Learning Soft Estimator of Keypoint Scale and Orientation with Probabilistic Covariant Loss - Supplementary Material

Pei Yan[1], Yihua Tan[1*], Shengzhou Xiong[1], Yuan Tai[1], and Yansheng Li[2*]
[1]National Key Laboratory of Science and Technology on Multi-spectral Information Processing, School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, China
[2]School of Remote Sensing and Information Engineering, Wuhan University, China
{yanpei, yhtan}@hust.edu.cn, xiongshengzhou@126.com, t_y_@hust.edu.cn, yansheng.li@whu.edu.cn

1. Optimization of the Orientation Estimator

The optimization algorithm of the proposed orientation estimator is in Tab. 1. This algorithm is similar to that of the scale estimator (introduced in Sec. 3 of the main text). Their equations in steps 6 and 7 are slightly different.

2. Proof for the Convergence of the Alternate Optimization Algorithm

The alternate optimization algorithm of the scale estimator is introduced in Sec. 3 of the main text. The algorithm for the orientation estimator is introduced in Tab. 1. In this section, the convergence of the algorithm is first proved for the scale estimator. Then similar proof is introduced for the orientation estimator.

2.1. Proof for the Scale Estimator

We first briefly revisited the loss function and optimization algorithm of the scale estimator, whose details are introduced in Sec. 3 of the main text. The loss function is

$$\min_{\omega_s,\mathbb{S}} -\sum_i \left(\frac{\mathbb{S}[i]}{Z_i} \cdot \sum_{m|\mathrm{IS}_m(i)\neq \mathrm{NaN}} \log\left(\tilde{\mathbb{S}}_m\left[\mathrm{IS}_m(i)\right]\right) \right) \quad (1)$$
$$\text{s.t. } \mathbb{S}[i] \geq 0, i = 1, 2, ..., N_s, \quad \sum_i^{N_s} \mathbb{S}[i] = 1.$$

Here ω_s is the network parameter of the scale estimator. $\mathbb{S}$ is the true distribution of the discretized scales. i is the index of the discretized scales. Z_i is the normalization factor whose definition is below Eq.(6) in the main text. m is the index of the transformed images. $\tilde{\mathbb{S}}_m$ is the predicting confidence of the discretized scales. $\mathrm{IS}_m(\cdot)$ is the function that maps the index in $\mathbb{S}$ to the index in $\tilde{\mathbb{S}}_m$, whose definition is Eq.(6) in the main text.

Table 1. Alternate Optimization Algorithm for the Orientation Estimator.

Input: image dataset D, maximum iterations T, initial value of network parameters ω_o^0, the number of transformed patches M, the largest concerned scale A.
Output: optimized parameter ω_o.
Process:
for t **from** 1 **to** T:
1 Randomly sample a training image I from D;
2 Randomly sample a coordinate c from I as the keypoint;
3 Randomly sample the scaling factors $\triangle S_m \in [A^{-1}, A]$ and rotation angles $\triangle O_m \in [-\pi, \pi]$, $m = 1, 2, ..., M$;
4 Taking c as the center, crop transformed patches X_m with parameters $\triangle S_m$ and $\triangle O_m$, $m = 1, 2, ..., M$;
5 Feed X_m into the estimator whose parameter is ω_o^{t-1}, and obtain confidence vectors $\tilde{\mathbb{O}}_m$, $m = 1, 2, ..., M$;
6 // fix ω_o^{t-1} and optimize $\mathbb{O}$, getting the solution $\mathbb{O}^*$
$i^* = \arg\min_i \sum_m \log\left(\tilde{\mathbb{O}}_m\left[\mathrm{IO}_m(i)\right]\right)$,
and then $\mathbb{O}^*[i] = \begin{cases} 1, i = i^* \\ 0, \text{otherwise} \end{cases}$;
7 // set $\mathbb{O} = \mathbb{O}^*$, and optimize ω_o with the gradient $\frac{\partial L}{\partial \omega_o}$
$\frac{\partial L}{\partial \omega_o} = \frac{\partial}{\partial \omega_o} - \sum_i \mathbb{O}^*[i] \cdot \sum_m \log\left(\tilde{\mathbb{O}}_m\left[\mathrm{IO}_m(i)\right]\right)$,
and update ω_o^{t-1} to ω_o^t with a gradient descent algorithm;
end for
$\omega_o \leftarrow \omega_o^T$;

The objective function in Eq. (1) is denoted as L, namely,

$$L(\omega_s, \mathbb{S}) = -\sum_i \left(\frac{\mathbb{S}[i]}{Z_i} \cdot \sum_{m|\mathrm{IS}_m(i)\neq \mathrm{NaN}} \log\left(\tilde{\mathbb{S}}_m\left[\mathrm{IS}_m(i)\right]\right) \right) \quad (2)$$

L is the cross entropy of two discrete distributions. Therefore, the minimum of L exists.

Eq. (1) is optimized with an iterative algorithm. The network parameter ω_s is randomly initialized as ω_s^0 and is updated as ω_s^t after t times iterations. In the t-th iteration, $\mathbb{S}$ and ω_s are optimized alternately. First, $\mathbb{S}$ is updated with:

$$\begin{aligned}\mathbb{S}^*[i] &= \begin{cases}1, i = i^* \\ 0, \text{otherwise}\end{cases} \\ i^* &= \arg\min_i -\frac{1}{Z_i} \sum_{m|\text{IS}_m(i)\neq\text{NaN}} \log\left(\tilde{\mathbb{S}}_m\left[\text{IS}_m(i)\right]\right).\end{aligned} \tag{3}$$

Here $\tilde{\mathbb{S}}_m$ is obtained with the current network parameter ω_s^{t-1}. Note that $\mathbb{S}^*$ maintains the probability constraint in Eq. (1) because only one element of $\mathbb{S}^*$ is equal to 1 while the others are equal to 0.

Then ω_s^{t-1} is updated to ω_s^t with the gradient descent algorithm. The gradient is computed as:

$$\frac{\partial L}{\partial \omega_s} = \frac{\partial}{\partial \omega_s} - \sum_i \frac{\mathbb{S}^*[i]}{Z_i} \cdot \sum_{m|\text{IS}_m(i)\neq\text{NaN}} \log\left(\tilde{\mathbb{S}}_m\left[\text{IS}_m(i)\right]\right) \tag{4}$$

Considering the minimum of L exists, the above algorithm is convergent if the following two inequalities hold:

$$\begin{aligned}&\forall \mathbb{S} \in \left\{\mathbb{S} \middle| \sum_i^{N_s} \mathbb{S}[i] = 1, \mathbb{S}[i] \geq 0, i = 1, 2, ..., N_s\right\}, \\ &L(\omega_s^{t-1}, \mathbb{S}^*) \leq L(\omega_s^{t-1}, \mathbb{S})\end{aligned} \tag{5}$$

and,

$$L(\omega_s^t, \mathbb{S}^*) \leq L(\omega_s^{t-1}, \mathbb{S}^*). \tag{6}$$

The proof of Eq. (5) is as below.

Proof.

$$\begin{aligned}L(\omega_s^{t-1}, \mathbb{S}) &= -\sum_i \left(\frac{\mathbb{S}[i]}{Z_i} \cdot \sum_{m|\text{IS}_m(i)\neq\text{NaN}} \log\left(\tilde{\mathbb{S}}_m\left[\text{IS}_m(i)\right]\right)\right) \\ &\geq -\sum_i \left(\frac{\mathbb{S}[i]}{Z_{i^*}} \cdot \sum_{m|\text{IS}_m(i^*)\neq\text{NaN}} \log\left(\tilde{\mathbb{S}}_m\left[\text{IS}_m(i^*)\right]\right)\right) \leftarrow (3) \\ &= -\left(\sum_i \mathbb{S}[i]\right) \frac{1}{Z_{i^*}} \cdot \sum_{m|\text{IS}_m(i^*)\neq\text{NaN}} \log\left(\tilde{\mathbb{S}}_m\left[\text{IS}_m(i^*)\right]\right) \\ &= -\frac{1}{Z_{i^*}} \cdot \sum_{m|\text{IS}_m(i^*)\neq\text{NaN}} \log\left(\tilde{\mathbb{S}}_m\left[\text{IS}_m(i^*)\right]\right) \leftarrow \sum_i \mathbb{S}[i] = 1 \\ &= -\sum_i \left(\frac{\mathbb{S}^*[i]}{Z_i} \cdot \sum_{m|\text{IS}_m(i)\neq\text{NaN}} \log\left(\tilde{\mathbb{S}}_m\left[\text{IS}_m(i)\right]\right)\right) \leftarrow (3) \\ &= L(\omega_s^{t-1}, \mathbb{S}^*)\end{aligned}$$

□

The proof of Eq. (6) is as below.

Proof.

As introduced in Eq. (4), ω_s^t is obtained by optimizing $L(\omega, \mathbb{S}^*)$ with the gradient descent algorithm while the start point is ω_s^{t-1}. So ω_s^t is the nearby local minimum point of ω_s^{t-1}. Therefore, with the premise for appropriate learning rate, the inequality $L(\omega_s^t, \mathbb{S}^*) \leq L(\omega_s^{t-1}, \mathbb{S}^*))$ holds. □

In our implementation, step 7 of the alternate optimization algorithm updates ω_s only once to ensure efficiency. Experimental results demonstrate that the S3Esti model is optimized appropriately with this strategy.

2.2. Similar Proof for the Orientation Estimator

According to Eq. (9) in the main text, the objective function of the orientation estimator can be denoted as:

$$L(\omega_o, \mathbb{S}) = -\sum_i \left(\mathbb{O}[i] \cdot \sum_m \log\left(\tilde{\mathbb{O}}_m\left[\text{IO}_m(i)\right]\right)\right) \tag{7}$$

Similar to the proofs in Sec. 2.2, the following two inequalities hold:

$$\begin{aligned}&\forall \mathbb{O} \in \left\{\mathbb{O} \middle| \sum_i^{N_o} \mathbb{O}[i] = 1, \mathbb{O}[i] \geq 0, i = 1, 2, ..., N_o\right\}, \\ &L(\omega_o^{t-1}, \mathbb{O}^*) \leq L(\omega_o^{t-1}, \mathbb{O})\end{aligned} \tag{8}$$

and,

$$L(\omega_o^t, \mathbb{O}^*) \leq L(\omega_o^{t-1}, \mathbb{O}^*). \tag{9}$$

The definitions of O^* and $\text{IO}_m(\cdot)$ are introduced in Sec. 3 of the main text. According to Eqs. (8) and (9), the algorithm of the orientation estimator is also convergent.

3. Results on Relative Pose Estimation Task

Metrics and evaluation configurations. Two metrics named *Prec.* [13] and *AUC* [17] are used to evaluate the pose estimation accuracy. In the computation of Prec., all keypoints in one image are re-projected to the other image with the estimated pose, and a keypoint is considered to be correctly matched if its normalized distance to the ground-truth epipolar line is less than 1e-4. The metric AUC first generates a curve by classifying each estimated pose as accurate or not. Then AUC computes the area under this curve up to a maximum threshold of 5°, 10° or 20°.

This experiment uses three keypoint extraction models, namely SuperPoint [4], POP [16], and HAN_HN [10], as the baseline models. SIFT [8], CovDet [6] and the proposed S3Esti are used as the scale estimators. Note that the images in the MegaDepth dataset are generally captured with the upright cameras, and their orientation changes are relatively small. Therefore, the orientation estimators of these methods are not used in this experiment. DRC-Net [7] is also

used as a comparison method. The combination of DRC-Net and an estimator is not efficient because DRC-Net is a dense matching method without keypoint localization. So we do not combine DRC-Net with the estimators in this experiment.

Results. The relative pose estimation results of different approaches are shown in Tab. 2. The combinations of S3Esti and the existing keypoint extraction models outperform other competitors. Note that the wide-baseline image pair is a common situation in the MegaDepth dataset because of the significant viewpoint changes. The results in Tab. 2 indicate that S3Esti can improve the matching accuracy of wide-baseline pairs. This superiority should be helpful to register more images and improve the triangulation precision in 3D reconstruction. Overall, this experiment demonstrates that S3Esti can improve the pose estimation accuracy, and explains how S3Esti improves the performance of 3D reconstruction.

Table 2. Pose Estimation Results on MegaDepth validation set.

	Prec. (%)	Pose estimation AUC (%) @5°	@10°	@20°
DRC-Net	w/o keypoint	27.01	42.96	58.31
SuperPoint	59.19	22.06	37.44	52.55
SuperPoint+SIFT	62.86	23.64	39.63	55.40
SuperPoint+CovDet	59.67	23.40	39.39	55.23
SuperPoint+S3Esti	**64.11**	**24.69**	**40.85**	**56.75**
POP	54.33	25.85	42.02	56.75
POP+SIFT	56.71	27.10	43.94	59.13
POP+CovDet	55.32	27.08	43.58	58.77
POP+S3Esti	**58.61**	**29.57**	**46.09**	**61.49**
HAN_HN	55.21	23.47	40.12	55.80
HAN_HN+SIFT	**62.82**	28.62	45.78	61.71
HAN_HN+CovDet	56.53	27.77	44.16	59.74
HAN_HN+S3Esti	62.50	**31.09**	**48.42**	**63.97**

4. Deriving the Scale and Orientation Changes from the Homography Transformation

In the experiment, it is not straightforward to compute the estimation errors of the scale and orientation for a single patch because the ground-truth scale and orientation are hardly determined. Instead, we evaluate the estimation errors by jointly considering a pair of patches because it is easier to obtain the ground truth of the relative changes of scales and orientations. Note that the image pairs in HPatches [1] generally involve homography transformations. Therefore, we introduce how to approximately derive the ground-truth changes of the scale and orientation from the ground-truth homography matrices.

Fig. 1 shows the process to compute the scale and orientation changes of every keypoint after a homography transformation. With this approach, different keypoints in the same images may have different scale and orientation changes. This phenomenon is consistent with the nonuniform characteristic of the homography transformation.

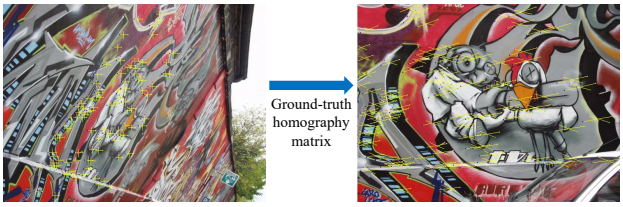

Figure 1. The scale and orientation changes of every keypoint after a homography transformation. The process to compute these changes consists of four steps. **First**, the keypoints in the first image are extracted. Then the horizontal and vertical lines with twenty-pixel length are drawn across every keypoint. Each pair of the horizontal and vertical lines is denoted as a "+". **Second**, every "+" is transformed to the second image with the ground-truth homography matrix. **Third**, for any keypoint, the scale change is the quotient that its size in the second image is divided by its size in the first image. Here the size of a keypoint is the product of the lengths of two lines in the "+". **Fourth**, for any keypoint, the orientation change is the average rotation angle of the two lines in the "+".

In Sec. 4.2 of the main text, two subsets HPatches-view-small and HPatches-view-large are divided according to the scale and orientation changes of the center point in the image. The computation process of the changes is the same as that in Fig. 1. Here the center point is selected to represent the entire image because its average distance to all the keypoints is relatively small.

Another related detail is how to evaluate the estimation error of a soft estimator. Note that an estimator predicting multiple scales and orientations is termed as a soft estimator in this paper. Fig. 2 demonstrates the process to find the nearest matching patch in the second image for a given patch in the first image. The scale/orientation pair corresponding to the highest similarity will be evaluated with the estimation error metrics.

5. More Results on the Estimation Error Metrics

Tab. 2 of the main text shows the overall estimation errors of different estimators. In this section, more details are provided for the estimation errors. Figs. 3 and 4 display the distributions of the estimation errors with the cumulative frequency histograms, involving both the HP-view-small and HP-view-large datasets. Every curve corresponds to a combination of a keypoint extraction model and an estimator. Here six keypoint extraction models are consid-

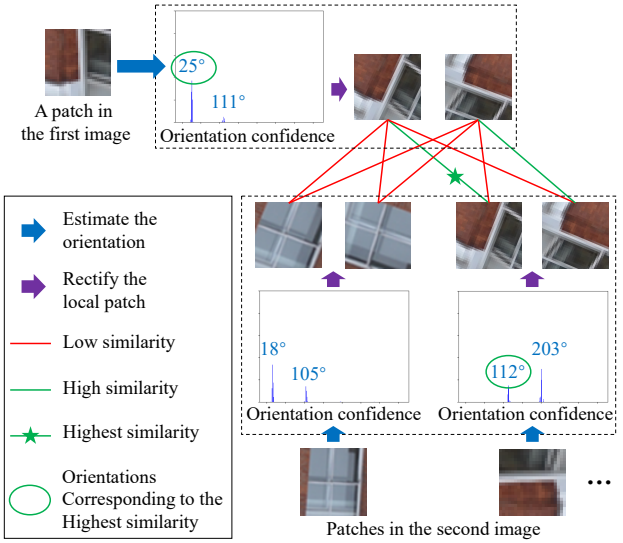

Figure 2. The process of finding the nearest matching patch in the second image for a given patch in the first image. This figure takes the orientation estimation as an example. The pair of orientations corresponding to the highest similarity will be evaluated with the estimation error metrics.

ered, namely, HAN_HN (HesAffNet+HardNet [9]), SuperPoint [4], Key.Net_HN (Key.Net [5]+HardNet), R2D2 [12], POP [16] and LFNet [11]. Every keypoint extraction model is combined with at least five scale&orientation estimators, namely SURF [2], SIFT [8], CovDet [6], and the proposed S3Esti and S3Esti-S. "No_esti" represents that no estimator is performed while the scale and orientation are assigned as 1.0 and 0° respectively. Note that LFNet [11] and HesAffNet [10] have their own estimators, whose results are also shown in Figs. 3 and 4. Besides, HesAffNet [10] does not perform the orientation estimation itself. Another model is designed in the same paper solely for the orientation estimation[1]. The results with the separate orientation estimator are annotated as "HesAffNet-O". The results without this orientation estimator are annotated as "HesAffNet".

A better estimator should provide a larger number of accurate estimations of the scale and orientation. Overall, the proposed S3Esti reliably outperforms other estimators on the HP-view-large dataset because the curves of S3Esti are generally on the upper and left of the others. Furthermore, S3Esti is competitive on the HP-view-small dataset. These results are consistent with the improvements on the image matching task (Sec. 4.4 of the main text and Sec. 6 of this supplementary), in which S3Esti improves the matching accuracy on HP-view-large while maintaining the accuracy on HP-view-small.

[1]We sincerely thank our reviewer for pointing out this detail.

6. More Results on Image Matching Task

Fig.5 of the main text shows the superiority of POP+S3Esti and HAN_HN+S3Esti. In this section, the combinations of S3Esti and more keypoint extraction models are evaluated. Figs. 5 and 6 show the matching accuracy for six keypoint extraction models on both the HP-view-small and HP-view-large datasets. Every curve corresponds to a combination of a keypoint extraction model and an estimator. The notations of the keypoint extraction models and estimators are the same as those of Figs. 3 and 4, which are introduced in Sec. 5.

Overall, the proposed S3Esti outperforms other estimators on HP-view-large. Its accuracy improvements relative to the baselines are overall more than 50%. Furthermore, S3Esti can maintain the accuracy of the baseline model on HP-view-small, generally slightly better than other estimators.

7. Re-implementation of CovDet

In Secs 4.3 and 4.4 of the main text, CovDet [6] is evaluated as a comparison model. This estimator is our re-implementing model. We re-implement CovDet for two reasons. First, the original CovDet only estimates the keypoint orientation but does not concern the keypoint scale. Therefore, the scale estimator needs to be implemented to complete the comparison. Second, the network architecture of the original CovDet is simpler than ours. So we re-implement it with the backbone architecture of S3Esti to ensure fairness.

The details of our implementation for the scale estimator of CovDet are as below. Following the configuration of the orientation estimator of CovDet, the scale estimator is implemented as a regression model, which takes a patch as input and outputs a scalar to represent the scale. The scale is truncated with the range $[\frac{1}{A}, A]$. Here $A = 9$ following the configuration in the main text. Then the scale is represented as a scaling transformation matrix, which can be optimized with Eq. (15) in CovDet [6].

8. Ablation Experiment

Different ablation models are evaluated in this section. The first experiment is designed to verify the contributions of different factors in S3Esti. The results on the HP-view-large dataset are shown in Fig. 7. POP [16] and HAN_HN [9, 10] are used as two baseline models. "xx+S3Esti" indicates a model integrated with the full S3Esti. "xx+S3Esti-w/o s" removes the scale estimator from "xx+S3Esti", while "xx+S3Esti-w/o o" removes the orientation estimator. "xx+S3Esti-S" is a hard estimator by setting $K = 1$, as introduced in the Sec. 4.1 of the main text. K is the maximal number of the scales and orientations allowed to

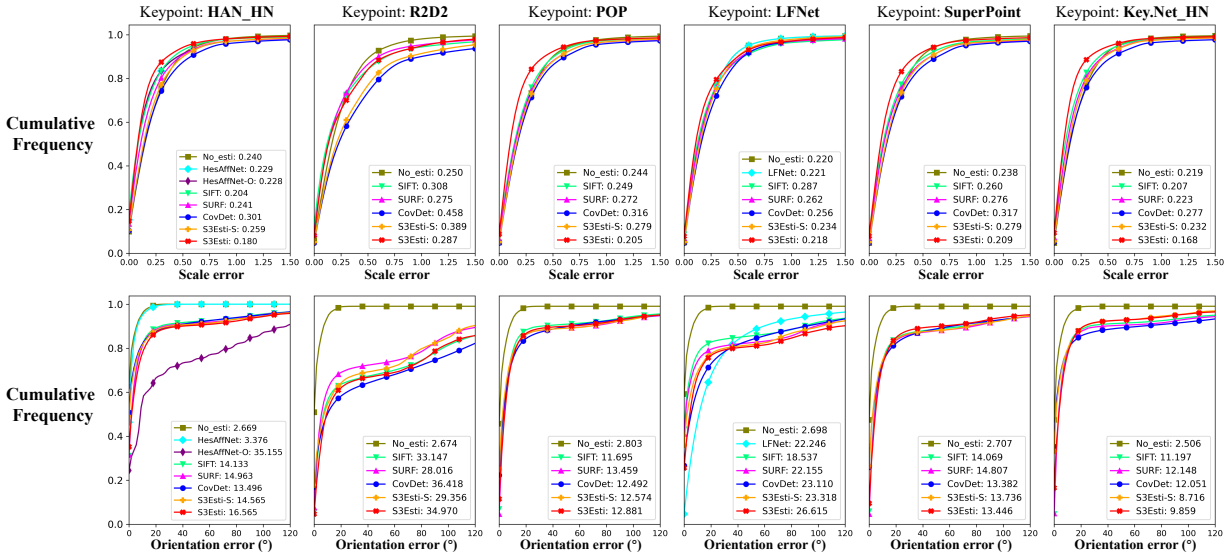

Figure 3. The cumulative frequency histograms of the estimation error on the **HP-view-small** dataset, evaluated for different combinations of the keypoint extraction model and estimator. Note that the legends only show the names of estimators, while the names of keypoint extraction models are marked on the top of each column. The mean values of the errors are also shown in the legends. The curves of a better estimator should be on the upper and left of the others. The proposed S3Esti outperforms other estimators in the scale estimation, and is competitive in the orientation estimation. Note that the rotation changes in HP-view-small are slight. So the lowest error of orientation is obtained by No_esti, which is a fictitious model predicting the orientation as the constant zero.

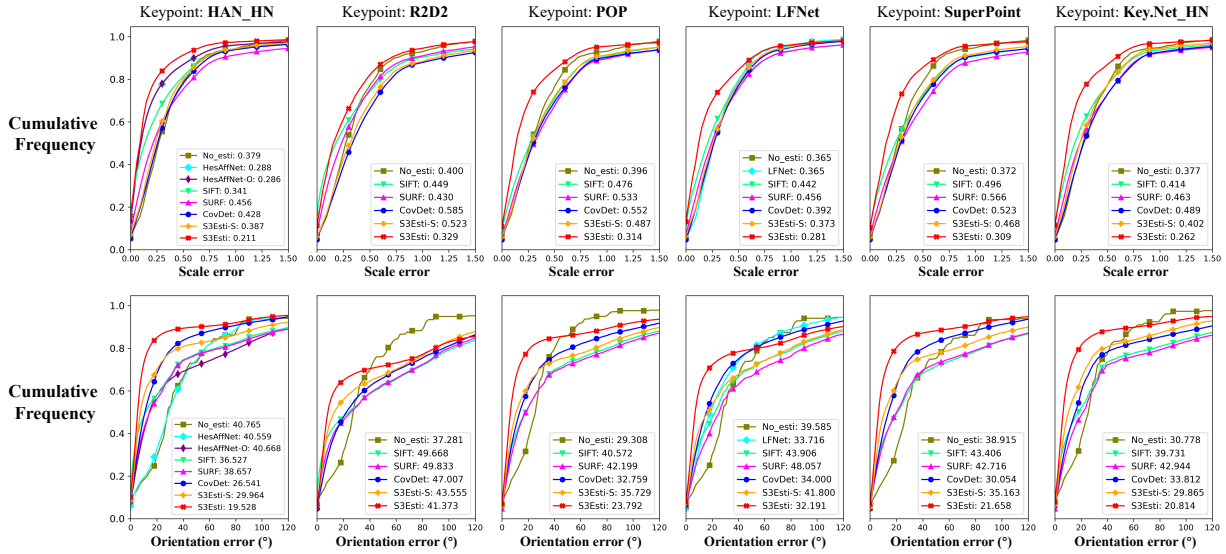

Figure 4. The cumulative frequency histograms of the estimation error on the **HP-view-large** dataset, evaluated for different combinations of the keypoint extraction model and estimator. Note that the legends only show the names of estimators, while the names of keypoint extraction models are marked on the top of each column. The mean values of the errors are also shown in the legends. The curves of a better estimator should be on the upper and left of the others. The proposed S3Esti outperforms other estimators in all the combinations. No_esti is inappropriate for this dataset because HP-view-large involves much more significant geometric changes compared with HP-view-small.

be kept. "xx+S3Esti-Two" sets $K = 2$, which is different from "xx+S3Esti" setting $K = 3$.

Overall, POP+S3Esti and HAN_HN+S3Esti outperform other ablation models, which demonstrates that both scale and orientation estimators contribute to the image accuracy. The fact that "xx+S3Esti" is superior to "xx+S3Esti-S" verifies that the soft predictions are more robust than the hard predictions. The accuracy of "xx+S3Esti-Two" is only

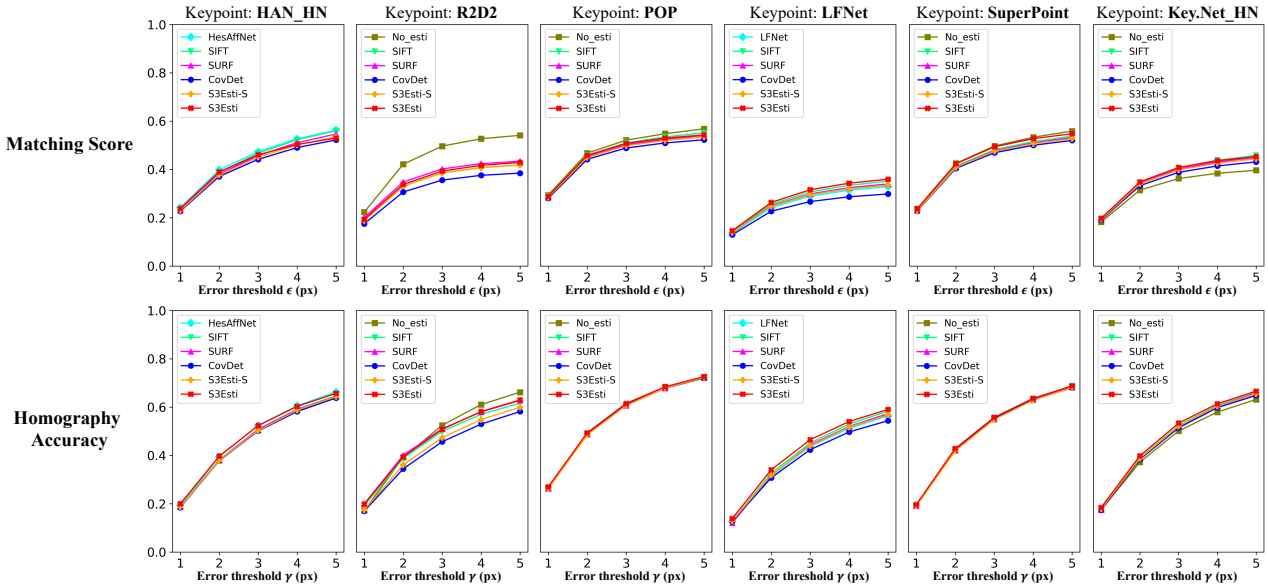

Figure 5. MScore and HA results on the **HP-view-small** dataset, evaluated for different combinations of the keypoint extraction model and estimator. Note that the legends only show the names of estimators, while the names of keypoint extraction models are marked on the top of each column. Generally, the proposed S3Esti is slightly better than other estimators. No_esti, the model without any estimator, is very competitive on HP-view-small because this dataset only involves slightly geometric changes.

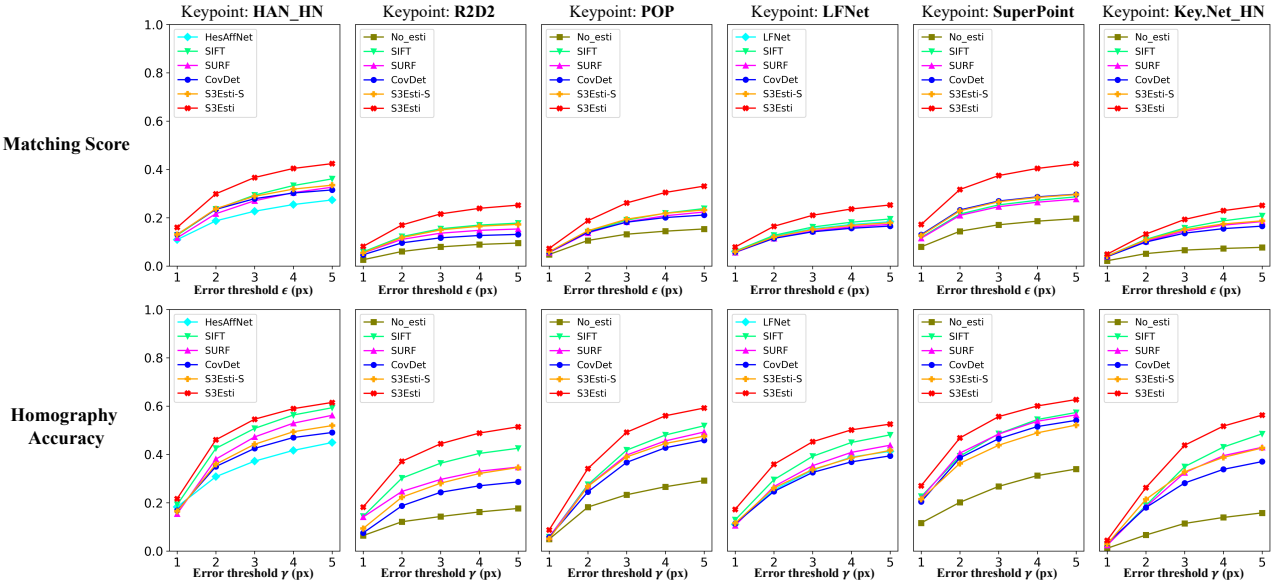

Figure 6. MScore and HA results on the **HP-view-large** dataset, evaluated for different combinations of the keypoint extraction model and estimator. Note that the legends only show the names of estimators, while the names of keypoint extraction models are marked on the top of each column. No_esti is inappropriate for HP-view-large that involves significant geometric changes. The proposed S3Esti outperforms other estimators in all combinations. Its improvements relative to the original keypoint extraction models are overall more than 50%.

slightly lower than that of "xx+S3Esti", which demonstrates that keeping at most two (rather than three) scales and orientations can also give effective improvements.

The second experiment is designed to verify that lowering the estimation errors of the scale and orientation can improve the matching accuracy. The results on the HP-view-large dataset are shown in Fig. 8. These results demonstrate that the lower estimation error of the scale and orientation generally leads to higher matching accuracy. The proposed S3Esti has lower estimation errors than SURF [2]

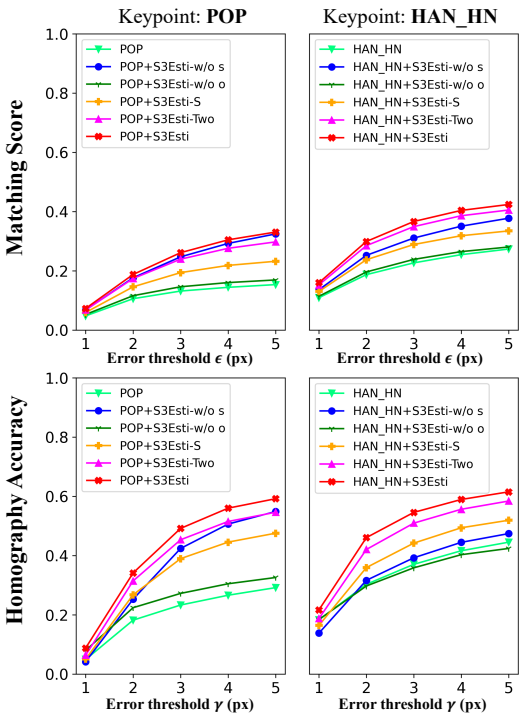

Figure 7. Results of different ablation models on the **HP-view-large** dataset. Overall, POP+S3Esti and HAN_HN+S3Esti outperform other ablation models.

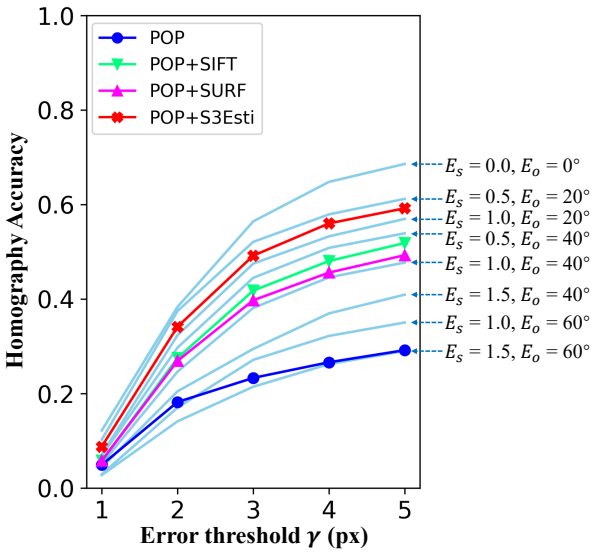

Figure 8. Results of the models corresponding to different levels of estimation errors. To analyze the "upper bound" of the improvement provided by the scale and orientation estimation, a perfect "estimator" is constructed to output the scale and orientation consistent with the image pairs' ground-truth homography matrices. The HA curve of the "perfect estimator" is indicated with "$E_s = 0.0$, $E_o = 0°$", where E_s and E_o represent the estimation errors of the scale and orientation. Then different levels of random errors are added to the perfect "estimator", whose HA results correspond to the curves from "$E_s = 0.5$, $E_o = 20°$" to "$E_s = 1.5$, $E_o = 60°$".

and SIFT [8], as shown in Sec. 4.3 of the main text. This superiority should be the reason why S3Esti achieves higher matching accuracy in Fig. 8.

The third experiment compares two different model architectures. As introduced in the main text, S3Esti implements the scale and orientation estimators as two separate networks. Another variant named S3Esti_Joint is constructed in this section. S3Esti_Joint implements the scale and orientation estimators with a shared backbone and two output heads. The structures of the backbone and output heads are identical to those of S3Esti.

As introduced in Sec. 4.1 of the main text, S3Esti converges well with directly performing the proposed alternate optimization algorithm. However, S3Esti_Joint usually converges to a poor local minimum with the same algorithm. The reason may be that the estimations of scale and orientation are two problems with low relevance. Therefore, the optimization directions of the two problems may be irreconcilable with a poor initialization of network parameters. We carefully design a training strategy to overcome this difficulty, consisting of two stages:

(1) Preliminarily optimize S3Esti_Joint with only the loss function of orientation estimation. In this stage, the loss of scale estimation is ignored to lower the optimization difficulty. The orientation estimation error (O. Err.) is computed for every mini-batch during training. This stage is finished once O. Err. on the current mini-batch is smaller than 75°, and then the second stage is started. This O. Err. threshold is appropriate for stable convergence in our experiment, but a smaller threshold may also work. This stage is generally finished within one epoch on the MS COCO 2014 training set.

(2) Optimize S3Esti_Joint with the complete loss function of scale&orientation estimation. The optimization configurations of this stage are identical to those of S3Esti. This stage generally finished after 30 epochs on the MS COCO 2014 training set, similar to S3Esti.

Intuitively, the first training stage provides a better initialization corresponding to smaller orientation estimation error. In our experiment, this strategy makes S3Esti_Joint converge well. Figs. 9 and 10 show the estimation errors and matching accuracy of POP+S3Esti_Joint and HAN_HN+S3Esti_Joint. Overall, S3Esti_Joint achieves a similar accuracy compared with S3Esti.

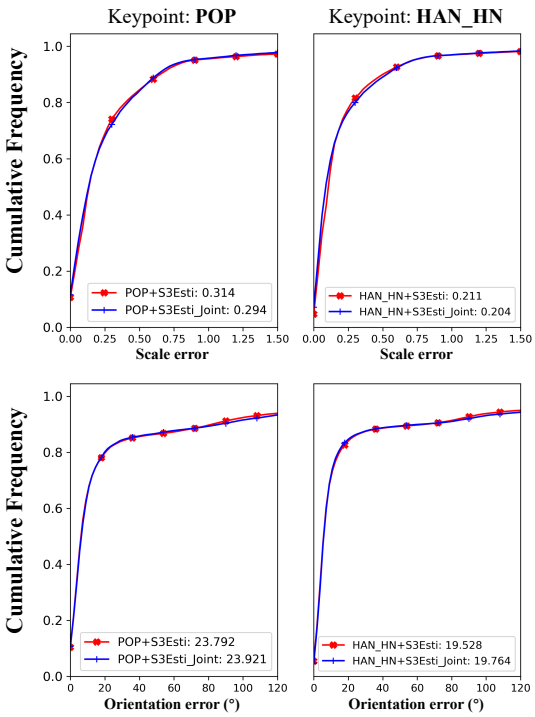

Figure 9. The cumulative frequency histograms of the estimation error on the **HP-view-large** dataset, evaluated for the S3Esti and S3Esti_Joint estimators. The mean values of the errors are also shown in the legends. S3Esti and S3Esti_Joint obtain almost identical estimation errors.

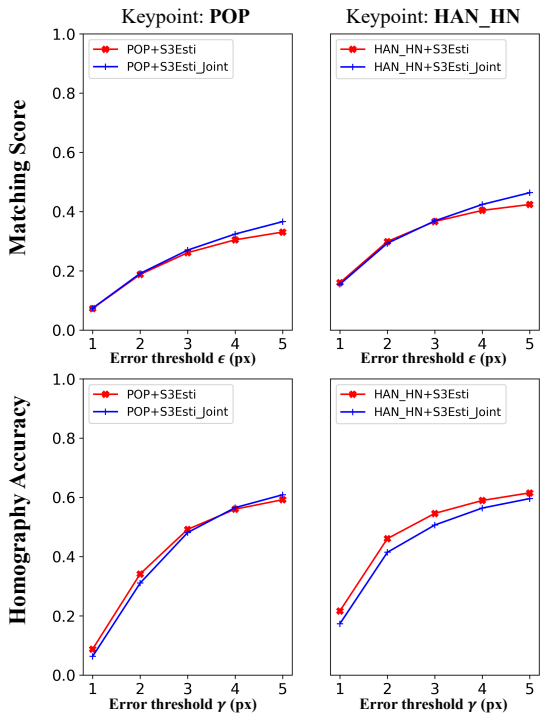

Figure 10. MScore and HA results on the **HP-view-large** dataset, evaluated for the S3Esti and S3Esti_Joint estimators. Overall, S3Esti and S3Esti_Joint achieve similar matching accuracy. Therefore, it is feasible to jointly train the scale and orientation estimators of S3Esti.

9. Comparison with the AEU Model

AEU is the abbreviation of Affine Estimation Unit [3]. Strictly speaking, AEU is not an estimator of the scale or orientation. The prediction of AEU is the relative scaling and rotation changes between two patches. A characteristic of AEU is that its computation complexity is $O(n^2)$, where n is the number of keypoints. Note that the complexity of a scale or orientation estimator is $O(n)$.

The results of AEU[2] and S3Esti are shown in Fig. 11. AEU is slightly superior to S3Esti. However, the inference speed of AEU is much slower than S3Esti, which is discussed in Sec. 10.

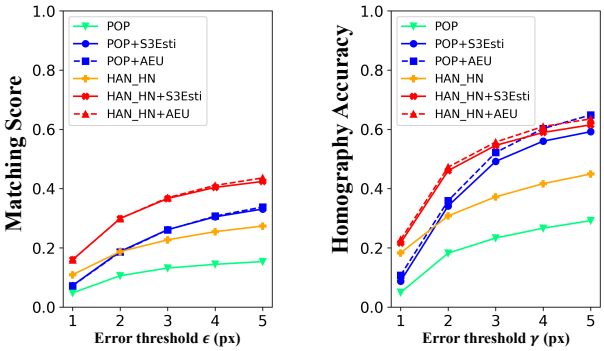

Figure 11. Comparison between the AEU [3] model and the proposed S3Esti. Overall, AEU is slightly superior to S3Esti. However, the inference speed of AEU is much slower than S3Esti, which is discussed in Sec. 10.

10. Inference Speed

The time consumptions of S3Esti and some comparison estimators are shown in Tab. 3. All the four estimators are combined with the HesAffNet detector [10] and HardNet descriptor [9]. The first part of the results is evaluated on the HPatches [1] dataset, in which an average of 1000 keypoints are extracted in an image (denoted as #Points=1000.0). The second part of results is evaluated on the Madrid Metropolis sequence of the ETH [14] dataset, in which an average of 4796.4 keypoints are extracted in an image (denoted as #Points=4796.4). "Esti&Rectify" represents the time consumption of the scale&orientation estimation and patch rectification. "Desc" represents the time consumption of computing the description vectors for the rectified patches. All

[2]Note that the official code of AEU [3] has not been released when this paper was submitted. Therefore, we re-implement AEU based on the backbone architecture of S3Esti.

the time consumptions are evaluated with an Intel Core i9-9820X CPU and a GeForce RTX 2080 Ti GPU. The rectifying processes are all implemented with the *grid_sample()* function in Pytorch, which is accelerated with the GPU operation.

AEU [3] is slow because of its $O(n^2)$ complexity, while SIFT, CovDet and S3Esti are much faster with the $O(n)$ complexity. Although S3Esti and CovDet use the same backbone architecture, S3Esti is slower than CovDet because S3Esti predicts multiple scales&orientations and involves multiple rectified patches. Still, the speed of S3Esti is acceptable for many tasks, such as offline image registration and 3D reconstruction.

Overall, S3Esti can reliably improve the matching accuracy compared with the existing estimators while maintaining an acceptable speed for offline tasks.

Table 3. Time Consumption (Second) of Different Estimators. Every estimator is combined with the HesAffNet [10] keypoint detector and the HardNet descriptor [9] to finish this evaluation.

	#Points=1000.0		#Points=4796.4	
	Esti&Rectify	Desc	Esti&Rectify	Desc
SIFT	**0.047**	0.0005	**0.26**	0.0018
AEU	63.51	0.28	1390.23	5.88
CovDet	0.10	**0.0003**	0.46	**0.0013**
S3Esti	0.21	0.0011	1.14	0.0032

11. More Visualization Results

More visualization results are shown in this section. As a supplement for Fig. 5 in the main text, Fig. 12 shows some matching results for the combination of SuperPoint [4] and S3Esti, termed as SuperPoint+S3Esti. The results demonstrate that SuperPoint+S3Esti reliably improves the matching accuracy under significantly geometric changes.

As introduced in Sec. 4.5 of the main text, S3Esti can decrease the reprojection error of 3D reconstruction. Fig. 13 visualizes how the reprojection error impacts the reconstruction result. The point cloud with S3Esti has clear details and fewer outliers.

Fig. 14 shows more intermediate results of S3Esti. Even though some new patterns appear after a viewpoint change, S3Esti can provide similar rectified patches by predicting multiple scales and orientations.

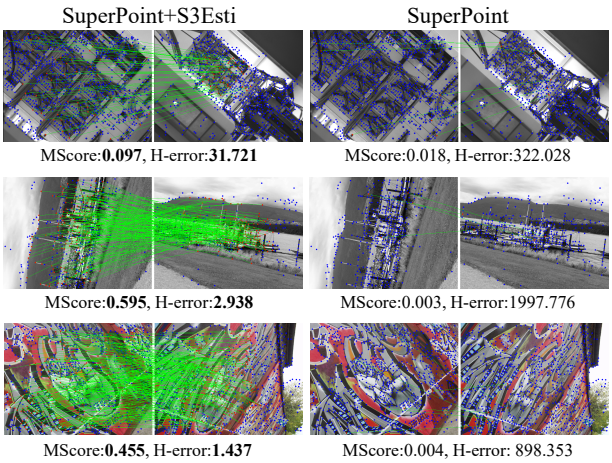

Figure 12. The visualization for the keypoint matching results with or without the proposed S3Esti. MScore denotes the matching score with the error threshold $\epsilon = 3$, while H-error indicates the homography estimation error. Both two metrics are introduced in Sec. 4.4 of the main text. The results in the third row indicate that S3Esti also benefits the image matching under more complex homography transformations.

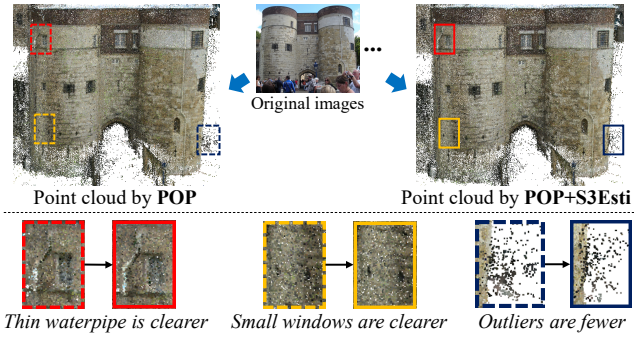

Figure 13. The visualization of the 3D reconstruction results with or without the proposed S3Esti. With lower reprojection error, S3Esti makes small objects clearer and decreases the outliers.

References

[1] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 5173–5182, 2017. 3, 8

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the European conference on computer vision*, pages 404–417. Springer, 2006. 4, 6

[3] Ji Dai, Shiwei Jin, Junkang Zhang, and Truong Q. Nguyen. Boosting feature matching accuracy with pairwise affine estimation. *IEEE Trans. Image Process.*, 29:8278–8291, 2020. 8, 9

[4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018. 2, 4, 9

[5] Axel Barroso Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key.net: Keypoint detection by handcrafted and learned CNN filters. In *Proceedings of the Inter-*

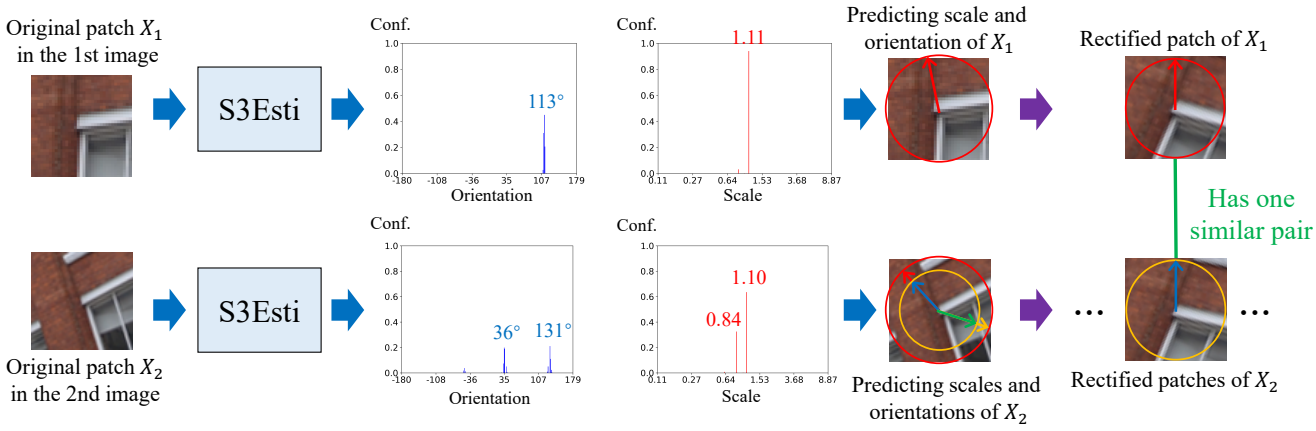

Figure 14. The visualization of more intermediate results. The notations in this figure are consistent with those in Fig. 1 in the main text.

national Conference on Computer Vision, pages 5835–5843. IEEE, 2019. 4

[6] Karel Lenc and Andrea Vedaldi. Learning covariant feature detectors. In *Proceedings of the European conference on computer vision*, volume 9915, pages 100–117, 2016. 2, 4

[7] Xinghui Li, Kai Han, Shuda Li, and Victor Prisacariu. Dual-resolution correspondence networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 17346–17357, 2020. 2

[8] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 2, 4, 7

[9] Anastasya Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 4826–4837, 2017. 4, 8, 9

[10] Dmytro Mishkin, Filip Radenović, and Jiři Matas. Repeatability is not enough: Learning affine regions via discriminability. In *Proceedings of the European Conference on Computer Vision*, 2018. 2, 4, 8, 9

[11] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: learning local features from images. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 6237–6247, 2018. 4

[12] Jérôme Revaud, César Roberto de Souza, Martin Humenberger, and Philippe Weinzaepfel. R2D2: reliable and repeatable detector and descriptor. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 12405–12415, 2019. 4

[13] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 4937–4946, 2020. 2

[14] Johannes L. Schönberger, Hans Hardmeier, Torsten Sattler, and Marc Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 6959–6968, 2017. 8

[15] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 8922–8931, 2021.

[16] Pei Yan, Yihua Tan, Yuan Tai, Dongrui Wu, Hanbin Luo, and Xiaolong Hao. Unsupervised learning framework for interest point detection and description via properties optimization. *Pattern Recognition*, 112:107808, 2021. 2, 4

[17] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 2666–2674, 2018. 2