

# Dense Match Summarization for Faster Two-view Estimation

Jonathan Astermark, Anders Heyden, and Viktor Larsson

Centre for Mathematical Sciences, Lund University

{jonathan.astermark, anders.heyden, viktor.larsson}@math.lth.se

## Abstract

*In this paper, we speed up robust two-view relative pose from dense correspondences. Previous work has shown that dense matchers can significantly improve both accuracy and robustness in the resulting pose. However, the large number of matches comes with a significantly increased runtime during robust estimation in RANSAC. To avoid this, we propose an efficient match summarization scheme which provides comparable accuracy to using the full set of dense matches, while having 10-100x faster runtime. We validate our approach on standard benchmark datasets together with multiple state-of-the-art dense matchers.*

## 1. Introduction

Determining the two-view camera geometry is an important sub-task in many computer vision problems, *e.g.* for Simultaneous Localization and Mapping or Structure-from-Motion. The most common approach is to first detect corresponding points in the two views, followed by robust estimation using RANdom Sample Consensus (RANSAC) [18] which both estimates the relative pose and identifies potential outlier matches. Traditionally, keypoint matching is performed by independently detecting keypoints in each view, followed by a matching step, either directly comparing descriptor similarity or using a learned matcher.

Recently, a lot of attention has been given to so-called detector-free matching, which takes the images as input and directly establishes (semi-)dense correspondences. These matchers produce significantly more matches, compared to the detector-based counterparts, especially in weakly textured image regions, such as walls, floors, and ceilings. Dense matching shows great promise and currently achieves the most accurate two-view estimates in standard benchmarks. However, this comes with a trade-off in runtime, both for running the matching and in the subsequent robust estimation (*i.e.* RANSAC).

In this paper, we aim to improve the runtime of the robust estimation step. We show that when using dense matchers, most of the matches provide redundant geometric constraints, and we present an efficient match summarization scheme that selects a subset of  $\approx 1\%$  of the matches, which we call *representative matches*, that are used for robust estimation. Using this smaller set of matches, we can obtain a 10-100x speedup with a marginal loss in pose accuracy. In addition, we present an approximation approach for enriching each representative match to capture the geometric constraints of nearby correspondences in a single  $9 \times 9$ -matrix (regardless of how many neighboring matches are included). This enrichment allows for an even smaller loss of pose accuracy, while still being significantly faster compared to dense correspondence. In Figure 1 we visualize one example from MegaDepth [26].



**Figure 1. Relative Pose Estimation from Dense Matches.** The top images show 10,000 semi-dense matches from DKM [15] with same colors indicating corresponding points. The large number of matches makes robust estimation with RANSAC accurate (pose error  $\epsilon = 0.38^\circ$ ) but slow (runtime  $t = 98$  ms). In this paper we show that we can get comparable accuracy with significantly lower runtime by sparsifying the dense matches (bottom images). Each sparsified match is represented by a  $9 \times 9$ -matrix that summarizes the geometric constraints from nearby matches.

We validate our claims and our approach with extensive experiments and ablation studies on standard benchmarks with different state-of-the-art dense matchers.

In summary, our contributions are:

- A scheme for summarizing any dense matches into sparse *representative matches*, which speeds up the robust estimation by a factor 10-100x, with only a very small loss in accuracy.
- An approximation-based approach to summarize the contributions of each cluster, which we use to refine the output from RANSAC. This lets us maintain the high accuracy from dense matches, with only a small increase in runtime compared to the approach above.
- In experiments on standard benchmarks, with multiple state-of-the-art dense matchers, we are able to consistently show a significant speedup. Furthermore, our experiments show that sparsified dense matches outperform state-of-the-art sparse matches in terms of accuracy.

## 1.1. Related Work

**Matching and Correspondences.** Image correspondence for geometric estimation traditionally works with a *detect-then-match* paradigm that first identifies keypoints (*e.g.* SIFT [28], SuperPoint [12]) independently in each image, followed by a matching step. The matching is either done by comparing descriptor similarity ([12, 14, 28]) or more recently via learned matchers ([27, 35]). In [36] the authors instead proposed to perform *detection-free* matching, which takes an image pair as input and directly regresses semi-dense pixel correspondences. This started a series of work on dense matching ([7, 15, 16, 20]) which improved on the initial work. Common for all dense methods is that they produce significantly more matches compared to classical detector-based methods. By considering both images jointly, the methods are able to propagate strong matches to weaker ones, allowing for even weakly textured regions to be matched between images.

In benchmarks for two-view relative pose estimation, these methods significantly improve the results in terms of pose accuracy. However, this comes at a trade-off with runtime, as the large number of matches makes robust estimation more costly. In this work, we offset some of this cost by speeding up the robust estimation. Our method is not specific to a particular matcher and can be applied to any dense correspondences.

**Robust Estimation.** For robust geometric estimation problems in computer vision, the RANSAC algorithm, originally introduced by Fischler and Bolles [18] in 1981, is the de facto standard. Modern RANSAC variants, while still working with similar principles as the original, usually incorporate some form of local optimization (LO-RANSAC [10, 24]) as well a more complex scoring beyond simple inlier counting, *e.g.* MSAC [37] or MAGSAC [3]. Other works improve by integrating various learned components in the pipeline, see *e.g.* [4, 6, 39]. There are also several works which focus on making RANSAC faster. In [9], the authors propose a probabilistic method (SPRT) for de-

ciding when to early exit. In [22], the authors propose to only perform scoring once two similar models are found. Rais et al. [32] instead aggregate multiple model hypotheses to form the final output. Barath et al. [2] speed up scoring by filtering out regions of the matches which cannot contain inlier correspondences for the current model. PROSAC [8] leverages per-match confidences to sample good models earlier, to speed up convergence. Ni et al. [30] group correspondences and model different inlier ratios for each cluster, which is used both for sampling and stopping criterion.

The match summarization method proposed in this paper is orthogonal (and could be combined) with the improvements in the above methods. In particular, most prior work focus on either speeding up the scoring or reducing the number of iterations, but do not consider refinement which is particularly costly with a large number of matches.

**Richer Correspondences.** For *affine-correspondences* (ACs) each match is associated with a  $2 \times 2$  matrix representing a local transformation around the keypoints. These matrices can be interpreted as a linearization of the local planar homography [33]. Affine correspondences have been used to derive more efficient minimal estimators ([5, 17, 33, 38]) as well as provide constraints for normal estimation [19]. In this paper, we derive a summarized correspondence expressed with a  $9 \times 9$ -matrix. While this matrix is similar to ACs in that it encodes additional local geometric constraints, it does not make assumption on local planarity and yields stronger geometric constraints, even allowing estimation from a single correspondence (see Section 4.4).

## 2. Background

Each 2D-point correspondence  $(\mathbf{x}, \bar{\mathbf{x}}) \in \mathbb{R}^3 \times \mathbb{R}^3$  (in homogeneous coordinates) constrain the relative pose by

$$\bar{\mathbf{x}}^T E \mathbf{x} = 0, \quad (1)$$

where  $E = [t]_{\times} R$  is the essential matrix. As the points  $(\mathbf{x}, \bar{\mathbf{x}})$  are measurements from the images, they will contain noise and will not satisfy (1) exactly, even if the match is correct. Thus, in the optimization, a residual such as the Sampson error is commonly used instead,

$$\mathcal{E}(E, \mathbf{x}, \bar{\mathbf{x}}) = \frac{(\bar{\mathbf{x}}^T E \mathbf{x})^2}{\|E_{12}\mathbf{x}\|^2 + \|(E^T)_{12}\bar{\mathbf{x}}\|^2}, \quad (2)$$

where  $E_{12}$  is the first two rows of  $E$ . The Sampson error approximates the squared reprojection error, and is used to determine match correctness (inlier or outlier). Most RANSAC-variants additionally use some form of MSAC-scoring [37] for the models, *i.e.* the essential matrix is chosen by minimizing the sum of truncated residuals

$$f(E) = \sum_{i=1}^N \min \left\{ \mathcal{E}(E, \mathbf{x}_i, \bar{\mathbf{x}}_i), \tau^2 \right\}, \quad (3)$$

where  $N$  denotes the number of matches and  $\tau \in \mathbb{R}_+$  is the inlier threshold. The model-scoring function  $f(E)$  is used both for selecting the best model, and for non-linear refinement. As the number of matches  $N$  grows large, evaluating  $f(E)$  will dominate the runtime cost in RANSAC.

### 3. Method

In this section, we present a method for efficiently summarizing the geometric constraints given by the dense matches. The idea is to first cluster the correspondences into subsets that yield similar geometric constraints, and replace each cluster with a single *representative match* used in robust estimation. Next, we compute a *proxy residual* that summarizes the geometric constraints from each cluster into a  $9 \times 9$ -matrix, irrespective of cluster size. This allows us to refine the pose without evaluating the full residual (3) in each step of the optimization. Figure 2 shows an overview of our method. In Section 3.1 we discuss different approaches for performing the clustering and selecting representative matches, and in Section 3.2 we derive the proxy residual.

In the paper we focus the presentation on the calibrated case (essential matrix), but the approach is directly applicable to the uncalibrated (fundamental matrix) setting as well.

#### 3.1. Clustering and Representative Matches

Given a set of dense matches, our goal is to find a sparse subset of representative matches that capture the same geometric constraints as the full set. To do this, we begin by clustering correspondences that contribute with similar residuals. From each cluster, we then select a single match which serves as a representative of that cluster.

We base our clustering on the assumption that matches close to each other should lead to similar residuals in (3). This motivates us to group matches based on their position in the two images. A simple approach is to use a standard clustering method, *e.g.* K-means, applied on the 4-dimensional match vectors (*i.e.* concatenated keypoint coordinates from both images). Alternatively, matches could be clustered based on the images, *e.g.* using superpixels. In Section 4.2, we discuss and compare different approaches for clustering and their trade-off in terms of runtime and accuracy for our method.

For each cluster, we then select the match closest to the cluster centroid (in 4D match space) as the representative match. The set of representative matches  $\{(c_i, \bar{c}_i)\}_{i=1}^K \subset \{(\mathbf{x}_i, \bar{\mathbf{x}}_i)\}_{i=1}^N$ , where  $K$  is the number of clusters, can be used directly in RANSAC to estimate the relative pose, using the sub-sampled cost

$$f_c(E) = \sum_{i=1}^K \min \{ \mathcal{E}(E, c_i, \bar{c}_i), \tau^2 \}. \quad (4)$$

Since the runtime of RANSAC is generally dominated by

scoring and refinement, which grows linearly with the number of matches, significant subsampling can drastically reduce the runtime. In our experiments (Section 4) we show that even when severely subsampling ( $K \approx N/80$ ), leading to 55x speedup, we can still obtain good relative pose estimates when using this method. This highlights that most constraints in the dense matches are redundant. However, the sparsified matches still will lead to a slight drop in accuracy. In the next section, we propose an approximation which enriches each of the matches to also encode the geometric constraints from the full cluster.

#### 3.2. Dense Match Summarization

We now present a method for approximating the full cost in (3) using the clustering. The idea is to replace the dense matches in each cluster with a more computationally efficient proxy residual. While we in the previous section reduced the number of residuals from  $N$  to  $K$  by only considering the representative matches, in this section we will capture more of the geometry per cluster by increasing this number to  $9K$ . We do this in two steps; first by assuming that each cluster is either all-inlier or all-outlier, and second by approximating the Sampson error in each cluster.

Let  $\mathcal{M} \subset \mathbb{R}^3 \times \mathbb{R}^3$  be the set of dense input matches, and let  $\mathcal{C}_1, \dots, \mathcal{C}_K \subset \mathcal{M}$  be a disjoint clustering of these with associated representative matches  $\{(c_k, \bar{c}_k)\}_{k=1}^K$ . By assuming that the matches in each cluster are either all inliers or all outliers, (3) can be rewritten as

$$f(E) = \sum_{k=1}^K \sum_{(\mathbf{x}, \bar{\mathbf{x}}) \in \mathcal{C}_k} \min \{ \mathcal{E}(E, \mathbf{x}, \bar{\mathbf{x}}), \tau^2 \} \quad (5)$$

$$\approx \sum_{k=1}^K \min \left\{ \sum_{(\mathbf{x}, \bar{\mathbf{x}}) \in \mathcal{C}_k} \mathcal{E}(E, \mathbf{x}, \bar{\mathbf{x}}), |\mathcal{C}_k| \tau^2 \right\}, \quad (6)$$

where  $|\mathcal{C}_k|$  denotes the number of matches in cluster  $\mathcal{C}_k$ . This means that each cluster either contributes the sum of all its dense residuals, or the constant factor  $|\mathcal{C}_k| \tau^2$  – whichever is smallest.

Now, consider a single cluster  $\mathcal{C}$  with representative match  $(c, \bar{c})$ . The sum of all its dense residuals is

$$f_{cl}(E; \mathcal{C}) = \sum_{(\mathbf{x}, \bar{\mathbf{x}}) \in \mathcal{C}} \mathcal{E}(E, \mathbf{x}, \bar{\mathbf{x}}), \quad (7)$$

*i.e.* the inner sum in (6). By construction, each match  $(\mathbf{x}, \bar{\mathbf{x}}) \in \mathcal{C}$  will lie relatively close to the cluster's representative match  $(c, \bar{c})$ . This motivates us to approximate the Sampson error (2) for each dense match as

$$\mathcal{E}(E, \mathbf{x}, \bar{\mathbf{x}}) \approx \frac{(\bar{\mathbf{x}}^T E \mathbf{x})^2}{\|E_{12} \mathbf{c}\|^2 + \|(E^T)_{12} \bar{\mathbf{c}}\|^2}, \quad (8)$$

*i.e.* by replacing  $(\mathbf{x}, \bar{\mathbf{x}})$  in the denominator with  $(c, \bar{c})$ . For brevity, we introduce  $\alpha(E; \mathcal{C}) = \|E_{12} \mathbf{c}\|^2 + \|(E^T)_{12} \bar{\mathbf{c}}\|^2$ .

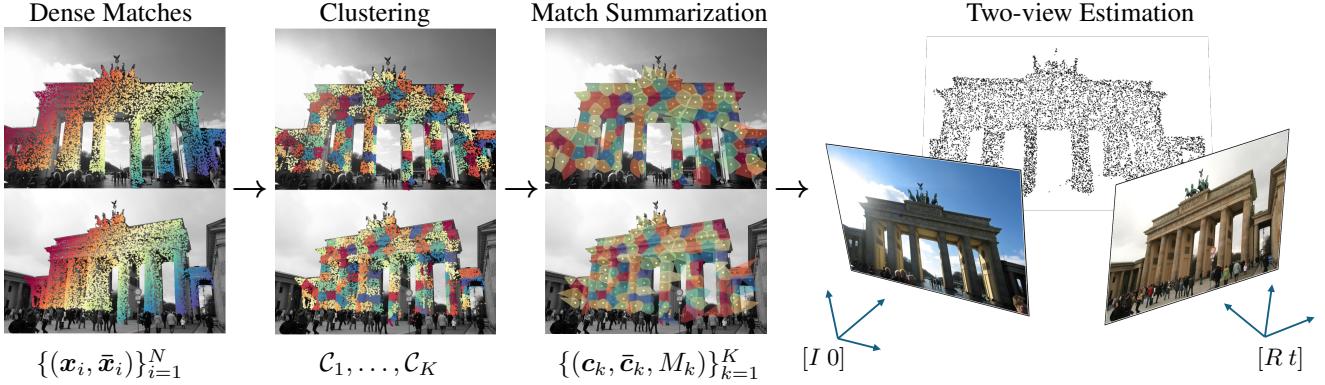


Figure 2. **Overview of our Summarization Scheme.** As input, our method takes a set of **dense matches**  $\{(\mathbf{x}_i, \bar{\mathbf{x}}_i)\}_{i=1}^N$ , where typically  $N = 10\,000$ . Through **clustering**, the matches are grouped into clusters  $\mathcal{C}_1, \dots, \mathcal{C}_K$ ,  $K \ll N$ , which yield approximately the same geometric constraints on the relative pose. For each cluster, we then perform **match summarization**, which replaces each group of matches with representative match  $(\mathbf{c}_k, \bar{\mathbf{c}}_k)$  and a  $9 \times 9$ -matrix  $M_k$  that encodes the geometric constraints. These summarized meta-matches can then be used for robust **two-view estimation** with significant speedup for a very small accuracy loss.

Then the approximation in (8) lets us write the sum of residuals for a cluster more compactly as

$$f_{\text{cl}}(E; \mathcal{C}) \approx \sum_{(\mathbf{x}, \bar{\mathbf{x}}) \in \mathcal{C}} \frac{(\bar{\mathbf{x}}^T E \mathbf{x})^2}{\alpha(E; \mathcal{C})} \quad (9)$$

$$= \frac{1}{\alpha(E; \mathcal{C})} \left\| \begin{pmatrix} \bar{\mathbf{x}}_1^T E \mathbf{x}_1 \\ \vdots \\ \bar{\mathbf{x}}_n^T E \mathbf{x}_n \end{pmatrix} \right\|^2 = \frac{1}{\alpha(E; \mathcal{C})} \|Ae\|^2, \quad (10)$$

where  $n = |\mathcal{C}_k|$ ;  $e \in \mathbb{R}^9$  is a vector containing the elements of  $E$ , and  $A \in \mathbb{R}^{n \times 9}$  is the matrix with rows  $A_i = (\mathbf{x}_i \otimes \bar{\mathbf{x}}_i)^T$  for all  $(\mathbf{x}_i, \bar{\mathbf{x}}_i) \in \mathcal{C}$ . Here,  $\otimes$  denotes the Kronecker product. The trick to efficiently evaluating (10) is that the large matrix  $A$  can be replaced with the *reduced measurement matrix* [34]  $M \in \mathbb{R}^{9 \times 9}$  using Cholesky factorization  $A^T A = M^T M$ , since

$$\|Ae\|^2 = e^T A^T A e = e^T M^T M e = \|Me\|^2. \quad (11)$$

Note that this is very fast to compute (in our experiments a total of 0.2 ms for 128 clusters with  $N = 10,000$ ) and only needs to be done once for each clustering.

The final approximate cost function for all matches, to be evaluated in each step of the optimization, is

$$f_{\text{approx}}(E) = \sum_{k=1}^K \min \left\{ \frac{1}{\alpha(E; \mathcal{C}_k)} \|M_k e\|^2, |\mathcal{C}_k| \tau^2 \right\}. \quad (12)$$

In our experiments (Section 4.3) we show that this provides a tight approximation to the sum of Sampson errors, while being significantly faster to compute.

Above, the approximation was applied to the Sampson residual, but it is in principle also applicable to residuals

with similar form, e.g. Symmetric Epipolar Distance:

$$\sum_{(\mathbf{x}, \bar{\mathbf{x}})} \left( \frac{(\bar{\mathbf{x}}^T E \mathbf{x})^2}{\|E_{12}\mathbf{x}\|^2} + \frac{(\bar{\mathbf{x}}^T E \mathbf{x})^2}{\|E_{12}^T \bar{\mathbf{x}}\|^2} \right) \approx \frac{\|Ae\|^2}{\|E_{12}\mathbf{c}\|^2} + \frac{\|Ae\|^2}{\|E_{12}^T \bar{\mathbf{c}}\|^2}.$$

## 4. Experiments

In this section, we evaluate the match clustering and summarization scheme presented in Section 3, through experiments on real image pairs. First, in Section 4.2, we compare clustering methods and perform an ablation study on both the clustering method and number of clusters. In Section 4.3, this is followed by an evaluation of the approximation error of our summarization scheme. Next, in Section 4.4, we perform an ablation study on the integration of our summarization scheme in RANSAC.

We use two standard benchmarks for relative pose estimation, ScanNet-1500 [11, 35] and MegaDepth-1500 [26, 36], which each contain 1500 image pairs of indoor and outdoor scenes, respectively. For inlier thresholds, we use 1.0 pixels in MegaDepth-1500, and 2.5 pixels in ScanNet-1500. Following prior work, we report the pose error (max of rotation and translation error in degrees) with the Area Under Curve (AUC) up to some threshold. For our ablations and approximation evaluation, we evaluate on MegaDepth-1500, using  $N = 10\,000$  dense matches from DKM [15]. Finally, in Section 4.5 we show that our results generalize to other state-of-the-art dense matchers and other datasets. We also present results on estimation of the fundamental matrix on the challenging WxBS dataset [29].

### 4.1. Implementation Details

We implement our approximate residual in an LO-RANSAC framework building on PoseLib [23], which is a state-of-the-art robust estimation library in C++. To get a fair comparison with the dense and subsampled methods,

we implement the standard Sampson residual in the same framework. For the ablations and evaluations in the following sections, our method runs RANSAC using the representative matches from Section 3.1, followed by refinement using the approximate residuals described in Section 3.2. All timings are measured on a modern desktop CPU.

## 4.2. Ablation on Clustering

First, we explore different approaches for performing the match clustering. We explore both keypoint-based and image-based clustering methods. For the keypoint-based methods, we evaluate K-means clustering on both the 4-dimensional match vectors and the 2-dimensional keypoints from one of the images. Motivated by the geometric constraints in (10), we also evaluate clustering in the 9-dimensional constraint vectors  $A_i = \mathbf{x}_i \otimes \bar{\mathbf{x}}_i \in \mathbb{R}^9$ . For the K-means clustering experiments we use FAISS [13] on a CPU, running a maximum of 5 iterations.

For image-based clustering, we first segment one of the images, then consider all matches within the same segment as a cluster. We evaluate both a simple 2D-grid segmentation, and a more sophisticated superpixel detection. The 2D grid is created by dividing the image region between the minimum and maximum keypoint coordinates into  $m^2$  equally sized rectangles, where  $m = \lceil \sqrt{K} \rceil$ . For the superpixel detection, we use SLIC [1, 21]. For each clustering method, we select the representative match by finding the match closest to the centroid.

In Figure 3, we show some qualitative examples of the different clusterings on two example image pairs from MegaDepth-1500 and ScanNet-1500, respectively. In the top example, we see that the keypoint-based methods (2D, 4D, 9D) all give similar clustering patterns, while in the bottom example 2D clustering leads to a coarser clustering in the second image. This is explained by the smaller image overlap; while 4D and 9D clustering takes the keypoint density in both images into account, 2D clustering only sees the keypoints from one of the images. The image-based methods (Grid and SLIC), on the other hand, lead to larger clusters in both of the image pairs. This is because the segmentation is not proportional to the match density, so they may “waste” clusters on image regions with few or no matches.

In Table 1, we compare the errors and runtimes for different clustering methods on MegaDepth-1500, using a cluster size of 128. We report both the time for clustering, and for estimation using the clusters. We assume that grid-clustering can be done at negligible runtime with an efficient implementation. In the table, we see that the difference between clustering methods is quite small, except for SLIC which is markedly more expensive to calculate. A key difference with image-based clustering, however, is that it only needs to be done once per image. Keypoint-based clustering, on the other hand, needs to be done once per image

pair, which can lead to many more total evaluations. For large datasets with high co-visibility, this may be important to consider.

Method	AUC@5°	$\epsilon_{avg}$	Runtime	
			Clustering	RANSAC
K-means 2D	66.4	3.62	1.53 ms	1.46 ms
K-means 4D	66.9	3.28	1.22 ms	1.44 ms
K-means 9D	66.6	3.25	1.38 ms	1.43 ms
Grid	65.9	3.73	$\approx 0$ ms	1.34 ms
SLIC [21]	64.5	4.07	22.38 ms	0.97 ms

Table 1. **Clustering Method Ablation.** The table shows the pose errors for each clustering approach on MegaDepth-1500, as well as the average runtime for each clustering strategy.

Next, we evaluate the impact of the number of clusters. Figure 4 shows the AUC@5° (left) and runtime (center) plotted against the number of clusters, as well as AUC@5° vs. runtime (right). We also include the dense baseline (dashed). The results further indicate that all K-means clustering variants perform similarly, and that the performance gets close to the dense baseline as number of clusters increases. Based on these results, we select K-means 4D with  $K = 128$  as our main clustering method to be used in the experiments, but note that other choices are viable as well.

## 4.3. Evaluation of Approximation Error

In Section 3.2, we introduced a proxy residual to approximate the sum of squared Sampson residuals for a cluster. In this section, we experimentally validate this approximation and show that it provides a tight approximation of the true residual. We use 128 clusters per image pair obtained via Kmeans-9D, as defined in the previous section. For each cluster  $\mathcal{C}$ , we compute the per-cluster average Sampson residuals exactly as

$$\varepsilon_S^2 = \frac{1}{|\mathcal{C}|} \sum_{(\mathbf{x}, \bar{\mathbf{x}}) \in \mathcal{C}} \mathcal{E}(E_{gt}, \mathbf{x}, \bar{\mathbf{x}}), \quad (13)$$

and the squared approximate cluster residual as

$$\varepsilon_a^2 = \frac{1}{|\mathcal{C}|} \frac{\|M\text{vec}(E_{gt})\|^2}{\|(E_{gt})_{12}\mathbf{c}\|^2 + \|(E_{gt}^T)_{12}\bar{\mathbf{c}}\|^2}. \quad (14)$$

In Figure 5 we show the distribution of the difference  $\varepsilon_S - \varepsilon_a$  between the true residual and the approximation, computed on the 1500 image pairs from MegaDepth-1500 with 10 000 DKM matches. The residuals are evaluated for the ground truth essential matrix  $E_{gt} = [\mathbf{t}_{gt}] \times R_{gt}$ . Note that positive values indicate that the approximation gives too small residuals, while negative values correspond to the approximation giving too large residuals. We plot residuals between the 1<sup>st</sup> and 99<sup>th</sup> percentile. Thus we see that more

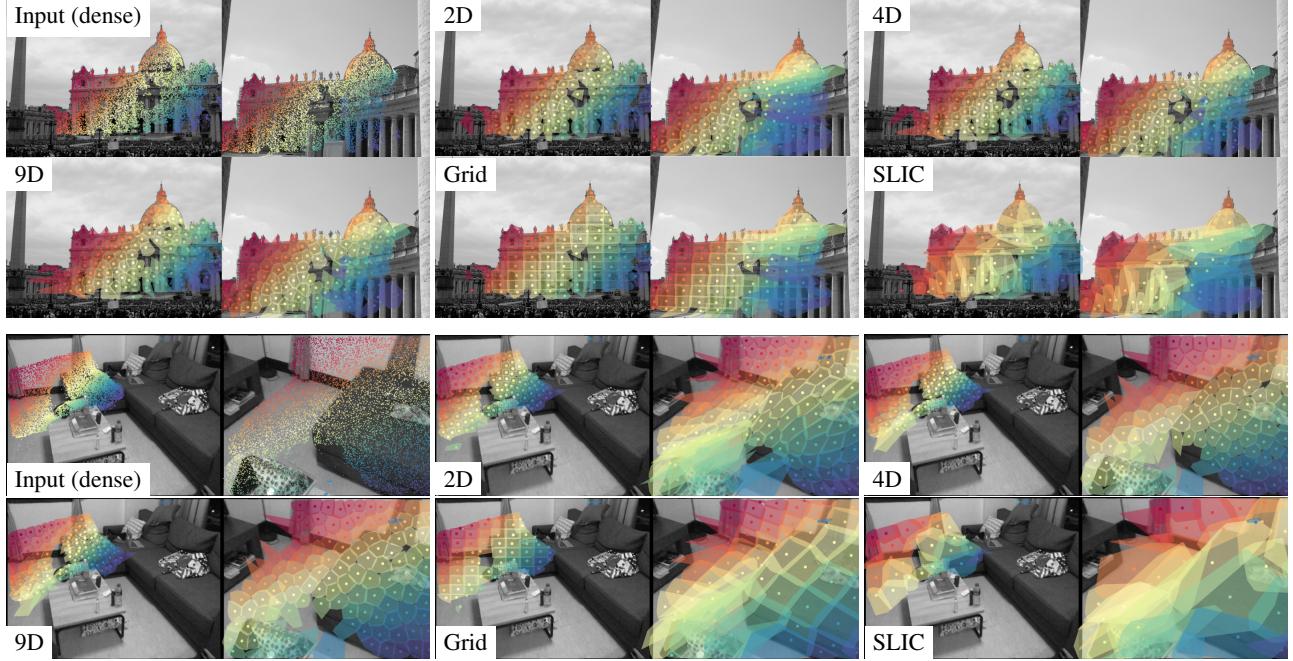


Figure 3. **Qualitative Comparison of Clustering Methods.** We compare different clustering methods on a single image pair from MegaDepth-1500 (top 2 rows) and ScanNet-1500 (bottom 2 rows), respectively. Dense matches from DKM are clustered using three keypoint-based methods (2D, 3D, 9D) and two image-based methods (Grid, SLIC).

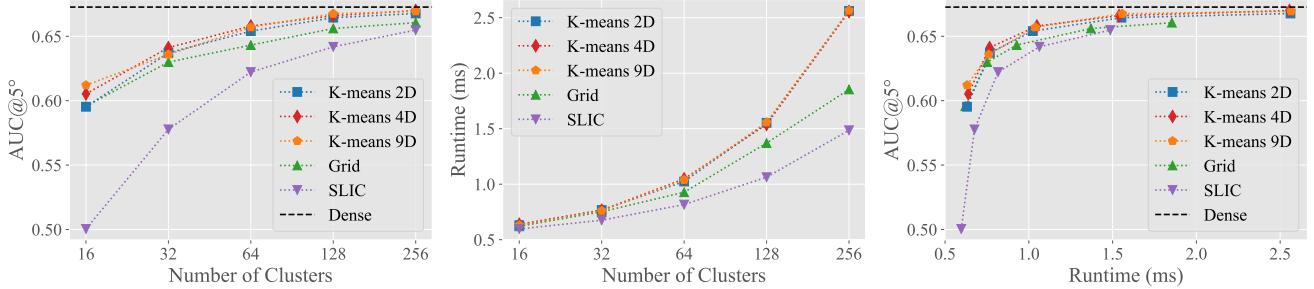


Figure 4. **Clustering Method and Hyperparameter Ablation.** The plots show AUC@5° (left), runtime (middle) and AUC-runtime-tradeoff (right) for different clustering methods and number of clusters  $K$ . We include AUC@5° for the dense baseline as reference.

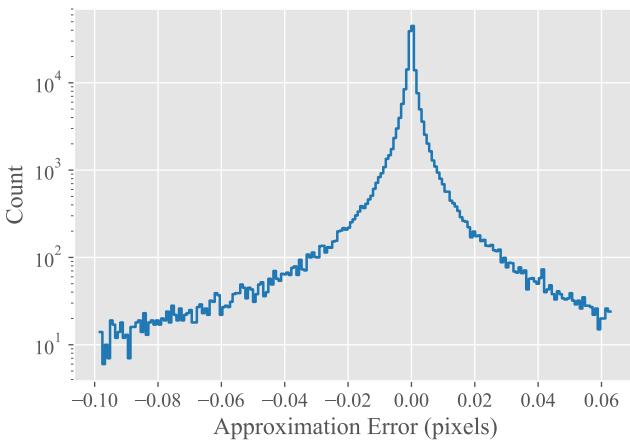
than 98 % of the residuals have an absolute approximation error of less than 0.1 pixels, with a small bias towards getting too large residuals.

In Table 2 we compare the inlier ratios obtained through exact and approximate residuals, using the ground-truth pose. We compute average inlier ratios for each of the 1500 image pairs in MegaDepth-1500. Note that we compute the ratio of inlier *clusters* for  $\varepsilon_a$ , while for the exact residual  $\varepsilon_S$  the inlier ratio is computed per *match*. We also include the runtime for computing the two residuals for an image pair (equivalent to scoring a model in RANSAC). We see that the approximated residual gives slightly lower inlier ratios, which decrease slowly for coarser clustering, while the speedup in computing time is significant.

#### 4.4. Ablation on Integration in RANSAC

In this section, we explore different ways of integrating our method in robust estimation with RANSAC. Robust estimation is typically done in two stages: the hypothesize-and-verify loop, which alternates between estimating and scoring models, and model refinement, which is done on the best model after the stop criterion is reached.

We evaluate the effect of our summarization scheme both on the model scoring and refinement stages of the estimation. For both stages, we evaluate usage of the representative match residuals (4), and the approximate residuals (12). We will refer to these methods as *Center* and *Approximate*, respectively. For the refinement stage, we also evaluate usage of the original residuals (3), referred to as the *Dense*



**Figure 5. Histogram over Approximation Error.** The graph shows the distribution of differences between the true and approximated residuals, *i.e.*  $\epsilon_S - \epsilon_a$ , scaled with focal length. Statistic taken over all clusters from all image pairs in MegaDepth-1500.

Type of residual	Inlier ratio		Runtime ( $\mu\text{s}$ )	
	Med.	Avg.	Avg.	Speedup
Exact	0.87	0.82	155.8	1.0x
Approx. $K = 1024$	0.85	0.80	31.1	5.0x
Approx. $K = 512$	0.84	0.80	14.8	10.5x
Approx. $K = 256$	0.84	0.79	7.1	21.9x
Approx. $K = 128$	0.82	0.78	3.2	48.7x
Approx. $K = 64$	0.80	0.76	1.3	119.8x

**Table 2. Comparison of RANSAC Scoring.** We compare the inlier ratios when calculating exact and approximate residuals for different numbers of clusters  $K$ , using the ground-truth essential matrix on all image pairs in MegaDepth-1500. The average and median ratios over all image pairs is reported. Note that inlier ratios are calculated per match for the exact residuals, and per cluster for approximate residuals. The runtime shows the average cost of scoring a model with MSAC in RANSAC.

method. With two model scoring costs to evaluate (*Center* and *Approximate*) and three refinement costs (*Center*, *Approximate*, and *Dense*) this results in 6 possible combinations for our ablation study, in addition to the fully dense baseline. However, we omit *Approximate* scoring followed by *Center* refinement, since this would use less information in the refinement than what was used for the model scoring.

For all of the above methods, we perform model estimation using the 5-point solver [31], sampling from the representative matches. Note that this sampling has no effect on runtime compared to sampling from the dense matches, since convergence of RANSAC is only based on the scoring. Since we get up to 9 constraints from each  $M_k$  matrix, it is in principle also possible to estimate a model from a single summarized correspondence. However, we found that this generally gives worse performance, see suppl. material.

In Table 3, we show AUC@ $5^\circ$  and median runtime for the different combinations of model scoring and refinement costs, compared with a fully dense baseline. We report the average and standard deviation of the AUC for 10 runs using different seeds. In the table, we abbreviate the methods using a three letter combination, where the first letter denotes sampling method, the second letter denotes scoring method, and the third letter denotes refinement method. We see that scoring using *Approximate* or *Center* residuals are both significantly faster than *Dense*. *Center* scoring is the fastest while not significantly less accurate than *Approximate*. For refinement, however, *Approximate* residuals give a small but significant improvement in score over *Center*, but at an increased runtime. This gives us a possible trade-off, where we can sacrifice some runtime for better accuracy.

We also compare our ablated methods with a dense baseline on subsampled DKM-matches, obtained by querying the balanced DKM sampler for a lower number of correspondences. In Figure 6, we plot AUC@ $5^\circ$  vs. runtime for our methods and different number of DKM matches. We see that compared to subsampling, most of our methods give shorter runtime for the same performance; in particular CCC and CCA. This indicates that our summarization better preserves the geometric constraints, compared to reducing the number of correspondences extracted from the matcher.

Method	AUC@ $5^\circ$	RT (ms)	Speedup
DDD (baseline)	$67.38 \pm 0.10$	66.0	1.0x
CAD	$67.21 \pm 0.08$	6.8	9.8x
CCD	$67.12 \pm 0.08$	5.3	12.3x
CAA	$66.87 \pm 0.07$	2.7	24.6x
CCA	$66.70 \pm 0.06$	1.5	45.2x
CCC	$65.95 \pm 0.10$	<b>1.2</b>	<b>55.0x</b>

**Table 3. Ablation on RANSAC Integration.** We compare usage of our two summarization schemes in different stages of the estimation. Method names denote the what data is used in sampling, scoring, and refinement. For example, ‘‘CAD’’ means sampling from representative matches (C), scoring with the summarized approximation (A), and refinement with the dense matches (D).

#### 4.5. Comparative Evaluation in RANSAC

We evaluate our match summarization scheme on robust estimation using different dense matchers. We run both the CCC- and CCA-methods, since they represent a possible trade-off between accuracy and runtime. We compare with a fully dense baseline on the full set of matches. To show that our results do not depend on the dense matcher, we report results using DKM [15], RoMA [16], and ASpanFormer [7]. From DKM and RoMA, we always sample 10 000 matches, while ASpanFormer gives a variable

Matches	Estimator	MegaDepth-1500					ScanNet-1500				
		AUC@5°	AUC@10°	AUC@20°	RT (ms)	Speedup	AUC@5°	AUC@10°	AUC@20°	RT (ms)	Speedup
DKM	Dense	67.4	79.6	88.0	66.6	1.0x	31.3	52.6	69.9	68.8	1.0x
	Ours CCA	66.7	79.1	87.5	1.5	45.2x	30.9	52.1	69.6	1.5	46.7x
	Ours CCC	66.0	78.7	87.3	1.2	<b>55.0x</b>	30.4	51.7	69.3	1.2	<b>58.6x</b>
RoMA	Dense	70.1	81.5	89.3	60.6	1.0x	33.3	55.2	72.2	64.6	1.0x
	Ours CCA	70.1	81.6	89.4	1.4	42.2x	33.0	54.9	72.0	1.4	46.3x
	Ours CCC	69.3	81.1	89.2	1.2	<b>52.1x</b>	32.8	54.7	71.8	1.1	<b>58.3x</b>
ASpanFormer	Dense	66.3	78.8	87.5	14.4	1.0x	30.3	50.5	66.5	15.2	1.0x
	Ours CCA	65.8	78.5	87.4	1.5	9.6x	29.7	49.9	66.1	1.5	10.3x
	Ours CCC	64.1	77.4	86.8	1.2	<b>12.1x</b>	28.9	49.2	65.7	1.2	<b>13.0x</b>
SP+LG	Dense	63.7	77.2	86.5	5.0	N/A	21.8	39.6	55.6	3.0	N/A

Table 4. **Comparison with State-of-the-art.** We compare our method with a dense baseline for several dense matchers on both MegaDepth-1500 and ScanNet-1500. For comparison, we also include AUC and runtime for state-of-the-art sparse matches (SP+LG).

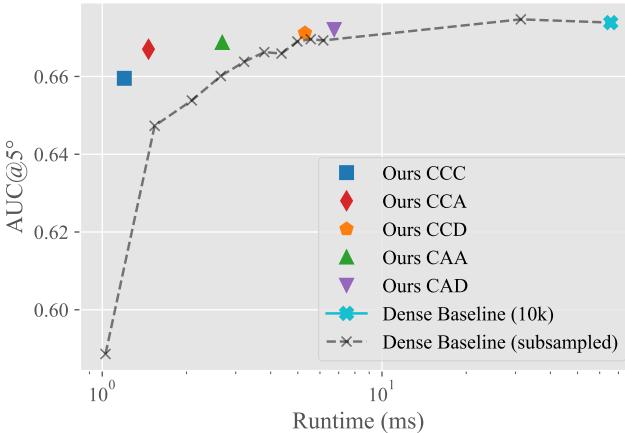


Figure 6. **AUC-Runtime Trade-off in Ablation Study.** The methods in Table 3 are plotted together with dense baseline for different samplings from DKM.

amount of matches. For image pairs where ASpanFormer gives less than 128 matches, we fall back to using dense estimation. The average results from 10 different seeds are shown in Table 4. The reported runtime is the median over all image pairs. For the convenience of the reader, we have also included estimation from sparse SuperPoint [12] + LightGlue [27] keypoints. In summary, we see that our summarization scheme achieves comparable accuracy to the dense baseline at a fraction of the runtime, irrespective of the dense keypoint matcher used.

Finally, we present an experiment on the WxBS benchmark [29]. This benchmark contains extremely challenging wide-baseline image pairs, where GT-correspondences are provided. As it does not provide intrinsics, we evaluate fundamental matrix estimation. The metrics are computed by checking consistency of the estimated  $F$  with the GT matches. For comparison, we also run on dense matches extracted with MAST3R [25], where we disabled subsampling in the fast reciprocal matching. The results in Table 5

Matches	Estimator	Recall (10px)↑	RT (ms)↓	Speedup↑
RoMa	Dense	86.3	63.0	1.0x
	Ours CCA	85.1	1.45	43.3x
	Ours CCC	83.8	<b>0.82</b>	<b>77.0x</b>
MASt3R	Dense	63.4	37.1	1.0x
	Ours CCA	63.2	1.43	25.8x
	Ours CCC	61.6	<b>0.76</b>	<b>49.6x</b>

Table 5. **Fundamental Matrix Estimation on WxBS.**

are qualitatively similar to the calibrated setting, showing that our method significantly speeds up the robust estimation with a marginal loss in accuracy.

## 5. Conclusion

In this work, we have shown that dense keypoint matches, while they improve estimation compared to sparse matches, contain significant redundancies. This redundancy makes it possible to heavily subsample the matches while keeping most of the geometric constraints. We demonstrated a method to significantly speed up geometric estimation with little effect on the estimation accuracy. We additionally showed that this reduction in accuracy can be partially recovered by using our proposed summarized meta-correspondences. With this scheme, we were able to reduce the number of residuals per image pair from the number of matches  $N$ , to  $9K$  where  $K$  is the number of clusters – a hyperparameter we can choose such that  $N \ll 9K$ . We also get a potential compression ratio in terms of storage, since the constraints from each cluster is entirely contained in a  $9 \times 9$ -matrix, plus a single correspondence. A limitation is that the method only makes sense if the original number of matches is very large. Another limitation of the paper is that we have only focused on speeding up the robust estimation step, while a large part of the total runtime comes from the dense matcher itself.

**Acknowledgments.** The project was supported by ELLIIT and the Swedish Research Council (Grant No. 2023-05424).

## References

- [1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012. 5
- [2] Daniel Barath and Gábor Valasek. Space-partitioning ransac. In *European Conference on Computer Vision (ECCV)*, 2022. 2
- [3] Daniel Barath, Jiri Matas, and Jana Noskova. MAGSAC: marginalizing sample consensus. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 2
- [4] Axel Barroso-Laguna, Eric Brachmann, Victor Adrian Prisacariu, Gabriel J Brostow, and Daniyar Turmukhambetov. Two-view geometry scoring without correspondences. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [5] Jacob Bentolila and Joseph M Francos. Conic epipolar constraints from affine correspondences. *Computer Vision and Image Understanding (CVIU)*, 2014. 2
- [6] Luca Cavalli, Daniel Barath, Marc Pollefeys, and Viktor Larsson. Consensus-adaptive ransac. *arXiv preprint arXiv:2307.14030*, 2023. 2
- [7] Hongkai Chen, Zixin Luo, Lei Zhou, Yurun Tian, Mingmin Zhen, Tian Fang, David McKinnon, Yanghai Tsin, and Long Quan. Aspanformer: Detector-free image matching with adaptive span transformer. *European Conference on Computer Vision (ECCV)*, 2022. 2, 7
- [8] O. Chum and J. Matas. Matching with prosac - progressive sample consensus. In *Computer Vision and Pattern Recognition (CVPR)*, 2005. 2
- [9] Ondřej Chum and Jiří Matas. Optimal randomized ransac. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 2008. 2
- [10] Ondřej Chum, Jiří Matas, and Josef Kittler. Locally optimized ransac. In *Pattern Recognition*, pages 236–243, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 2
- [11] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 4
- [12] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018. 2, 8
- [13] Matthijs Douze, Alexandre Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvassy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *arXiv preprint arXiv:2401.08281*, 2024. 5
- [14] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint description and detection of local features. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 2
- [15] Johan Edstedt, Ioannis Athanasiadis, Mårten Wadenbäck, and Michael Felsberg. DKM: Dense kernelized feature matching for geometry estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 4, 7
- [16] Johan Edstedt, Qiyu Sun, Georg Bökmán, Mårten Wadenbäck, and Michael Felsberg. RoMa: Robust Dense Feature Matching. *Computer Vision and Pattern Recognition (CVPR)*, 2024. 2, 7
- [17] Iván Eichhardt and Dmitry Chetverikov. Affine correspondences between central cameras for rapid relative pose estimation. In *European Conference on Computer Vision (ECCV)*, 2018. 2
- [18] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 1, 2
- [19] Levente Hajder, Lajos Lóczy, and Daniel Barath. Fast globally optimal surface normal estimation from an affine correspondence. In *International Conference on Computer Vision (ICCV)*, 2023. 2
- [20] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. Cotr: Correspondence transformer for matching across images. In *International Conference on Computer Vision (ICCV)*, 2021. 2
- [21] Alchan Kim. Fast-slic. <https://github.com/Algy/fast-slic>, 2019. 5
- [22] Simon Korman and Roee Litman. Latent ransac. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 2
- [23] Viktor Larsson and contributors. PoseLib - Minimal Solvers for Camera Pose Estimation. <https://github.com/vlarsson/PoseLib>, 2020. 4
- [24] Karel Lebeda, Jiri Matas, and Ondrej Chum. Fixing the locally optimized ransac—full experimental evaluation. In *British Machine Vision Conference (BMVC)*, 2012. 2
- [25] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European Conference on Computer Vision (ECCV)*, 2024. 8
- [26] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 4
- [27] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys. LightGlue: Local Feature Matching at Light Speed. In *International Conference on Computer Vision (ICCV)*, 2023. 2, 8
- [28] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60:91–110, 2004. 2
- [29] Dmytro Mishkin, Jiri Matas, Michal Perdoch, and Karel Lenc. WxBS: Wide baseline stereo generalizations. In *British Machine Vision Conference (BMVC)*, 2015. 4, 8
- [30] Kai Ni, Hailin Jin, and Frank Dellaert. Groupsac: Efficient consensus in the presence of groupings. In *International Conference on Computer Vision (ICCV)*, 2009. 2
- [31] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, 2004. 7
- [32] Martin Rais, Gabriele Facciolo, Enric Meinhardt-Holzapfel, Jean-Michel Morel, Antoni Buades, and Bartomeu Coll. Accurate motion estimation through random sample aggregated consensus. *arXiv preprint arXiv:1701.05268*, 2017. 2

- [33] Carolina Raposo and Joao P Barreto. Theory and practice of structure-from-motion using affine correspondences. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. [2](#)
- [34] A. L. Rodríguez, P. E. López-de Teruel, and A. Ruiz. Reduced epipolar cost for accelerated incremental SfM. In *Computer Vision and Pattern Recognition (CVPR)*, 2011. [4](#)
- [35] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. [2, 4](#)
- [36] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *Computer Vision and Pattern Recognition (CVPR)*, 2021. [2, 4](#)
- [37] P.H.S. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding (CVIU)*, 78(1): 138–156, 2000. [2](#)
- [38] Jonathan Ventura, Zuzana Kukelova, Torsten Sattler, and Dániel Baráth. P1ac: Revisiting absolute pose from a single affine correspondence. In *International Conference on Computer Vision (ICCV)*, 2023. [2](#)
- [39] Tong Wei, Jiri Matas, and Daniel Barath. Adaptive reordering sampler with neurally guided magsac. In *International Conference on Computer Vision (ICCV)*, 2023. [2](#)

# Dense Match Summarization for Faster Two-view Estimation

## Supplementary Material

Sampling	Scoring	Refinement	AUC			Runtime (ms)		
			5°	10°	20°	Med.	Avg.	Speedup
Center	Dense	Dense	67.38 $\pm$ 0.10	79.61 $\pm$ 0.08	87.96 $\pm$ 0.06	66.0	79.0	1.0x
	Approx.	Dense	67.21 $\pm$ 0.08	79.65 $\pm$ 0.06	88.12 $\pm$ 0.05	6.8	9.3	9.8x
	Center	Dense	67.12 $\pm$ 0.08	79.36 $\pm$ 0.08	87.68 $\pm$ 0.09	5.3	6.9	12.3x
	Approx.	Approx.	66.87 $\pm$ 0.07	79.43 $\pm$ 0.05	87.98 $\pm$ 0.05	2.7	4.4	24.6x
	Center	Approx.	66.70 $\pm$ 0.06	79.09 $\pm$ 0.08	87.55 $\pm$ 0.08	1.5	2.2	45.2x
	Center	Center	65.95 $\pm$ 0.10	78.75 $\pm$ 0.07	87.33 $\pm$ 0.09	<b>1.2</b>	<b>1.9</b>	<b>55.0x</b>
Approx.*	Approx.	Dense	64.70 $\pm$ 0.00	76.79 $\pm$ 0.00	85.08 $\pm$ 0.00	8.4	9.7	7.9x
	Approx.	Approx.	63.14 $\pm$ 0.00	75.27 $\pm$ 0.00	83.76 $\pm$ 0.00	4.2	4.3	15.8x

Table 6. **Ablation on Integration in RANSAC.** We compare usage of our two summarization schemes (Center, Approx.) in different stages of RANSAC. \*For sampling and model estimation with approximated sampling, we modify RANSAC to sample exhaustively.

## 6. Additional Ablation Results

In Table 6, we include some additional results on the ablation on RANSAC integration to what was presented in the main paper. In addition to AUC@5°, we also present AUC@10° and AUC@20°. In addition to the median estimation runtime, we also present the average runtime per image pair and seed.

We also include sampling and model estimation from the summarized correspondences, using the  $M_k$  matrix. Using SVD, we find a 4-dimensional approximate nullspace to  $M_k$  and use as basis in the 5-point solver. Since this allows us to estimate a model from a single summarized correspondence, reducing the number of unique minimal samples to  $K$ , we modify RANSAC to sample exhaustively instead of randomly for this method. We refer to this method as *Approximated sampling*.