years, we are able to provide some new insights into improving performance. As a contribution to quantity of data and quality of training, we rigorously formulate a versatile and non-trivial data augmentation strategy and show empirically that it is effective. As for our contributions to models for this problem, we show one model, LoFTR to be extensively trained enough that fine-tuning on a large new dataset offers no significant improvements. With this shown, we direct our attention to exploring the aggregated predictions of an ensemble of models for the task. This is well-motivated since all the most successful approaches to image-matching arrive at a list of candidate pairs of points (one in each image) which should correspond to the same point in 3D space and thus impose a conditional on the fundamental matrix. We explore several methods of aggregating models' lists of candidate matching point pairs and report on their efficacy.

1. Introduction

What if machine learning could help better capture the richness of the world using the vast amounts of unstructured collections of images freely available on the internet?

Two or more images of the same object taken from different perspectives can be used to reconstruct a 3D model of the object. *Image matching* involves identifying the correspondence between pixels of a pair of images which capture the same physical features of a scene. This task is central to 3D reconstruction.

Using crowd-sourced images (instead of images captured by professionals for the explicit purpose of 3D reconstruction) is made difficult by varying lighting, weather

conditions, viewpoints and sometimes even user-applied filters. In addition, crowd-sourced images may have occlusions from people and vehicles. Nevertheless, the relative ease of obtaining crowd-sourced images versus professional images makes it worthwhile to solve the 3D reconstruction problem in the harder crowd-sourced setting.

Our goal, in this challenge, centers on an important intermediate step in 3D reconstruction: estimating the relative camera pose between two images. That is, given a pair of images and internal properties of the cameras which captured these images, we seek to capture the relative pose of the second camera with respect to the first by finding the *fundamental matrix*, F which maps pixels in the first image to their corresponding pixels in the second. The core task involved in the estimation of the fundamental matrix is image matching.

1.1. Forward Imaging model of a camera

To understand the fundamental matrix, we examine the process by which a camera projects a 3-D point in the external world onto a 2-D plane. This projection can be described by a forward imaging model which consists of two stages. First, a point in a global frame of reference is transformed into a point in the camera's frame of reference. This first step can be described in terms of a rotation matrix R and a translation vector t . Together, R and t are known as the camera *extrinsics*, capturing the position and orientation of the camera in the global reference frame. Precisely, a 3-D point M_i in the global frame of reference is transformed into $RM_i + t$ in camera's frame of reference.

In the second stage, the 3-D point in camera's frame of reference is projected onto the 2-D plane of the image. The calibration matrix K , the camera *intrinsics*, captures camera properties that determine this projection. The projection of a 3D point M_i , in meters, into 2D coordinates m_i , in pixels, is given by $m_i = KM_i$ where a convenient parametrization of K is

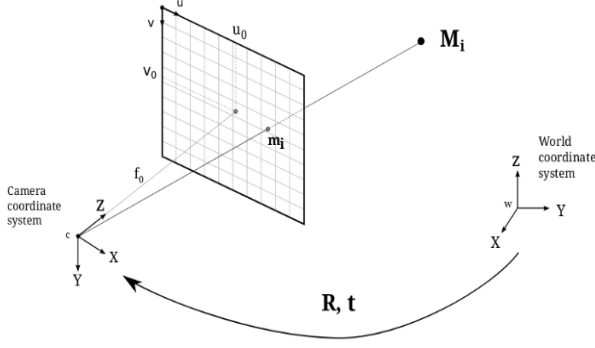


Figure 1. Forward Imaging Model of a Camera. Figure from [2].

$$\begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Above, f_x, f_y are the focal lengths of the camera in pixels along x and y axes of the image plane, and u_0, v_0 are the principal point offsets (i.e. the center of image plane assuming left-upper corner of the image plane to be the origin).

Altogether, a point M_i in global frame of reference gets projected to 2-D point m_i according to

$$m_i = K(RM_i + T).$$

1.2. Epipolar Constraint

Given a 3D point M_i , in meters, in the global frame of reference and its projections on two different cameras m_1 and m_2 , in pixels, the fundamental matrix F for these two views must satisfy

$$m_1^T F m_2 = 0.$$

Essentially every effective approach to finding F proceeds by creating several candidate pairs (m_1, m_2) and using the constraint $m_1^T F m_2 = 0$ from each to estimate F . The search for (m_1, m_2) pairs is called *image matching*.

2. Dataset

Only the training data has been made available to the contestants. It contains data from 16 landmarks. There are hundreds of images of each landmark (a total of 5720 images across landmarks). For each landmark, there is a calibration matrix that provides the camera intrinsics, rotation matrix and the translation vector for each image. Also available is a covisibility matrix that provides, for each possible pair of images of the same landmark, an estimate of overlap between the images as well as the fundamental matrix. Our goal is to estimate the fundamental matrix given a pair of images.

The test set is hidden and will contain approximately 10,000 images.

2.1. Evaluation

The output of the model, say $\hat{F}$, is an estimate of relative pose of one image with respect to another, given a pair of images. A model is evaluated on the *mean Average Accuracy (mAA)* of the estimated poses. For estimated fundamental matrix $\hat{F}$ and hidden ground truth F , error is estimated first in terms of the rotation (ϵ_R , in degrees) and second in terms the translation (ϵ_T , in meters) required to obtain F from $\hat{F}$. Given a rotation threshold and a translation threshold, a pose is classified as accurate if it meets both thresholds. This is done over ten pairs of thresholds, one pair at a time (e.g. at 1° and 20cm at the finest level, and 10° and 5m at the coarsest level):

`thresholds_r = np.linspace(1, 10, 10) # In degrees.`

`thresholds_t = np.geomspace(0.2, 5, 10) # In meters.`

Then, for each pair of threshold, the percentage of image pairs (A, B) for which $\hat{F}(A, B)$ meets the threshold is calculated. This percentage is then averaged over all thresholds. As the dataset contains multiple scenes, which may have a different number of image pairs, this metric is calculated separately for each scene and then averaged to a get a single number.

3. Related Work

The classical approach to the problem of estimating the fundamental matrix F from two images A and B is a modular approach consisting of five subtasks: 1) detect keypoints in both images (e.g. corners), 2) construct a visual descriptor in $\mathbb{R}^d$ for each keypoint, 3) match keypoints in A to keypoints in B , 4) retain only the highest confidence matches, and 5) estimate F from these matches. Below we begin with a benchmark pipeline which handles each of these five steps separately. We then discuss SuperGlue, introduced in 2020, which handles steps 3) and 4) at once, and LoFTR, developed in 2021, which implements an end-to-end approach for steps 1) through 4).

Benchmark Pipeline Following [4], we describe the image matching pipeline. The image matching benchmark pipeline takes as input N images per scene. Input images are fed into feature-extraction module, which computes up to K features from each images. This module generates a list of putative matches for each of the $\binom{N}{2}$ image pairs. The next module (outlier pre-filtering) is an optional processing step in this pipeline. After pre-filtering, matches are fed to two tasks: the stereo task, is used to estimate the Fundamental matrix between tentative matches. Parallel to the stereo task is the multi-view module which provides re-constructed scenes from small image subsets (of sizes 5, 10 and 25). The image matching benchmark pipeline is built to benchmark local features and robust estimation algorithms.

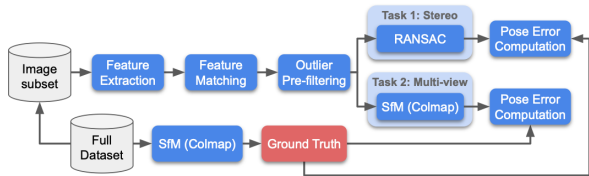


Figure 2. Benchmark Pipeline. Figure from [4].

Its modular structure allows easy integration, configuration and combinations of different methods and popular algorithms. For example, various descriptors could be used in the feature extraction module, such as Hardnet, HardnetA-mos, GeoDesc, SOSNet, L2Net, D2-Net etc.

The SuperGlue [13] approach uses a graph neural network to obtain matchings of keypoints along with confidence scores attached to the matching. In this way, the authors retain only the highest confidence matches 'for free' and outperform other learned approaches on multiple standard datasets for the task. With an input of keypoints and their descriptors from each image, SuperGlue uses learned parameters to formulate the matching between keypoints in A and B as an optimal transport problem. For each pair of keypoints (a, b) (from images A and B respectively) SuperGlue outputs a similarity score $S_{a,b}$ representing the confidence of the model in matching keypoint a to keypoint b . The optimal transport problem, to which the Sinkhorn algorithm provides an "efficient yet simple approximate solution," is simply to optimize $\sum_{a,b} S_{a,b} P_{ab}$ for a matrix $P \in [0, 1]^{M \times N}$ where there are M keypoints in image A and N in B . Here P represents a *partial assignment* of keypoints: P_{ab} gives the probability of matching keypoint a with keypoint b . That P encode a probability distribution means P is subject to linear constraints, and maximizing $\sum_{a,b} S_{a,b} P_{ab}$ subject to these constraints is the well-understood aforementioned optimal transport problem. The remaining problem then is to choose scores $S_{a,b}$ well. To do this, SuperGlue learns embeddings (f_a for keypoint a) through alternating self and cross attention. After calculating embeddings using many layers (9 in their implementation), $S_{a,b}$ is simply taken to be the inner product $\langle f_a, f_b \rangle$.

In contrast to SuperGlue which is a detector dependent method, LoFTR [14] does a detector-free local matching. SuperGlue is dependent on a feature detector and its attention range is limited to local area around interest points. Due to these factors, it struggles to match repetitive patterns in input image pair. LoFTR avoids both these limitations using a CNN to extract dense features and using a transformer to capture global context in an end-to-end model. It builds on NCNet [12] which proposed an end-to-end learning approach for doing dense image matching. In NCNet, features from the two images are first extracted using a CNN.

Then, a 4-D pair-wise feature-mapping is calculated using cosine similarity. 4-D convolutions are then used to enforce neighborhood consensus. In Efficient NCNet [11], lower resolution versions of input images are used decrease the computation requirements. Then, a relocalization layer which uses high-resolution local features around candidate points is used to improve the estimate of the position of the matches. LoFTR has the same blocks in its architecture. It uses a Feature Pyramid Network [7] for features extraction. Its authors recognized that 4-D convolutions will capture local context but fail to capture the global context of the image. They use a transformer stage to capture global context and calculate the similarity between extracted features. This improves the model's performance on image pairs with repetitive patterns.

ALSFeat [8] puts geometric (similarity, affine, homography) constraints to extract more informative features. It uses the coarse-to-fine hierarchy like previous models for accurate keypoint localization.

Deep Kernelized Dense Geometric Matching [3] use kernel regressions, unlike interleaved attention used by LoFTR, on global correlation volume to predict dense matches.

4. Methods

We begin with methods for determining several baselines for the task of predicting the fundamental matrix. Most of the choices that we make and defend occur in the genuine experiments which follow, but baseline methods are included for completeness. We proceed to give our methods for data augmentation and describe a process of generating new data which exactly matches the input data of the following experiment - fine-tuning the LoFTR model. Finally, we discuss our approach for LoFTR fine-tuning.

4.1. Baselines

We run the models from many previous publications or projects attacking this problem. To evaluate according to involved procedure described in section 2.1, we leverage freely available Kaggle notebooks. The trained models themselves were all publicly shared by other participants in the contest. The first model uses pretrained KeyNet [5] and AffNet [10] feature detectors and HardNet [9] feature descriptor. The second model uses a pretrained SuperGlue with SuperPoint [1] extractor-descriptor. The third baseline is for a pre-trained LoFTR model. We also use pretrained ALSFeat and Deep Kernalized Matcher ("DKM") to generate baselines.

4.2. Data Augmentation

4.2.1 Formula for Data Generation

Recall from the epipolar constraint that points m_1 and m_2 which are projections of the same 3D point onto two differently positioned cameras satisfy

$$m_1^T F m_2 = 0$$

where F is the fundamental matrix describing the difference in camera pose. Consider two images, I_1 and I_2 so that m_1 and m_2 above correspond to I_1 and I_2 respectively. To create new examples from existing training examples as data augmentation, we use the epipolar constraint to solve the general problem of how the fundamental matrix F should change if we apply traditional data augmentation techniques from computer vision to the associated images - I_1 and I_2 . Specifically, given any invertible linear transformation A on $\mathbb{R}^3$ and training example (I_1, I_2, F) , we develop a procedure to determine F' such that (AI_1, AI_2, F') is a new, valid training example. As a motivating example for discussion, we will frequently refer below to the special case of cleanest non-trivial A , namely the linear transformation encoding "reflection across the y-z plane" (or, colloquially, "flipping left and right"). Below, A is a general invertible linear transformation on $\mathbb{R}^3$ and $A_{\text{left-right}}$ is the specific case

$$A_{\text{left-right}} := \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Before solving the problem as stated, one final clarification is needed. The "traditional data augmentation techniques from computer vision" discussed above usually describe transformations (linear or otherwise) of data which has two spatial dimensions. How does this relate to the general linear transformation A of $\mathbb{R}^3$ which we consider? In the case of $A_{\text{left-right}}$, the associated transformation of image data with two spatial dimensions is clear: exchange pixels with spatial coordinates (x, y) and $(W - 1 - x, y)$ for any x, y where W is the width of the image in pixels. Not every A yields such an unambiguous transformation of image data. Consider, for example

$$A_{\text{near-far}} := \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix}.$$

for which we can conceive no analogous transformation of pixels. Nevertheless, since we can handle general A mathematically, we do so below. In code, we implement the $A_{\text{left-right}}$ case as ample proof of concept. By abuse of notation (by which we hope to be more clear) we refer to the 2D image resulting from applying the pixel map above to image I as $A_{\text{left-right}}I$.

For any A as defined above and training example (I_1, I_2, F) , the new fundamental matrix F' must satisfy the following for all m_1 and m_2 :

$$(Am_1)^T F' (Am_2) = 0.$$

Equivalently, we must have

$$m_1^T (A^T F' A) m_2 = 0$$

for all m_1, m_2 . Thus, the correct new fundamental matrix F' satisfies

$$F = A^T F' A.$$

We deduce the correct expression for F' using that A is invertible:

$$F' = (A^T)^{-1} F A^{-1} = (A^{-1})^T F A^{-1}.$$

In the specific case of $A = A_{\text{left-right}}$, we have $A_{\text{left-right}}^{-1} = A_{\text{left-right}}$, giving

$$F' = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} F \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

It is worth noting that $A_{\text{left-right}}$ does not commute with all invertible linear transformations (in other words, $A_{\text{left-right}}$ is not in the center of $\text{GL}_3(\mathbb{R})$) so "almost always," we have

$$F \neq \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} F \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = F'.$$

4.2.2 Procedure for Data Generation

Our full pipeline for data augmentation and evaluation consists of 1) generating new data according to the rule outlined above, using $A = A_{\text{left-right}}$ and 2) evaluating the new data, or demonstrating that this is reasonable training data. There genuinely is something to be shown here. Unlike traditional data augmentation of data with two spatial dimensions, there is an assumption we make which underpins the success of data augmentation. Specifically, we assume that $A_{\text{left-right}}$ more-or-less corresponds to the transformation of 2D given by $(x, y) \mapsto (W - 1 - x, y)$ as discussed above. As mentioned, the corresponding 2D transformation is much less clear for other linear transformations, and thus we see no rigorous way of justifying this assumption except by empirical performance. To this end, we load a trained image-matching model (specifically LoFTR) which is known to have high accuracy in the fundamental matrix it produces. We then compare its output, say $\hat{F}'$ on image pairs $(A_{\text{left-right}}I_1, A_{\text{left-right}}I_2)$ to the label, F' , generated by data augmentation. The appropriate distance metric to use here is the operator

norm, since F' is viewed as an operator on $\mathbb{R}^3$, not simply a vector in $\mathbb{R}^9$. We interpret this difference as giving the error of our generated training example F' from $\hat{F}'$, which is treated as the ground truth. In short, we have the following procedure:

1. **Generate new data** according to $(I_1, I_2, F) \rightarrow (A_{\text{left-right}} I_1, A_{\text{left-right}} I_2, (A_{\text{left-right}}^{-1})^T F A_{\text{left-right}}^{-1})$.
2. **Evaluate new data** by measuring $\|F - A_{\text{left-right}}^{-1})^T F A_{\text{left-right}}^{-1}\|_{\text{GL}_3(\mathbb{R})}$.

4.3. LoFTR Fine-tuning

The pre-trained models were all trained on MegaDepth dataset [6]. Although the model has images from 196 locations many of which are important historical landmarks, we tune the model to the dataset provided by Image Matching Challenge 2022. Our hope was that the model will focus more on the sub-population of locations expected in our test set and learn more complex features. To finetune the model, we first estimated the depth as the original model was trained on MegaDepth. Then, we follow the training procedure described in the LoFTR paper (and use the config files provided by the authors).

4.4. Ensemble

The challenge in this problem is that output of the models is not a single number but a large collection of numbers. The models first yield pair of matching points from the pair of input images. These matching points are then used to calculate the F-matrix which is then used to extract the relative pose.

The obvious choice for an ensemble is calculating an average of the estimates that we get from the various models. We call the "simple average of F ensemble". Given the models may perform differently, we may want to weight the F-estimate according to the models performance. We call this "probability weighted F ensemble". We convert mean average accuracy of the models into a weights that sum to 1. Because the input F-matrices meet the epipolar constraint, a linear combination thereof will also meet the epipolar constraint.

A novel ensemble We also consider ensembles of matching points. We consider two ensembles: first take an intersection of matching points. Consensus by models on a pair of points increases our confident in them. It is likely that such a set will yield a less noisy estimate of F. The other choice we consider is a union of matching points. One can also expect that in presence of repeating texture, obstacles etc. having a larger set of matching points may help in a better prediction of F.

For the ensembles we consider our two best performing baselines as input models: LoFtr and ALSFeat.

5. Experiments

5.1. Baselines

Model	Score on Public Test
KeyAffHardNet baseline	0.523
SuperGlue baseline	0.676
LoFTR baseline	0.726
ALSFeat baseline	0.673
DKM baseline	0.668
LoFTR finetuned	0.721
LoFTR + ASL Simple-avg	0.682
LoFTR + ASL Prob-wted	0.655
LoFTR + ASL Intersection	0.557
LoFTR + ASL Union	0.717

Table 1. Model performance: Baselines and Ensembles

Among the baselines LoFTR does the best. All other baselines have an mAA around 0.67 indicating that LoFTR is doing something novel or trains better on landmarks. A deeper look at the points matched by the models 3 shows us that DKM's output is the most dense whereas the ALSFeat's output is the least dense. This indicates that although DKM outputs a dense match, the match is probably of lower quality. ALSFeat's geometric restrictions yield high quality but fewer matches.

5.2. Data Augmentation

We ran the data augmentation procedure on a small test set provided by the Kaggle competition and obtained the following results.

Mean of errors	Variance of errors
0.030	0.00027

This is undoubtedly quite a low mean error. Nevertheless, we recognize that it is tempting to have some baseline for what size error should be considered "good enough." It is meaningless to compare the operator norm difference $\|F' - \hat{F}'\|$ to the operator norm of the difference of two random 3×3 matrices, which is surely at least 1 for any reasonable definition of 'random'. Instead we content ourselves with the fact the operator norm inherently measures the worst-case deviation of the second camera's projection of a point from the projection predicted from the first camera's projection of the same point and from F' (scaling this quantity by the size of the point in $\mathbb{R}^3$). That this ratio is 0.03 at best is good enough.

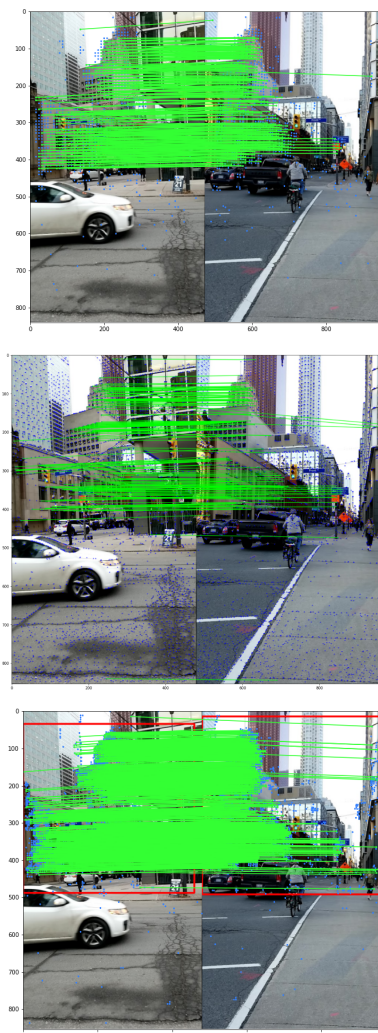


Figure 3. Matching-point Cloud: Top image is output of LoFTR, middle image is from ALSFeat and the last image is from DKM

5.3. LoFTR Fine-tuning

Finetuned LoFTR model does marginally worse on the public test. We note the finetuning has similar matched points 5. The pretrained model does as well as the finetuned model as it is trained on a larger more diverse dataset which has a large coverage of fine-tuning data.

5.4. Ensemble

Our ensembles yield mixed results. Our input models were ASLFeat and LoFTR. We find that intersection of matched-points leads to dramatic fall in performance. This may be due to the fact that ASLFeat is very selective as it puts geometric constraints on points matched. This possibly reduces that matched-pairs that are common across the two models. Surprisingly, the union of point clouds doesn't

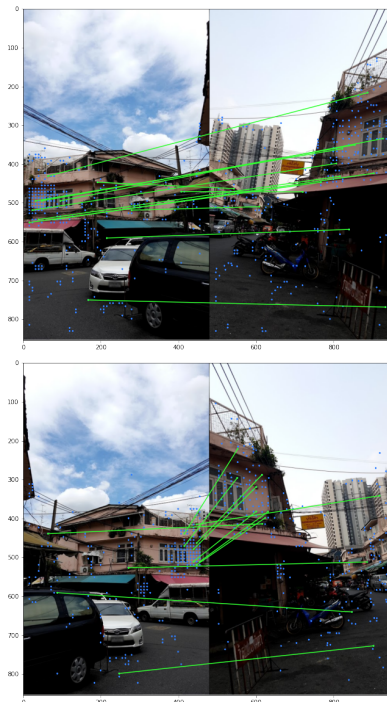


Figure 4. The upper picture is the original training example, and the lower picture has been flipped horizontally. Green lines connect points matched by LoFTR in the process of predicting F and F' .

improve performance either.

We find that aggregating estimates of F doesn't help. It in general leads to worse performance. This is counter-intuitive as an average should lead to a less noisy estimate.

6. Conclusion

Our problem setting provides fertile ground for investigating generally settled questions: how should we augment data? How should we build an ensemble? For example, the relative pose between camera changes if we horizontally flip a pair of images. Similarly, since our output is multi-dimensional, it is not obvious how averaging or consensus techniques should be applied.

We developed a rigorous approach to augment data. This was achieved by mapping image operations to changes in F -matrix.

For aggregating estimates from many input models, we proposed averaging-ensembles for our estimates of F and consensus ensembles (intersection and union) at the level of our estimates of set of matched points.

7. Contributions and Acknowledgements

We have used the original code and pretrained weights provided by the authors of the models considered in this

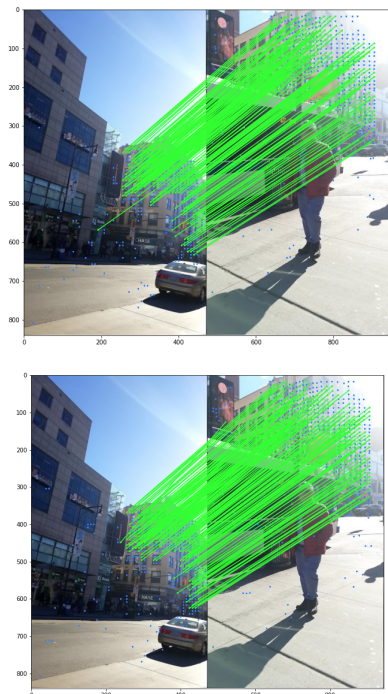


Figure 5. Matching-point Cloud: Top image is output of LoFTR fine-tuned, bottom is from the pretrained model

paper to run our baselines. We also looked at some public notebooks in kaggle to get started.

References

- [1] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. *CoRR*, abs/1712.07629, 2017. 3
- [2] Kornia developers. "Pinhole Camera". [KorniaSite](#)
- [3] Johan Edstedt, Mårten Wadenbäck, and Michael Felsberg. Deep kernelized dense geometric matching. *CoRR*, abs/2202.00667, 2022. 3
- [4] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image matching across wide baselines: From paper to practice. *CoRR*, abs/2003.01587, 2020. 2, 3
- [5] Axel Barroso Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key.net: Keypoint detection by hand-crafted and learned CNN filters. *CoRR*, abs/1904.00889, 2019. 3
- [6] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 5
- [7] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 3
- [8] Zixin Luo, Lei Zhou, Xuyang Bai, Hongkai Chen, Jiahui Zhang, Yao Yao, Shiwei Li, Tian Fang, and Long Quan.

- Aslfeat: Learning local features of accurate shape and localization. *CoRR*, abs/2003.10071, 2020. 3
- [9] Anastasiya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. *CoRR*, abs/1705.10872, 2017. 3
- [10] Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Learning discriminative affine regions via discriminability. *CoRR*, abs/1711.06704, 2017. 3
- [11] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic. Efficient neighbourhood consensus networks via submanifold sparse convolutions. *CoRR*, abs/2004.10566, 2020. 3
- [12] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelovic, Akihiko Torii, Tomás Pajdla, and Josef Sivic. Neighbourhood consensus networks. *CoRR*, abs/1810.10510, 2018. 3
- [13] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 3
- [14] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loft: Detector-free local feature matching with transformers. *CoRR*, abs/2104.00680, 2021. 3