XRefine: Attention-Guided Keypoint Match Refinement

*Jan Fabian Schmid *Annika Hagemann
Bosch Research
{JanFabian.Schmid, Annika.Hagemann}@de.bosch.com

Abstract

Sparse keypoint matching is crucial for 3D vision tasks, yet current keypoint detectors often produce spatially inaccurate matches. Existing refinement methods mitigate this issue through alignment of matched keypoint locations, but they are typically detector-specific, requiring retraining for each keypoint detector. We introduce XRefine*, a novel, detector-agnostic approach for sub-pixel keypoint refinement that operates solely on image patches centered at matched keypoints. Our cross-attention-based architecture learns to predict refined keypoint coordinates without relying on internal detector representations, enabling generalization across detectors. Furthermore, XRefine can be extended to handle multi-view feature tracks. Experiments on MegaDepth, KITTI, and ScanNet demonstrate that the approach consistently improves geometric estimation accuracy, achieving superior performance compared to existing refinement methods while maintaining runtime efficiency. Our code and trained models can be found at https://github.com/boschresearch/xrefine.*

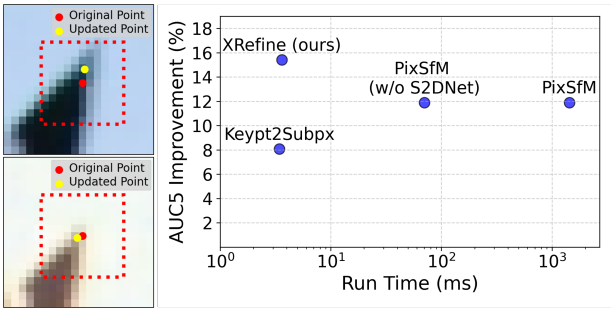

Figure 1. **Attention-guided match refinement efficiently improves relative pose estimation**. **Left:** Exemplary matched SuperPoint [5] keypoints on MegaDepth [17]. The input to our model are the 11×11 patches within the red dotted lines. The refined keypoints of our model are presented as yellow dots. **Right:** Runtime and pose estimation improvement on MegaDepth (measured as relative increase in AUC5) of match refinement approaches averaged over five feature extractors: DeDoDe [8], SIFT [20], SuperPoint [5], and XFeat [25]. We compare our generalizing model to Keypt2Subpx [14] and the match refinement solution of PixSfM [18]. PixSfM extracts dense S2DNet [11] embeddings for feature-metric refinement. Depending on the use case this might be done exclusively for the refinement. Accordingly, we show the runtime of PixSfM with and without S2DNet inference.

1. Introduction

Extracting and matching sparse keypoints remain central to 3D computer vision systems, including structure-from-motion, visual localization, and SLAM. Despite the growing adoption of end-to-end, fully learned pipelines [16, 32, 33], many practical systems - particularly those with memory and runtime constraints - still depend on explicitly detected and matched keypoints. Sparse approaches offer clear benefits: they are lightweight, interpretable, and thus well-suited if dense inference is unnecessary or infeasible.

The accuracy of keypoint-based systems is crucially influenced by the spatial accuracy of matched keypoints, *i.e.*, how accurately the keypoints reflect the same physical 3D point geometrically (see Figs. 1 and 2). However, recent work [14] shows that even state-of-the-art detectors suffer from inaccurate keypoint matches, decreasing geometric accuracy in downstream tasks. This limitation emerges naturally in keypoint detectors that only process each image separately, rendering the detection of keypoints at the exact same position in both images inherently difficult.

To address this limitation, recent refinement networks [14, 25] adjust matched keypoint locations by simultaneously considering information of both images. Given a pair of matched keypoints, these models predict keypoint displacements using either keypoint descriptors [25] or scores and surrounding image patches [14]. While these refinements improve the accuracy of downstream tasks like relative pose estimation, they require access to internal feature extractor representations (descriptors and keypoint scores). This necessity requires retraining for each detector architecture, limiting their generality and practical deployment.

We expand on this research by proposing a novel,

*Indicates equal contribution.

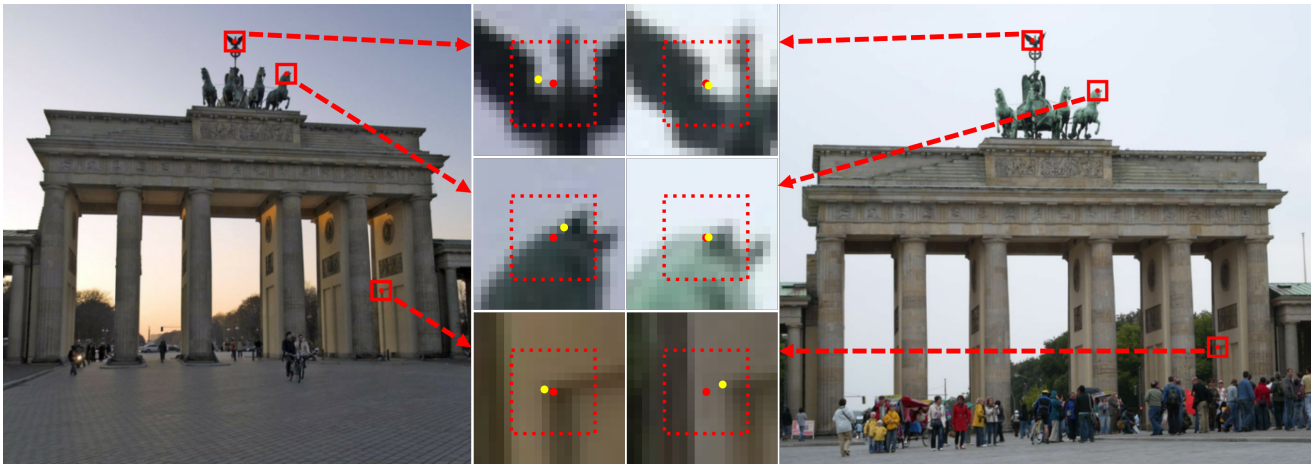

Figure 2. Example match refinements from our model on MegaDepth [17] for SuperPoint [5] keypoints. The original keypoints are shown as red dots. In the magnified patches, the refined keypoints are shown as yellow dots. While the presented patches in this figure have a size of 21×21 pixels, the refinement model receives only the 11×11 area framed by the red dotted rectangle as input.

detector-agnostic method for sub-pixel keypoint refinement called XRefine. Unlike the refinement networks in [14, 25], XRefine operates exclusively on image patches centered at matched keypoints *without* requiring descriptors or keypoint scores. Thus, our model only needs to be trained once and generalizes across a wide range of classical (e.g., SIFT [20]) and learned (e.g., SuperPoint [5], ALIKED [34]) detectors without requiring per-detector adaptation.

Inferring matched keypoint displacements solely from image patches requires information from both patches, which we realize using a cross-attention layer. Unlike existing image-patch-based refinement methods like PixSfM [18], the proposed method does not rely on costly feature-metric optimization, but infers the refinement in a single forward pass. This makes XRefine lightweight and applicable on common edge AI chips.

Finally, we also propose a generalization of the approach from two-view matches to n-view feature tracks. This enables using the approach in SfM pipelines, as showcased for 3D point cloud triangulation on the ETH3D dataset [28].

We demonstrate that our approach consistently improves the accuracy of geometric estimation tasks across standard benchmarks such as MegaDepth [17], KITTI [10], and ScanNet [4], achieving higher pose accuracy than existing refinement approaches (see Fig. 1).

In summary, our contributions are:

1. A cross-attention-based architecture for sub-pixel keypoint refinement that operates on image patches alone.
2. A detector-agnostic training scheme achieving generalization across a wide range of keypoint detectors.
3. A model variant for consistent multi-view refinement.
4. Superior performance across diverse datasets and feature extractors, without sacrificing runtime efficiency.

2. Related work

Sparse local feature extraction Tasks like camera pose estimation and calibration depend on the availability of point correspondences between images. Sparse local features are an efficient tool for determining correspondences:

1. For each image, individually extract a set of keypoints along with corresponding score values and descriptors.
2. Select the best keypoints per image based on their score.
3. Identify potentially corresponding keypoints between images as those matched based on descriptor similarity.

Classical hand-crafted feature extraction, such as SIFT [20], detects keypoints as intensity extrema using a Difference of Gaussian pyramid. More recently, learning-based approaches started to outperform the classical approaches. DeTone *et al*. introduced SuperPoint [5], a fully-convolutional, single-forward-pass approach, leveraging Homographic Adaptation for self-supervised pre-training, which was later extended to fully self-supervised training in UnsuperPoint [3] and KP2D [30]. Other methods focus on learning refined metrics, such as R2D2 [26], which distinguishes descriptor reliability and keypoint repeatability, and DISK [31], which uses reinforcement learning to train the extractor end-to-end. Some feature extractors like DeDoDe [8] and DeDoDev2 [7] aim for high performance, incorporating with DINOv2 [23] a large vision transformer as encoder. Efficiency-focused methods use lightweight CNN architectures, as in XFeat [25], or compute descriptors only at keypoint positions, like ALIKED [34]. While the overall performance of local feature extraction has improved over time, recent work [14] has shown that the spatial accuracy of keypoints still limits the accuracy of geometric downstream tasks (see Fig. 3).

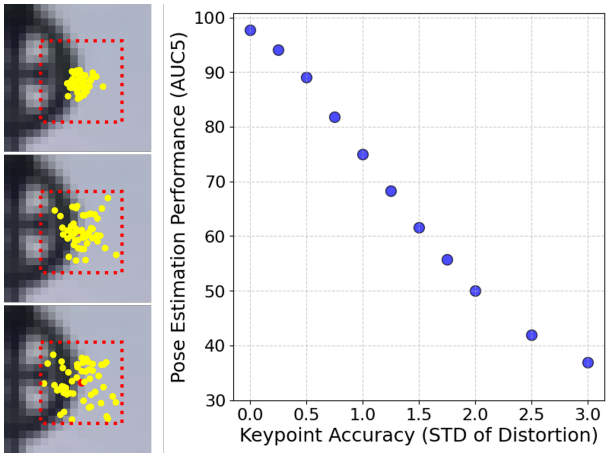

Figure 3. Effect of inaccurate keypoint locations on the accuracy of relative pose estimation. **Left:** A patch of size 21 × 21 with a true keypoint shown as red dot and yellow dots representing sampled distortions to the keypoint (from top to bottom with a standard deviation of 1, 2, and 3 pixels). The red dotted rectangle shows the 11 × 11 center area of the patch. **Right:** A graph illustrating the measured AUC5 pose estimation performance on the MegaDepth1500 dataset [17], using 2048 ground truth correspondences perturbed with zero-mean Gaussian noise of varying standard deviation (STD) in pixels.

Dense feature matching An alternative to the aforementioned sparse feature extraction methods are dense feature matching methods like LoFTR [29] and RoMa [9], which directly process image pairs. The availability of information from both images enables dense methods to outperform their sparse counter parts in terms of accuracy. However, dense matching approaches are computationally costly. Furthermore, extracting features independently in a first step can be advantageous, for example, in a Simultaneous Localization And Mapping (SLAM) context, where local features can be stored in a map to be matched with features of many other images recorded in the future. Match refinement techniques consider information from both images after matching and therefore have the potential to bridge the accuracy gap between sparse and dense approaches.

Approaches to match refinement Match refinement can be applied after feature matching to adjust the image coordinates of matched keypoints based on the assumption that they represent corresponding points. This is useful as even small inaccuracies of a single pixel or less can disturb resulting estimates, *e.g.* of the camera pose (see Fig. 3).

One class of approaches directly uses photometric alignment of local patches for match refinement, *e.g.* Lucas–Kanade (LK) alignment [21] and the inverse compositional LK [1]. Such approaches, however, are computationally expensive and limited in their accuracy [22], particularly in cases of significant appearance changes.

Kim, Pollefeys, and Barath proposed Keypt2Subpx [14], an efficient learning-based method for match refinement that leverages the corresponding image patches and descriptors of matched keypoints. The authors argue that their refinement method simplifies the keypoint detection task as it is no longer required to detect sub-pixel accurate keypoints. Subsequently, as it is done in SuperPoint [5] and XFeat [25], the extractor can save computational effort by providing pixel coordinates as keypoints. Keypt2Subpx is trained to minimize the epipolar error. Accordingly, instead of requiring ground truth coordinates for matched keypoints, it is sufficient to have ground truth essential matrices for given image pairs, allowing the model to optimize keypoint positions directly for camera pose estimation.

Dusmanu *et al*. [6] propose Patch Flow, a refinement approach that aligns patches based on local optical flow and its resulting geometric cost. Lindenberger, Sarlin *et al*. improve upon Patch Flow with PixSfM [18], which presents a solution for match refinement in a multi-view scenario. They identify matches of the same keypoint over multiple images as tracks and then adjust the coordinates of all involved keypoints jointly in a featuremetric optimization.

As described previously, the feature extractor XFeat [25] detects keypoints only at pixel accuracy. However, Potje *et al*. propose a learned match refinement module that takes only the descriptors of matched keypoints as input and provides a sub-pixel offset as output that is added to the keypoints to improve their accuracy.

The match refinement solution presented in this paper differs from these approaches in several aspects. In contrast to Keypt2Subpx [14] and XFeat [25], our model takes only image patches at keypoint positions as input and not the descriptors or other output of the feature extractor, like the keypoint score. Hence, our model does not have to be trained specifically for each feature extractor. Unlike PixSfM [18], our method does not rely on costly featuremetric optimization, but infers the refinement, using a lightweight neural network, in a single forward pass. This makes the approach fast, while giving highest accuracy in match refinement across feature extractors (Fig. 1).

3. Method

We present *XRefine*, an attention-based keypoint match refinement model that takes only image patches as input and provides adjusted keypoint positions as output. For best generalizability, the model is trained feature extractor independently; we refer to this variant as **XRefine general**. For best accuracy, the model can be trained specifically for a feature extractor; we refer to this variant as **XRefine specific**. An overview of the approach is presented in Fig. 4.

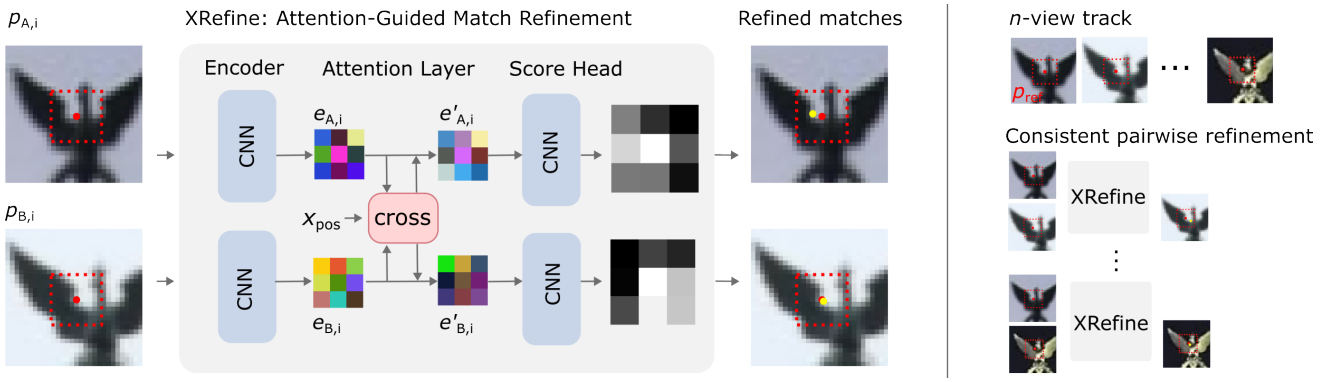

Figure 4. Architecture of our attention-guided match refinement. **Left:** The model takes 11×11 image patches $p_{A,i}, p_{B,i}$ (red dotted rectangle) around matched keypoints (red dots) as input. A CNN extracts embeddings $e_{A,i}, e_{B,i}$ which are updated using cross-attention. The score head then maps the updated embeddings $e'_{A,i}, e'_{B,i}$ to score maps $S_{A,i}, S_{B,i}$. A soft-argmax operation on these score maps finally yields the updated keypoint positions (yellow dots). **Right:** Extension to n-view problems. By using one patch as reference p_{ref} and using a model variant that refines only the second (non-reference) keypoint, consistent refinements can be obtained.

3.1. Architecture

The model takes two gray-scale patches p_A and p_B of size 11×11 as input. Both patches are processed independently by an encoder. The encoder performs five convolutions with 3×3 kernels, increasing the channel size from 1 to 16 with the first operation and then to 64 with the third operation. The first three and the last convolution are executed without padding; hence, the final embeddings e_A and e_B have a size of only 3×3. Now, a single block of multi-head cross-attention is applied between the two patch embeddings e_A and e_B. Each embedding is translated into a sequence of $3 \times 3 = 9$ tokens of dimensionality 64. To provide spatial information for each token, a learned positional encoding x_{pos} is added to the sequences. In order to update e_A, we use e_A as query and e_B as key and value, and vise versa to update e_B. After the cross attention, a score map head individually takes the updated embeddings as input, outputting their respective score map. The score map head is performing a single convolution with kernel size 3×3, with padding to keep the same size for the output. Then, a tanh operation brings the values into a range of $[-1, 1]$. Finally, similarly as in [14], a spatial soft-argmax is applied to each score map to obtain the updated keypoint position. The resulting coordinates are interpreted as relative coordinates to the center of the original patch. Accordingly, they are scaled-up to represent positions in the original 11×11 patch.

3.2. Training

The model is trained with the geometric training objective proposed by Kim, Pollefeys, and Barath [14], optimizing the epipolar error directly.

Dataset generation Training our models requires image pairs with overlapping field-of-view and known relative poses. We use two different training paradigms for the *specific* and *general* model: For the feature extractor specific datasets for *XRefine specific*, we use the respective feature extractor and detect 4096 keypoints with highest score values in each image. They are then matched, using mutual nearest neighbor matching (MNN), double soft max (DSM) [8], or LightGlue [19], depending on the extractor. To train *XRefine general*, we randomly select 4096 pixel coordinates with available depth information in the first image and project it into the second image to create a matching pair of *keypoints*. Then, both keypoints are randomly perturbed by adding a vector with x and y values sampled from a zero-mean normal distribution with a standard deviation of 1.5 pixels. We also tested smaller and larger standard deviations, as well as a uniform distribution, but observed best results with this setting. Subsequently, 11×11 image patches are cropped at the center of each matched keypoint.

We train on MegaDepth [17], splitting the dataset as in [29] into 45900 train samples, 655778 evaluation samples, and 1500 samples for validation (also referred to as MegaDepth1500). Each sample represents two views with partially overlapping content. Images are loaded with the GlueFactory [24] library, resizing them to 1024 pixels on the longer side, while keeping the aspect ratio.

Details Our training runs for 120 epochs. In each epoch, 2048 matches are randomly sampled for each image pair of the training split of MegaDepth [17]. We use PyTorch 2.1.2, the Adam optimizer [15] with a learning rate of 0.0001, and a batch size of 8. We validate after each epoch on MegaDepth1500 [17]. The weights for a given setup are selected as those with highest AUC5 performance on the validation dataset within two trainings with different seeds.

Dataset	Refinement	Avg.(%)	Min.(%)	Max.(%)
MegaDepth	Keypt2Subpx	8.07	0.91	28.61
MegaDepth	PixSfM	11.92	0.30	33.60
MegaDepth	XRefine general	15.42	2.28	42.26
MegaDepth	XRefine specific	15.99	3.80	41.92
ScanNet	Keypt2Subpx	6.26	-0.51	10.56
ScanNet	PixSfM	8.74	-3.77	17.81
ScanNet	XRefine general	16.34	3.43	27.11
ScanNet	XRefine specific	17.52	4.29	29.11
KITTI	Keypt2Subpx	0.23	-0.43	1.07
KITTI	PixSfM	0.75	-0.72	2.01
KITTI	XRefine general	1.07	-0.38	2.91
KITTI	XRefine specific	1.18	-0.21	2.73

Table 1. Summary of results for the extractors DeDoDe [8], SIFT [20], SuperPoint [5], and XFeat [25] on each of our three evaluation datasets MegaDepth [17], ScanNet [4], and KITTI [10]. We present the average, minimum, and maximum improvement of the AUC5 relative to the results without match refinement.

3.3. Generalization to n images

The proposed model adjusts keypoint locations across two views. However, some 3D vision tasks require consistent keypoints across n views. One example is Structure-from-Motion which typically builds feature tracks consisting of $T \geq 2, T \in \mathbb{N}$ matched keypoints. Naively applying our refinement to individual image pairs within a track could result in inconsistent refinements across pairs. To address this issue, we propose an architecture variant which only adjusts the second keypoint (see Fig. 4). In this variant, we still perform cross-attention between feature maps, but the score map S_i as well as the keypoint shift d_i are only inferred for the second image.

Given a feature track $\{\mathbf{u}_1, \mathbf{u}_2, ...\mathbf{u}_T\}$, we then define one of the keypoints as reference $\mathbf{u}_{\text{ref}}$, and apply the refinement to all other keypoints by passing pairs $\{(\mathbf{u}_{\text{ref}}, \mathbf{u}_2), (\mathbf{u}_{\text{ref}}, \mathbf{u}_3), ...(\mathbf{u}_{\text{ref}}, \mathbf{u}_{T-1})\}$ to the model. Thereby, all keypoints are refined towards the reference keypoint, resulting in a consistently refined track.

4. Evaluation

We evaluate match refinement for relative pose estimation in Sec. 4.1 and point cloud triangulation in Sec. 4.2.

4.1. Relative pose estimation

We evaluate on the photo-tourism dataset MegaDepth [17], the indoor dataset ScanNet [4], and the KITTI [10] visual odometry dataset. Our use of MegaDepth [17] is described in Sec. 3.2. Due to the large size of the evaluation dataset, we consider only every 10th image pair, *i.e.* 65577 pairs. For ScanNet [4], we evaluate on the 1500 samples selected in [27], resizing images to 640×480. For KITTI, we use the 2790 image pairs selected in [12] at a size of 1240×376.

We compare *XFeat specific* and *XFeat general* with three state-of-the-art match refinement approaches described in Sec. 2: Keypt2Subpx [14], PixSfM [18], and the refinement approach proposed in XFeat [25]. For Keypt2Subpx, weights for only a few feature extractors are publicly available; therefore, we train the model with the same procedure described in Sec. 3.2. We observe very similar performance of our re-trained Keypt2Subpx weights as for the publicly available weights. Details can be found in the appendix. The PixSfM solution for match refinement is independent of the feature extractor, so we can use the publicly available solution. The XFeat refinement approach is trained specifically for a variant of XFeat [25] that is called XFeat*, which, in contrast to the default XFeat, extracts features at two image sizes, and is reported to achieve better performance when using a larger number of features per image. We use the weights provided by the authors and use the XFeat solution only for XFeat*.

If not specified differently, we extract always 2048 features per image and match features using mutual nearest neighbor matching (MNN), double soft max (DSM) [8], or LightGlue (LG) [19]. For essential matrix estimation, we employ, as suggested in [14], GC-RANSAC [2] with 1000 iterations and a threshold of 1 pixel.

In terms of evaluation metrics, we follow [13], measuring pose estimation performance as area under curve (AUC) of pose errors that represent the maximum of translation direction error and the rotation error of the estimated pose compared to the given ground truth. We report the AUC for thresholds of 5, 10, and 20 degrees. The reported values are averages from 10 repetitions of the same experiment.

Main results As a brief overview, Tab. 1 summarizes the results over the evaluated feature extractor and matcher pairings, including DeDoDe [8], SIFT [20], SuperPoint (SP) [5], and XFeat [25]. Individual results are shown in Tab. 2 for MegaDepth [17], Tab. 3 for ScanNet [4], and Tab. 4 for KITTI [10]. More feature extractors are presented in the appendix. Results for XFeat* are not included in the summary Tab. 1, as it is an outlier extractor that was not intended to be used without match refinement and therefore has an unusually large benefit from it, *e.g.* for *XFeat general* an improvement of 158.51% on MegaDepth and 484.70% on KITTI for the AUC5. Furthermore, the XFeat refinement approach is not included in the summary table, as it can be only evaluated on XFeat*, but individual results can be found in Tabs. 2 to 4.

Overall, we observe that XRefine performs significantly better than existing methods, including Keypt2Subpx [14] and the match refinement method of PixSfM [18]. It can further be observed that *XRefine specific* performs a bit bet-

Extract+Match	Refinement	AUC5	AUC10	AUC20
SP+MNN		34.91	45.00	53.56
SP+MNN	Keypt2Subpx	36.20	46.09	54.32
SP+MNN	PixSfM	38.30	47.96	55.86
SP+MNN	XRefine general	**38.87**	**48.50**	56.35
SP+MNN	XRefine specific	38.86	**48.50**	**56.36**
SP+LG		58.48	71.41	80.83
SP+LG	Keypt2Subpx	60.16	72.73	81.78
SP+LG	PixSfM	62.05	74.15	82.72
SP+LG	XRefine general	62.86	74.83	83.27
SP+LG	XRefine specific	**63.07**	**75.02**	**83.40**
DeDoDe+DSM		34.88	48.64	60.84
DeDoDe+DSM	Keypt2Subpx	44.86	58.08	68.84
DeDoDe+DSM	PixSfM	46.60	59.09	69.20
DeDoDe+DSM	XRefine general	**49.62**	**62.20**	72.06
DeDoDe+DSM	XRefine specific	49.50	62.12	**72.07**
SIFT+MNN		19.76	26.53	33.00
SIFT+MNN	Keypt2Subpx	19.94	26.67	33.09
SIFT+MNN	PixSfM	19.82	26.49	32.82
SIFT+MNN	XRefine general	20.21	26.95	33.31
SIFT+MNN	XRefine specific	**20.51**	**27.31**	**33.69**
XFeat+MNN		36.45	47.89	57.81
XFeat+MNN	Keypt2Subpx	38.01	49.06	58.52
XFeat+MNN	PixSfM	40.06	50.99	60.13
XFeat+MNN	XRefine general	41.46	52.37	61.35
XFeat+MNN	XRefine specific	**41.94**	**52.83**	**61.78**
XFeat*+MNN		18.80	30.26	42.25
XFeat*+MNN	XFeat-Refine.	31.18	43.25	54.15
XFeat*+MNN	Keypt2Subpx	33.11	45.18	55.96
XFeat*+MNN	PixSfM	34.63	46.07	56.16
XFeat*+MNN	XRefine general	37.96	49.62	59.60
XFeat*+MNN	XRefine specific	**38.36**	**49.99**	**59.90**

Table 2. Pose estimation results on MegaDepth [17]. Bold indicates best performance and underscores second best per feature.

Extract+Match	Refinement	AUC5	AUC10	AUC20
SP+MNN		11.51	23.87	37.92
SP+MNN	Keypt2Subpx	12.36	25.07	39.20
SP+MNN	PixSfM	13.56	26.78	40.66
SP+MNN	XRefine general	14.63	28.23	42.33
SP+MNN	XRefine specific	**14.86**	**28.38**	**42.48**
SP+LG		19.48	37.40	54.79
SP+LG	Keypt2Subpx	20.31	38.21	55.43
SP+LG	PixSfM	21.25	39.29	55.93
SP+LG	XRefine general	**22.49**	**40.58**	**57.27**
SP+LG	XRefine specific	21.90	40.10	56.83
DeDoDe+DSM		10.13	21.04	32.42
DeDoDe+DSM	Keypt2Subpx	11.20	23.02	34.99
DeDoDe+DSM	PixSfM	10.44	21.85	33.98
DeDoDe+DSM	XRefine general	11.55	23.15	35.27
DeDoDe+DSM	XRefine specific	**11.71**	**23.55**	**35.51**
SIFT+MNN		5.83	12.32	20.14
SIFT+MNN	Keypt2Subpx	5.80	12.37	20.16
SIFT+MNN	PixSfM	5.61	11.88	19.36
SIFT+MNN	XRefine general	6.03	12.71	20.65
SIFT+MNN	XRefine specific	**6.08**	**12.79**	**20.72**
XFeat+MNN		10.28	22.04	35.77
XFeat+MNN	Keypt2Subpx	11.27	23.32	37.08
XFeat+MNN	PixSfM	12.08	24.40	38.23
XFeat+MNN	XRefine general	12.51	25.49	39.43
XFeat+MNN	XRefine specific	**12.97**	**26.07**	**40.02**
XFeat*+MNN		8.32	18.65	32.21
XFeat*+MNN	XFeat-Refine.	12.89	26.30	41.17
XFeat*+MNN	Keypt2Subpx	13.23	26.29	40.66
XFeat*+MNN	PixSfM	12.42	24.99	39.25
XFeat*+MNN	XRefine general	**14.16**	27.23	41.63
XFeat*+MNN	XRefine specific	**14.16**	**27.50**	**41.99**

Table 3. Pose estimation results on ScanNet [4]. Bold indicates best performance and underscores second best per feature.

ter than *XRefine general* which is expected as *XRefine specific* is specifically trained for the respective detector, and can therefore exploit learned priors, such as the magnitude of keypoint displacements.

Differences across datasets While the performance gains achieved through refinement are significant on MegaDepth and ScanNet, we observe only small performance gains on KITTI for most detectors. This can be explained by the relatively simple visual odometry use case: in contrast to the more challenging MegaDepth and ScanNet datasets, KITTI visual odometry presents only minor visual appearance changes in the paired images. Hence, state-of-the-art feature extractors often deliver sufficiently accurate keypoints even without refinement.

Differences across detectors It can further be observed that the effectiveness of match refinement depends on the keypoint detector. We observe significant performance gains for SuperPoint, DeDoDe, XFeat and XFeat*. Since SuperPoint and XFeat are providing keypoint positions only with pixel accuracy, their gain from match refinement can be expected. On the other hand, the performance of SIFT benefits only marginally, if at all, from match refinement, which could be explained by its elaborate Difference of Gaussian pyramid based keypoint detection approach.

Runtime evaluation Table 5 shows the computation time of all refinement methods averaged over 10000 image pairs with 2048 64-dimensional XFeat [25] features per image, evaluated on an Nvidia RTX A5000 GPU. While XRefine is with 3.61ms only marginally slower than Keypt2Subpx

Extract+Match	Refinement	AUC5	AUC10	AUC20
SP+MNN		83.11	90.74	95.16
SP+MNN	Keypt2Subpx	83.07	90.70	95.15
SP+MNN	PixSfM	84.11	91.28	95.44
SP+MNN	XRefine general	84.22	91.33	95.46
SP+MNN	XRefine specific	**84.37**	**91.40**	**95.49**
SP+LG		83.37	90.84	95.12
SP+LG	Keypt2Subpx	83.63	90.93	95.14
SP+LG	PixSfM	84.22	91.30	95.39
SP+LG	XRefine general	84.34	91.35	**95.40**
SP+LG	XRefine specific	**84.42**	**91.38**	**95.40**
DeDoDe+DSM		83.98	91.31	95.42
DeDoDe+DSM	Keypt2Subpx	84.21	91.42	95.50
DeDoDe+DSM	PixSfM	84.17	91.35	95.45
DeDoDe+DSM	XRefine general	84.24	91.42	95.50
DeDoDe+DSM	XRefine specific	**84.50**	**91.54**	**95.57**
SIFT+MNN		**83.79**	**91.32**	**95.51**
SIFT+MNN	Keypt2Subpx	83.43	91.14	95.44
SIFT+MNN	PixSfM	83.19	91.03	95.39
SIFT+MNN	XRefine general	83.47	91.17	95.45
SIFT+MNN	XRefine specific	83.61	91.24	95.49
XFeat+MNN		81.55	89.99	94.79
XFeat+MNN	Keypt2Subpx	82.42	90.47	95.00
XFeat+MNN	PixSfM	83.19	90.93	95.27
XFeat+MNN	XRefine general	**83.92**	**91.26**	**95.39**
XFeat+MNN	XRefine specific	83.78	91.21	95.37
XFeat*+MNN		13.79	19.99	24.07
XFeat*+MNN	XFeat-Refine.	77.59	86.98	92.55
XFeat*+MNN	Keypt2Subpx	72.99	84.01	90.43
XFeat*+MNN	PixSfM	78.73	87.90	93.17
XFeat*+MNN	XRefine general	80.63	89.05	93.85
XFeat*+MNN	XRefine specific	**80.90**	**89.30**	**94.11**

Table 4. Pose estimation results on KITTI [10] odometry. Bold indicates best performance and underscores second best per feature.

with 3.43ms, the feature-metric optimization approach of PixSFM is significantly slower with 70.28ms. Additionally, PixSFM extracts feature embeddings for the entire images with S2DNet, which, if included in the measurement, results in a runtime of 1435.71ms. The XFeat-Refinement approach, on the other hand, is with a runtime of 0.55ms very light weighted, but also limited in its accuracy.

Ablation results In Tab. 6, we present results for several variants of our proposed model for XFeat [25] on MegaDepth1500 [17]. *Small General* and *Small Specific* represent *XRefine general* and *XRefine specific* as described in Sec. 3. Removing the cross-attention layer (Small Specific - No Attn.) significantly reduces performance as information is no longer exchanged between the matched keypoint regions. Replacing the score map head with descriptor cosine similarity (Small Specific - Co-Sim), as in Keypt2Subpx [14], is marginally faster but less accurate and sacrifices generalizability, because this model requires per-descriptor training. Refining only the second keypoint (Small Specific - Only 2nd), as proposed in Sec. 3.3, lowers

Refinement Method	Runtime (ms)
XFeat-Refinement	0.55
Keypt2Subpx	3.43
XRefine (ours)	3.61
PixSfM without S2DNet	70.28
PixSfM with S2DNet	1435.71

Table 5. Runtime measurements on a NVIDIA RTX A5000.

Refinement	AUC5	AUC10	AUC20	t(ms)
	37.95	52.43	64.83	
Small Specific - No Attn.	41.20	54.96	66.32	2.46
Small Specific - Co-Sim	45.58	59.14	69.72	3.54
Small Specific - Only 2nd	46.53	59.70	69.88	3.34
Small General	46.86	60.05	70.28	3.61
Small Specific	47.52	60.53	70.68	3.59
Small Specific - Desc. Attn.	47.55	60.76	70.86	4.34
Large General	49.00	61.97	71.70	19.68
Large Specific	50.05	62.82	72.32	19.71

Table 6. Results for variants of our model on MegaDepth1500 [17] with XFeat [25] features and MNN matching. In bold, we highlight the two models for which results are presented in Tabs. 1 to 4.

#KP per image	Refinement	AUC5	AUC10	AUC20
2048		37.95	52.43	64.83
2048	Keypt2Subpx	40.46	54.63	66.31
2048	XRefine general	46.86	60.05	70.28
2048	XRefine specific	**47.52**	**60.53**	**70.68**
4096		40.16	54.00	65.62
4096	Keypt2Subpx	42.62	56.18	67.04
4096	XRefine general	48.01	60.90	70.70
4096	XRefine specific	**48.79**	**61.33**	**71.01**
8192		39.59	53.43	64.91
8192	Keypt2Subpx	41.36	54.86	65.91
8192	XRefine general	47.19	60.18	69.98
8192	XRefine specific	**47.97**	**60.62**	**70.38**
16384		39.58	53.29	64.78
16384	Keypt2Subpx	41.39	54.89	65.92
16384	XRefine general	47.14	60.04	69.90
16384	XRefine specific	**47.93**	**60.66**	**70.37**

Table 7. Results for varying numbers of extracted keypoints (KPs) per image on MegaDepth1500 [17] with XFeat [25] features and mutual nearest neighbor matching.

Refinement Method	ETH3D indoor						ETH3D outdoor					
	Accuracy (%)			Completeness (%)			Accuracy (%)			Completeness (%)		
	1cm	2cm	5cm	1cm	2cm	5cm	1cm	2cm	5cm	1cm	2cm	5cm
	78.89	87.73	94.49	0.61	2.26	9.03	54.08	69.29	83.64	0.10	0.59	4.08
XRefine general	84.31	90.80	96.04	0.63	2.25	8.78	62.83	76.10	87.76	0.12	0.63	4.14
PixSfM KA	89.09	93.55	96.96	0.71	2.43	9.19	70.55	82.39	91.46	0.15	0.76	4.70

Table 8. Triangulation results of different refinement methods on ETH3D indoor and outdoor datasets. Our proposed n-view refinement consistently improves triangulation accuracy. PixSfM yields most accurate results for this use-case, as it performs a joint keypoint refinement across the full tracks, rather than separate pairwise refinements. Keypt2Subpx and the XFeat-Refinement approach cannot be applied for this use-case as it is limited to 2-view refinement which would yield inconsistent tracks.

accuracy slightly due to its restriction. Finally, incorporating an additional attention mechanism with the average descriptor (Small Specific - Desc. Attn.) yields a small accuracy gain but at a disproportionately increased runtime.

Large General and *Large Specific* are similar to *Small General* and *Small Specific*, but they make use of a larger architecture. In contrast to the small models, the large models reduce the embedding size to 5×5 instead of 3×3, by adding padding once more in the encoder. Also, they employ three cross attention blocks between the patch embeddings, instead of only one. We observe significantly improved pose estimation results for the large variants, but also a significantly increased runtime. These models could be used in use cases without strict runtime requirements.

Varying numbers of keypoints We investigate the effect of having varying numbers of keypoints extracted per image. Results for XFeat matches are presented in Tab. 7. XFeat reaches best performance at 4096 keypoints per image. With 21.49% the relative improvement of the AUC5 metric from using *XRefine specific* refinement at this number of keypoints per image is a bit smaller than it is at 2048 keypoints per image with 25.22%, but still significant. For larger numbers of keypoints per image the performance of XFeat, with and without refinement, decreases slightly. The reduced advantage of using refinement with larger numbers of keypoints per image might be explained by a higher chance of obtaining a consistent set of accurate matches.

4.2. Point cloud triangulation

To demonstrate the benefit of the proposed refinement in n-view 3D vision problems, we evaluate its effect on 3D point cloud triangulation. Using the ETH3D dataset [28], we follow the protocol from [18] and use n-view feature tracks to triangulate a sparse 3D model, given reference camera poses and intrinsics. Evaluation is based on the PixSfM [18] repository with SuperPoint and MNN matching, where we integrated our refinement, but deactivated the feature-metric bundle adjustment for all methods, to compare only the effect of keypoint refinement.

Tab. 8 shows that our n-view refinement introduced in Sec. 3.3 consistently improves triangulation accuracy compared to no refinement, demonstrating the suitability of XRefine for 3D vision tasks beyond relative pose estimation. The improvement achieved by PixSfM is not reached which is expected as PixSfM is designed to jointly optimize all keypoints within a track, whereas our approach takes separate pairs of keypoints as input. While the joint optimization of PixSfM results in highest accuracy, it comes at the cost of computation time: While PixSfM scales quadratically with track length T, *i.e.* with $\mathcal{O}(T^2)$, our pairwise refinement exhibits linear scaling $\mathcal{O}(T)$. Together with the generally higher computation time of PixSfM for a single image pair (see Tab. 5), this shows a trade-off between accuracy and runtime. While our refinement is significantly faster, most accurate n-view triangulation results can be obtained by the global refinement used in PixSfM.

5. Conclusion

We presented a novel match refinement model that outperforms other state-of-the-art refinement methods in its impact on pose estimation performance without sacrificing computational efficiency. This is achieved through cross-attention between image patch embeddings without requiring detector-specific inputs like descriptors or score maps. It was shown that the model can be trained in a generalized manner, making it applicable to any keypoint detector without retraining. While extending the approach from two views to n views yielded clear improvements in 3D point cloud triangulation, future work may enhance this further by adapting the architecture to directly accept n image patches as input. This would enable globally optimal refinement and potentially lead to higher accuracy gains in multi-view applications. Overall, this work represents a step toward more accurate 3D vision, and can be readily incorporated into existing sparse keypoint-based systems.

References

[1] S. Baker and I. Matthews. Equivalence and efficiency of image alignment algorithms. In *CVPR*, pages I–I, 2001. 3

[2] Daniel Barath and Jiří Matas. Graph-cut RANSAC. In *CVPR*, 2018. 5

[3] Peter Hviid Christiansen, Mikkel Fly Kragh, Yury Brodskiy, and Henrik Karstoft. UnsuperPoint: End-to-end unsupervised interest point detector and descriptor. *arXiv preprint arXiv:1907.04011*, 2019. 2

[4] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Niessner. ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 2, 5, 6, 1

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *CVPRW*, 2018. 1, 2, 3, 5

[6] Mihai Dusmanu, Johannes L. Schönberger, and Marc Pollefeys. Multi-view optimization of local feature geometry. In *ECCV*, pages 670–686, Cham, 2020. Springer International Publishing. 3

[7] Johan Edstedt, Georg Bökman, and Zhenjun Zhao. DeDoDe v2: Analyzing and improving the dedode keypoint detector. In *CVPRW*, pages 4245–4253, 2024. 2, 1

[8] Johan Edstedt, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. DeDoDe: Detect, don't describe — describe, don't detect for local feature matching. In *Int. Conf. on 3D Vision*, pages 148–157, 2024. 1, 2, 4, 5

[9] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. RoMa: Robust dense feature matching. In *CVPR*, pages 19790–19800, 2024. 3

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, pages 3354–3361, 2012. 2, 5, 7, 1

[11] Hugo Germain, Guillaume Bourmaud, and Vincent Lepetit. S2DNet: Learning image features for accurate sparse-to-dense matching. In *ECCV*, 2020. 1

[12] You-Yi Jau, Rui Zhu, Hao Su, and Manmohan Chandraker. Deep keypoint-based camera pose estimation with geometric constraints. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 4950–4957, 2020. 5

[13] 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. *IJCV*, 129(2):517–547, 2021. 5

[14] Shinjeong Kim, Marc Pollefeys, and Daniel Barath. Learning to make keypoints sub-pixel accurate. In *ECCV*, pages 413–431. Springer Nature Switzerland, 2025. 1, 2, 3, 4, 5, 7

[15] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[16] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision*, pages 71–91. Springer, 2024. 1

[17] Zhengqi Li and Noah Snavely. MegaDepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 1, 2, 3, 4, 5, 6, 7

[18] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-perfect structure-from-motion with featuremetric refinement. In *ICCV*, pages 5987–5997, 2021. 1, 2, 3, 5, 8

[19] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local feature matching at light speed. In *ICCV*, pages 17627–17638, 2023. 4, 5, 1

[20] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1, 2, 5

[21] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981. 3

[22] Vincent Lui, Jonathon Geeves, Winston Yii, and Tom Drummond. Efficient subpixel refinement with symbolic linear predictors. In *CVPR*, 2018. 3

[23] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Trans. on Machine Learning Research*, 2024. Featured Certification. 2

[24] Rémi Pautrat*, Iago Suárez*, Yifan Yu, Marc Pollefeys, and Viktor Larsson. GlueStick: Robust image matching by sticking points and lines together. In *ICCV*, 2023. 4

[25] G. Potje, F. Cadar, A. Araujo, R. Martins, and E. R. Nascimento. XFeat: Accelerated features for lightweight image matching. In *CVPR*, pages 2682–2691, 2024. 1, 2, 3, 5, 6, 7

[26] Jerome Revaud, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel. R2D2: Reliable and repeatable detector and descriptor. In *NeurIPS*, 2019. 2, 1

[27] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 5

[28] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *CVPR*, 2017. 2, 8

[29] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. In *CVPR*, pages 8922–8931, 2021. 3, 4

[30] Jiexiong Tang, H. Kim, V. Guizilini, S. Pillai, and A. Rares. Neural outlier rejection for self-supervised keypoint learning. In *ICLR*, 2020. 2

[31] MichałTyszkiewicz, Pascal Fua, and Eduard Trulls. DISK: Learning local features with policy gradient. In *NeurIPS*, pages 14254–14265, 2020. 2, 1

[32] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 1

[33] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D vision made easy. In *CVPR*, 2024. 1

[34] Xiaoming Zhao, Xingming Wu, Weihai Chen, Peter C. Y. Chen, Qingsong Xu, and Zhengguo Li. ALIKED: A lighter

keypoint and descriptor extraction network via deformable transformation. *IEEE Trans. Instrum. Meas.*, 72:1–16, 2023. 2, 1