SiLK: Simple Learned Keypoints

Pierre Gleize
gleize@meta.com

Weiyao Wang
weiyaowang@meta.com

Matt Feiszli
mdf@meta.com

Meta AI
https://github.com/facebookresearch/silk

Abstract

*Keypoint detection & descriptors are foundational technologies for computer vision tasks like image matching, 3D reconstruction and visual odometry. Hand-engineered methods like Harris corners, SIFT, and HOG descriptors have been used for decades; more recently, there has been a trend to introduce learning in an attempt to improve keypoint detectors. On inspection however, the results are difficult to interpret; recent learning-based methods employ a vast diversity of experimental setups and design choices: empirical results are often reported using different backbones, protocols, datasets, types of supervisions or tasks. Since these differences are often coupled together, it raises a natural question on what makes a good learned keypoint detector. In this work, we revisit the design of existing keypoint detectors by deconstructing their methodologies and identifying the key components. We re-design each component from first-principle and propose **Si**mple **L**earned **K**eypoints (SiLK) that is fully-differentiable, lightweight, and flexible. Despite its simplicity, SiLK advances new state-of-the-art on Detection Repeatability and Homography Estimation tasks on HPatches and 3D Point-Cloud Registration task on ScanNet, and achieves competitive performance to state-of-the-art on camera pose estimation in 2022 Image Matching Challenge and ScanNet.*

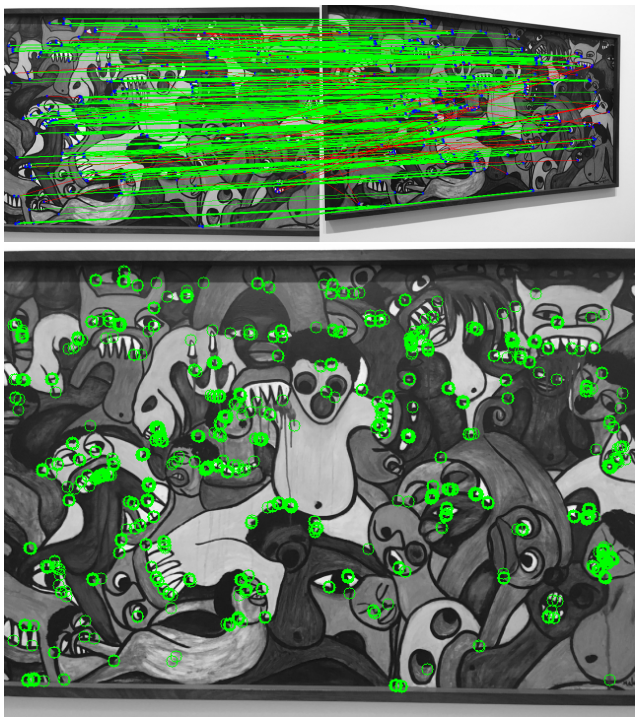


Figure 1: The top image is an example of keypoint matching under viewpoint change; correct matches are green, while incorrect ones are red. The bottom image shows keypoints which are cycle-consistent by SiLK. As can be observed, SiLK has learned to find distinctive geometric features (corners, curves, intersections,..); from single non-annotated images.

1. Introduction

Keypoint detection and matching is a foundational computer vision technique to obtain a sparse yet informative representation of an image or video sequence. Image stitching [7, 1], SLAM [14, 34], SfM [44], camera calibrations, tracking [37], and object detection [30] are important tasks which have been built on keypoint correspondences [32]. For a given task, a good keypoint representation should be able to identify a small set of points which are useful and informative for the task. One typically also wants robustness of the descriptor to some set of transformations, like scale, point of view, or lighting variation. In this writeup we are primarily interested in keypoints for 3D geometric problems like SLAM and SfM, which rely on finding correspondences between pairs of images. For such problems, good keypoints should contain enough information to maximize the probability of correct matching from different points of

view in varying conditions.

Our contributions are twofold.

1. After reviewing a number of alternative approaches we propose Simple Learned Keypoints (SiLK), designed to be the simplest possible self-supervised approach to learn distinctive and robust keypoints from arbitrary image data in a traditional "detect-and-describe" framework. Despite its simplicity, SiLK is competitive or surpasses SOTA in most settings.
2. Leveraging SiLK's simple one-stage training protocol and modular architecture, we are able to ablate various dimensions of detector performance for different tasks. In particular, with an eye toward real-time performance, we identify tasks where extremely lightweight backbone architectures are sufficient.

2. Related Work

Early work focused on carefully engineered methods to identify distinctive keypoints with descriptors which are robust to changes such as viewpoint and illumination. Hand-crafted techniques like Harris-corners[20], SIFT[31], ORB[41] and others [40, 8, 26, 2, 5, 13] have used explicit geometric notions like corners, gradients, and scale-space extrema to achieve results which remain both efficient and competitive to date [44, 34].

More recent work like SuperPoint[16] chose to learn to find corners; they generate a large set of synthetic shapes with annotated corners and train a model with this ground truth. While providing compelling evidence for learned methods, their training procedure is quite complex: it contains multiple training phases, a synthetic dataset, and employs a homography adaptation trick that can be difficult to tune (see our reproduced results in Tab. 2).

In the same spirit as more recent work [17, 39, 50, 49, 21, 11, 48], SiLK aims to learn keypoints in simple end-to-end fashion, without explicitly defining them as corners.

Several attempts have been made to learn keypoints implicitly; either by the careful design of the loss [17, 39, 11, 48]; or by directly predicting the matching success of descriptors [50, 21, 49]. SiLK falls in the second category, but with slight twist (cf. Sec. 3.4.4).

To learn descriptors, contrastive losses are commonly used. Similar to [50, 47], SiLK adopts a probabilistic approach by modeling the matching probabilities in a double-softmax, cycle-consistency setting and optimizes the log likelihood. The probabilistic formulation, similar to InfoNCE [35], gives us a clean way to reason about matching, and the abundant supply of hard negative examples (from pixels in the same image) makes it an attractive choice.

Context aggregation (CA) is a recent addition to the toolkit. Initiated by SuperGlue[42], CA aims to refine or transform descriptors from a *pair* of images before matching them. Implemented as a GNN[43] in [42], or as a Transformer[51] in [47], CA's predictions are conditional on all descriptors from the pair of images. In other words, for an image, the descriptors will be different when matching against different images. As a result, CA needs to run on every pair of images prior to matching, as opposed to running on single images. The run-time implications render CA prohibitively expensive in some applications (quadratic versus linear complexity). SiLK does not use CA, but outperforms [42] and performs competitively with [47]. Incorporating CA is optional for future works if performance is paramount and computation cost is less of a concern.

As a postprocessing after CA, SuperGlue[42] introduced the concept of differentiable optimal transport (OT) to improve matching, using the Sinkhorn algorithm[46]. LoFTR [47] leveraged OT as well, but found little difference between OT and the simpler approach of mutual nearest neighbor (MNN) in some benchmarks.

3. Methodology

SiLK's contribution is simplicity and flexibility. Our solution is built on the traditional approach of identifying distinctive pixels via robust local descriptors. We use modern but established techniques to learn to localize and describe keypoints given an arbitrary source of unlabeled images. Unlike classical methods, our descriptors and invariances are learned, and unlike some modern methods, there is no particular complexity in the matching process (SiLK employs only cosine distances and mutual nearest neighbor); this leaves few structural hyperparameters to tune. The simple backbone+heads design is backbone-agnostic, allowing experimentation. The annotation-free SSL approach means SiLK can be trained on any image or video dataset. Finally, a simple, one-stage training pipeline allows us to easily train and ablate different architectures, datasets, and hyperparameters for different tasks.

SiLK is trained to identify keypoints from single grayscale images. It provides both keypoint detections (location) and keypoint descriptors (for matching). Cycle consistency is employed for descriptor learning and a binary classifier identifies distinctive keypoints at pixel-level.

To learn descriptors, we take a source image and a transformed copy, extract descriptors for each point, and use descriptor similarity to define transition probabilities from a source location to each transformed location (and vice-versa). We optimize the descriptors to maximize cycle-consistency; i.e. we maximize the probability of a round-trip from the source to its transformed location and back.

To locate good keypoints, we train a binary classifier to identify points which will satisfy a matching criterion. A point and its transformed copy are positives when they are mutual nearest neighbors in the sense of transition proba-

	Keypoint Detection					Descriptors & Matching			Model & Training			
	Learned	Sparse	Cell-based	NMS	Supervision	CA	Matcher	Supervision	Input	Backbone	Data	E2E
SIFT	No	✓	**No**	✓	-	**No**	**MNN**	-	-	-	-	-
SuperPoint	✓	✓	✓	✓	Homo. Adapt.	**No**	**MNN**	**Rand. Homo.**	**Grey**	VGG	COCO	No
SuperGlue	-	✓	-	-	-	GraphNN	OT	SfM	**Grey**	VGG	Oxford & Paris	No
GLAMPoints	✓	✓	**No**	✓	**Matching Succ.**	**No**	**MNN**	-	**Grey**	UNet	SlitLamp	✓
D2-Net	✓	✓	✓	✓	Triplet Ranking	**No**	**MNN**	SfM	RGB	VGG	MegaDepth	✓
R2D2	✓	✓	✓	✓	Matching AP	**No**	**MNN**	SfM	RGB	L2-Net	Aachen	✓
DISK	✓	✓	✓	✓	**Matching Succ.**	**No**	**MNN**	SfM	RGB	UNet	MegaDepth	✓
URR	-	No	-	-	-	-	**MNN**	3D Rendering	RGB	ResNet	ScanNet	✓
LoFTR	-	No	-	-	-	Transformer	**MNN**,OT	SfM	**Grey**	FPN	ScanNet,MegaDepth	✓
SiLK	✓	**Optional**	**No**	**No**	**Matching Succ.**	**No**	**MNN**	**Rand. Homo.**	**Grey**	**Generic**	**Any Image Set**	✓

Table 1: **Non-exhaustive deconstruction of keypoint detectors along different dimensions.** On each, SiLK adopts the simplest choice or is flexible to different choices. In particular, SiLK does not depend strongly on backbone type or training data (Tab. 7 & Tab. 9).

bilities, and they are negatives otherwise. We train both the cycle-consistency and classification losses jointly.

We provide simple pseudo-code in Fig. 2.

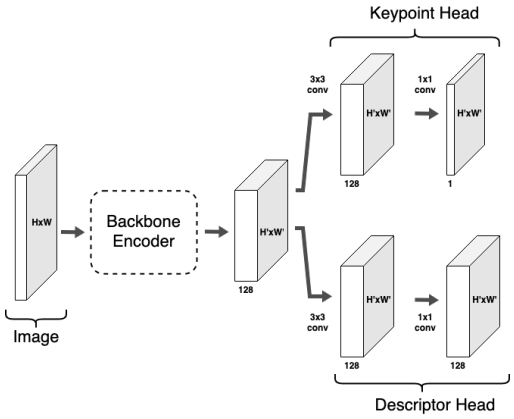
```
def loss(image_0, model):
    # get warped image and pixel correspondences
    image_1, corr_0, corr_1 = rand_homo(image_0)

    # apply image augmentations
    image_0 = augment(image_0)
    image_1 = augment(image_1)

    # extract dense descriptors and keypoints
    desc_0, kpts_0 = model(image_0)
    desc_1, kpts_1 = model(image_1)

    # compute similarity matrix
    sim_mat = cosim(desc_0, desc_1)

    # compute the descriptor loss
    # using ground truth correspondences
    loss_desc = nll(sim_mat, corr_0, corr_1)

    # measure matching success
    y = is_match_success(sim_mat, corr_0, corr_1)

    # compute keypoint loss
    # using current matching success
    loss_kpts = bce(kpts_0, y, corr_0)
    loss_kpts += bce(kpts_1, y, corr_1)

    return loss_desc + loss_kpts
```

Figure 2: Pseudo-code: learning keypoints from a single image.

3.1. Architecture

The SiLK architecture (Fig. 3) is inspired by the "detect-and-describe" architecture originally proposed by SuperPoint [16]. A dense feature map is first extracted by feeding an image to an encoder backbone. The shared feature map is then fed to two heads (a keypoint head and a descriptor head).

Figure 3: Architecture for SiLK



- The *keypoint head* extracts the logits; used to calculate the dense keypoint probabilities.
- The *descriptor head* extracts a dense descriptor map; used later to calculate keypoint similarities.

The model is backbone-agnostic and can easily be swapped.

3.2. High Matching Probability Defines Keypoints

As mentioned above, the keypoint probability estimate predicts the probability that a pixel will be correctly matched (i.e. survive a round-trip). Points with the highest likelihood of matching correctly are exactly those which we select as keypoints.

A common approach [16, 50, 39, 17] for obtaining keypoint probabilities is to use a softmax cell-based approach. A cell is a $N \times N$ patch in which the probability of each cell position is determined by a local softmax. The softmax operates on $N \times N + 1$ bins; $+1$ being the *dustbin*, handling the case of cells devoid of keypoints.

SiLK's approach is equivalent to a cell-based approach with a cell size of $N = 1$. This has several consequences. First, the softmax formulation becomes a sigmoid $\sigma(x) = \frac{1}{1+e^{-x}}$. Second, it removes the sparsity constraint that keypoints are exclusive events inside a cell. And third, this removes a free parameter (the cell size N) that we do not have to later tune.

In the same spirit, SiLK does not use NMS during inference. Even though NMS is an established pruning technique that aims to spread out keypoints [16], we find SiLK doesn't need NMS to perform (cf. Tab. 2).

3.3. Descriptors Define Matching Probability

Similar to [47, 50], we model the cycle matching probability using a double softmax (i.e. the probability of matching i to j, and back).

$$P_{i \leftrightarrow j} = P_{i \rightarrow j} P_{i \leftarrow j}$$

where $P_{i \rightarrow j}$ is the directional probability of matching the ith descriptor from image I to the jth descriptor in image I'. $P_{i \leftarrow j}$ is similar, but in the reverse direction. Both forward and backward probabilities are modeled as a softmax with temperature over descriptor cosine similarities. For fixed i, $P_{i \rightarrow j}$ is a softmax over the i-th row of the descriptor similarity matrix, and $P_{i \leftarrow j}$ takes softmax over column j.

3.4. Training

3.4.1 Self-Supervision

Pixel-accurate correspondences are required during training. Similar to SuperPoint [16], we obtain image-pair correspondences by applying a random transformation (homography) to an image. However, the homography is a linear mapping that gives correspondences at a subpixel level.

To obtain pixel correspondences, we apply the sampled homography to all the pixel positions of image I; positions being the center of pixels (e.g. the top-left pixel has position $(0.5, 0.5)$). This first step establishes dense directional correspondences from I to I'. We then run the same process from I' to I (using the inverse homography this time). Once both directional correspondences are obtained, the resulting positions are discretized; out-of-bound and non-bijective correspondences are discarded.

3.4.2 Image Augmentations

As done in [16, 42], we employ image augmentation to improve robustness; augmentations include random brightness, contrast, gaussian noise, speckle noise and motion blur. We refer to supplementary materials for details.

3.4.3 Negative and Positive Selection

One defining property of a keypoint is *distinctiveness* – the point can be reliably distinguished from its peers. In our case this means the point can be reliably identified in a matching algorithm, similar to [49, 50]. With that in mind, we adopt a similar supervision technique as [49]. Keypoints that are correctly matched (using the currently trained descriptors) are labelled as positive; otherwise negative.

3.4.4 Descriptor and Keypoint Losses

Similar to [47, 50], the descriptor loss is the negative log-likelihood loss applied to the matching probabilities for the positive round-trips (i.e. paths from point i to its location i' in the transformed image and back again).

$$\mathcal{L}_{desc} = -\left(\log P_{i \rightarrow i'} + \log P_{i \leftarrow i'}\right)$$

This implicitly penalizes non-positive paths via softmax. One might notice $\mathcal{L}_{desc}$ requires the computation of a large matrix (size $HW \times HW$). To handle potential GPU out-of-memory, we provide a simple, yet efficient implementation which computes the similarity matrix in a block-wise fashion, and recomputes block dot products instead of storing them for backpropagation (in the same vein as [36, 25]).

The keypoint loss $\mathcal{L}_{key}$ is a simple binary cross-entropy loss applied to a logistic sigmoid, in contrast to [49]. It is trained to identify keypoints with successful round-trip matches (defined by mutual-nearest-neighbor) among all others (unsuccessful).

4. Experiments

In this section, we empirically evaluate SiLK together with representative baselines and state-of-the-art methods. On HPatches we evaluate a suite of complementary keypoint quality metrics (Repeatability, Mean Matching Accuracy [33]) and planar stereo estimation capabilities (Homography Estimation). In addition, we benchmark on three real-world stereo tasks: outdoor camera pose estimation on Image Matching Challenge (IMC) 2022, and both indoor camera pose estimation and 3D point-cloud registration on ScanNet. In these experiments, we study the following:
(1) Many methods employ complex strategies to learn and predict good keypoints and descriptors, including elements like multi-stage training, cell-based priors, complicated post-processing, context aggregation, and groundtruth 3D pose supervision (see Tab. 1), in various combinations. *What machinery is necessary?* SiLK contains no such machinery, and can be viewed as a reduction from these methods. However, SiLK either achieves new SOTA or compares very favorably. This questions the need for complex schemes for the evaluated tasks.

(2) We observed rather strong performance from engineered features (e.g. SIFT in Tab. 2&Tab. 6) vs learned methods. This motivates us to revisit design choices in a learned keypoint detector: *what makes a good keypoint detector?* We ablate each model component (data, backbone, etc.) and test generalization performance under various conditions (e.g. input size, test data, task, etc). SiLK proves very robust to these choices (Sec. 4.5). In particular, a very lightweight version of SiLK (two 3x3 convolution layers) is competitive to SOTA on homography estimation, camera pose estimation and point cloud registration (Tab. 7&Tab. 8).

We hope these results can serve the community and help adapt keypoint models to their tasks and needs. For example, labeling tasks (e.g. self-training [37], object pose [38]) might focus on high accuracy (i.e. larger backbone and denser keypoint selection), while tasks requiring speed (e.g. SLAM [34]) might find our lightweight backbone attractive.

4.1. Implementation Details

Our own training pipeline has been used for all experiments with SiLK, as well as our reproduced SuperPoint (Tab. 2) results. Training time is ~5 hours on 2 Tesla V100-SXM2 GPUs using our default setup.

Default Setup. Unless specified otherwise, all results use the following setup. Trained on COCO [29] images (randomly sampled), with Adam [24] optimizer with learning rate $1e^{-4}$ and betas $(0.9, 0.999)$; trained for 100k iterations; batch size of 1 per GPU; dense descriptor map is 146×146 for all architectures and input resolutions; cosine similarities scaled by temperature 20^{-1}; VGGnp-4 backbone (VGG architecture **with max-pooling removed**, details in Sec. 4.5.1); sparse keypoints obtained with top-k ($k = 10000$); detection head is 1 3x3 convolution (128-dims), and 1 1x1 convolution; descriptor head is 1 3x3 convolution (128-dims), and 1 1x1 convolution (128-dims out); no padding in convolutions; ReLU and batchnorm used as non-linearity and normalization (see supplementary).

4.2. HPatches Homography Estimation

Following [16, 42, 39, 47], we evaluate homography estimation on HPatches [3]. HPatches contains 57 scenes (of 6 images) with significant illumination changes and 59 scenes with large viewpoint variations. Images in each scene are related by groundtruth homographies. We follow LoFTR [47], (currently SOTA on HPatches), and and scale the shorter image edge to 480 at inference time.

Evaluation Protocol. For every pair of images, the model detects a set of keypoints. These keypoints are desired to be distinctive and therefore repeatably detected across views. We use *Repeatability* to evaluate detection performance as in [16]. To test invariance of keypoint descriptors, each model's preferred matching algorithm establishes correspondence across images to obtain a subset of keypoints. We distinguish this subset as *post-matching* and the entire set as *pre-matching*. The accuracy of each correspondence is evaluated by *Mean Matching Accuracy* as in [17, 39, 50]. Finally, we use OpenCV RANSAC algorithm to compute homography based on matching results and evaluate *Homography Estimation Accuracy* [16] and *Homography Estimation AUC* [42, 47].

Baselines. We compare SiLK against both sparse detector-based methods and dense detector-free methods. The sparse detector-based methods generally follow "detect *then* match": the model first detects a sparse set of distinctive points, then matches features. We include SIFT [31] as well as learned detectors SuperPoint [16] (both official release and our repro), R2D2 [39] and DISK [50]. On the other hand, dense detector-free perform "detection *by* matching": the model first extracts features, then applies a learned pre-matching CA module (e.g. GNN [42] or transformer [47]) to adapt the features to a specific pair of images, and then finds matches. While SiLK does not employ CA, we still include comparisons to the SOTA detector-free LoFTR [47].

Results. Despite its simplicity, SiLK outperforms all methods on repeatability, homography accuracy and homography estimation (Tab. 2). In particular, SiLK has a strong margin when the error threshold is small ($\epsilon = 1$). This validates our pixel-accurate keypoint localization. SiLK lags only on the MMA@3 metric. This may be caused by our pixel accurate contrastive loss, which amplifies local descriptor differences and makes local matching errors less likely. Consequently, SiLK does not benefit from increasing error threshold in MMA. In addition, even vs LoFTR (which uses dense features and CA) in Tab. 3, SiLK shows strong performance on Homography Estimation AUC and competitive performance on Homography accuracy. This questions the necessity of CA for these particular tasks; this may be valuable in applications which are particularly sensitive to runtime performance.

4.3. IMC 2022 outdoor pose estimation

The Image Matching Challenge (IMC 2022) [22] provides pairs of outdoor images from different viewpoints; participants are required to estimate the fundamental matrix. Camera pose accuracy is then computed for ten thresholds of rotation and translation error (ranging from $(1°, 20\text{cm})$ to $(10°, 5\text{m})$). Mean average accuracy (mAA) is reported by averaging across thresholds and scenes.

Evaluation Protocol. Sparse methods, such as DISK and SiLK, detect keypoints in individual images; mutual nearest neighbor is used to select matches from each pair. Methods with CA, such as SuperGlue and LoFTR, directly identify matches from each pair of images. In either case, results are passed to MAGSAC [4] to estimate camera pose. The challenge allows different image sizes and tun-

	Repeatability		Hom. Est. Acc.		Hom. Est. AUC		MMA		# of keypoints	
	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	pre-match	post-match
SuperPoint (MagicLeap)	0.34	0.61	0.43	0.8	0.2	0.51	0.41	0.72	847	499
SuperPoint (Ours)	0.33	0.52	0.48	0.75	0.26	0.52	0.38	0.53	1143	474
SIFT	0.31	0.52	0.6	0.84	0.34	0.61	0.41	0.55	2189	910
R2D2	0.36	0.72	0.45	0.79	0.2	0.5	0.34	0.75	6088	1967
DISK	0.38	0.69	0.45	0.8	0.22	0.52	0.52	**0.84**	3349	1794
SiLK (top-10k)	**0.62**	**0.81**	**0.62**	**0.87**	**0.4**	**0.66**	**0.59**	0.71	10000	4283
SiLK (top-5k)	0.56	0.76	0.6	0.85	0.39	0.64	0.57	0.69	5000	2074
SiLK (top-1k)	0.43	0.61	0.53	0.81	0.32	0.58	0.52	0.63	1000	389

Table 2: **SiLK achieves new SOTA on HPatches compared to other methods with sparse keypoints and features.** Despite its simplicity, SiLK achieves higher performance on all metrics except MMA@3. We include the # of keypoints to ensure a fair comparison.

	Hom. Est. Acc.		Hom. Est. AUC		MMA	
	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$
LoFTR (MegaDepth)	**0.65**	**0.87**	0.37	0.65	**0.64**	**0.91**
LoFTR (ScanNet)	0.24	0.57	0.07	0.33	0.36	0.76
SiLK (top-10k)	0.62	**0.87**	**0.4**	**0.66**	0.59	0.71
SiLK (top-5k)	0.60	0.85	0.39	0.64	0.57	0.69

Table 3: **SiLK achieves competitive performance to SOTA LoFTR on HPatches, despite not using context aggregation.** The top-5k SiLK has similar number of matches compared to outdoor LoFTR. We remark that LoFTR has large generalization gap when training on different types of dataset (indoor vs. outdoor), possibly due to the contexualizer.

	private mAA	public mAA
Sparse Features		
DISK	0.491	0.502
SuperGlue	0.676	0.678
SiLK	**0.685**	**0.684**
Dense Features		
LoFTR (MegaDepth)	**0.735**	**0.721**

Table 4: **SiLK achieves new SOTA for sparse methods on IMC2022 and performs competitively to dense methods with context aggregation.**

ing MAGSAC parameters [22]. We use 30k keypoints and MAGSAC threshold .25.

Baselines. We consider the best leaderboard results from three baselines. (i) DISK is the winner of IMC 2020 and is SOTA among sparse methods. 2022 DISK results are from the IMC team (submission). We also take the best version of (ii) SuperGlue (submission) and (iii) LoFTR (submission) provided by the community.

Results. We coarsely tune SiLK for this task for 30 trials (compared to LoFTR's 200 trials). SiLK again performs competitively (c.f. Tab. 4), outperforming DISK by a significant margin (**+0.19/+0.18** mAA). SiLK also performs favorably compared to SuperGlue, which uses context aggregation and optimal transport matching.

4.4. ScanNet: Indoor Pose & Point Clouds

ScanNet [12] is a large-scale dataset of 1513 indoor scenes of RGB-D images and ground-truth camera poses. Using the official train/val/test split we evaluate both relative camera pose estimation and point-cloud registration. Relative camera pose estimation has been used in multiple previous works [52, 53, 6, 42, 47]. The task is to estimate the essential matrix with RANSAC from point matches in a pair of images. We report pose error AUC at thresholds (5°,10°,20°) using 20k keypoints and RANSAC threshold .5.

This protocol measures translation error in degrees and is known to suffer from scale ambiguity [52]. Angular translation error may be unstable, in particular if the underlying translation (in meters) is small. In response, recent works [19, 18] have introduced a 3D point-cloud registration task, using the ground-truth depth provided in ScanNet. A pair of images 20 frames apart is first sampled. Given this pair, a model predicts point matches. After matching, relative camera pose is estimated. Different from the previous protocol, ground-truth depth and camera intrinsics are now used to align matches in 3D. In addition to relative pose errors (reported separately for translation (cm) and rotation (degree)), the Chamfer distance (in cm) is measured between the registered point cloud and the groundtruth point cloud. We refer to the original papers [19, 18] for details on the evaluation setup and metrics.

4.4.1 Relative pose estimation

Baselines. We compare SiLK with both sparse detector-based methods and dense detector-free methods. For the sparse methods, we consider local feature descriptors (R2D2, SuperPoint) with mutual nearest neighbor (MNN) for matching. SiLK falls into this category. In addition, we consider multiple learned context aggregation methods for matching that operates on SuperPoint, including PointCN [52], OANet [53] and SuperGlue. For detector-free dense methods, we consider DRC-Net [27]

Pose Estimation AUC	@5°	@10°	@20°
Sparse Features			
D2-Net [17] + MNN	5.3	14.5	28.0
SuperPoint [16] + MNN	9.4	21.5	36.4
SuperPoint + PointCN [52]	11.4	25.5	41.4
SuperPoint + OANet [53]	11.8	26.9	43.9
SuperPoint + SuperGlue [42]	16.2	33.8	**51.8**
SiLK + MNN	**18.0**	**34.4**	50.4
Dense Features			
DRC-Net [27]	7.7	17.9	30.5
LoFTR (MegaDepth)	16.9	33.6	50.6
LoFTR (ScanNet)	**21.5**	40.8	57.6

Table 5: **SiLK advances SOTA on relative pose estimation among sparse methods on ScanNet and performs competitively against dense method LoFTR.**

and LoFTR. We include two versions of LoFTR: one trained on MegaDepth with optimal transport post-processing and one trained on ScanNet.

Results. As summarized in Tab. 5, SiLK significantly outperforms D2-Net (**+12.7**) and SuperPoint (**+8.6**) when using mutual nearest neighbor matching. In addition, SiLK outperforms the previous SOTA sparse method SuperGlue, despite its simpler design without context aggregation. SiLK performs similarly to LoFTR trained with MegaDepth. SiLK is only outperformed by LoFTR trained on ScanNet, the same as evaluation data.

4.4.2 Pairwise 3D point-cloud registration

Baselines. We consider three main types of baselines. **(i) Sparse Features + RANSAC** We extract sparse keypoints and their features from off-the-shelf models, and use RANSAC to estimate alignment. This includes SIFT and SuperPoint, and 3D geometry model FCGF [10]. **(ii) Dense Feature Matching** We follow the [18] and select high-quality corresponding pairs using the ratio test, and then solve a weighted Procustes problem [9, 23] to produce alignment. We add a dense version of SuperPoint by discarding the keypoint prediction. We include the current state-of-the-art URR [18] that learns invariant point descriptors through cross-view synthesis. By ignoring the keypoint scoring prediction, SiLK also belongs to this category. The goal is to evaluate the quality of the dense point features. **(iii) Pose/Geometry Supervised** We consider methods that use groundtruth poses to supervise, which are not required in (i) and (ii). These include LoFTR, DGR [9] and 3D MV Registration [19]. For LoFTR, we include only the MegaDepth model, since it performed better in this task.

Results. As shown in Tab. 6, SiLK achieves new SOTA across all metrics (except rotation accuracy at 45°). In particular, SiLK achieves very high accuracy at small thresholds (5° angular, 5cm translation and 1cm for Chamfer), validating SiLK's pixel-level precision. Comparing with DGR, 3D MV Reg and LoFTR that use groundtruth camera poses during training, SiLK significantly outperforms, indicating that groundtruth 3D supervision is not necessary to train good keypoint features. We did not include the Chamfer results for LoFTR as the provided positions do not match the required resolution for correct Chamfer evaluation. SiLK also achieves superior performance vs previous SOTA URR. We note that URR requires two different frames sampled from the same scene during training, and is supervised by an advanced differentiable cross-view rendering process. SiLK, on the other hand, only requires a single 2D image and is trained with a simple point matching loss. Finally, we observe that SuperPoint performs competitively when evaluated in this dense fashion; this is an important difference from the results reported in URR using sparse features.

4.5. What makes good keypoint detectors?

Leveraging SiLK's flexibility (Sec. 3), we comprehensively ablate a large pool of design choices such as model architecture and image resolution. Surprisingly, we found that reducing architecture size, compute cost, and training input size only mildly impact model performance on homography estimation, camera pose estimation and point cloud registration. This benefits many important applications, such as on-device inference.

Here we discuss the key findings. We use the metrics Repeatability@1 (R), Homography Estimation Accuracy@1 (HA), Mean Matching Accuracy@1 (MMA) for HPatches, and Rotation Accuracy@5° (RA), Translation Accuracy@5cm (TA) and Chamfer@1 (C) for ScanNet, all at lowest error thresholds. Additional results and analysis are included in supplementary.

4.5.1 Agnostic to backbone

Existing methods use various backbones (as shown in Tab. 1); the effects on keypoint models are not well understood. We consider FPN from LoFTR and UNet from DISK; both are modern compared to SiLK's VGGnp [45] backbone. We find no empirical performance gain despite far greater parameter counts (Tab. 7). This questions the need for high-capacity models for these keypoint problems.

Next we reduce the complexity of the original SuperPoint backbone VGG-4. **Max-pooling and up-sampling layers are removed**. Our VGGnp-4 contains four convolution blocks, each with two convolution layers followed by ReLU. We discard convolution blocks from VGGnp-4 to obtain VGGnp-3, VGGnp-2 and VGGnp-1. On top of VGGnp-1, we reduce channel and descriptor sizes to 64 and 32 respectively and obtain an ultra-lightweight model, VGGnp-μ. Results on repeatability, homography estimation, camera pose estimation, and point cloud registra-

	Rotation					Translation					Chamfer				
	Accuracy ↑			Error ↓		Accuracy ↑			Error ↓		Accuracy ↑			Error ↓	
	5°	10°	45°	Mean	Med.	5	10	25	Mean	Med.	1	5	10	Mean	Med.
Sparse Features + RANSAC															
SIFT [31]	55.2	75.7	89.2	18.6	4.3	17.7	44.5	79.8	26.5	11.2	38.1	70.6	78.3	42.6	1.7
SuperPoint [16]	65.5	86.9	96.6	8.9	3.6	21.2	51.7	88.0	16.1	9.7	45.7	81.1	88.2	19.2	1.2
FCGF [10]	70.2	87.7	96.2	9.5	3.3	27.5	58.3	82.9	23.6	8.3	52.0	78.0	83.7	24.4	0.9
Pose/Geometry Supervised															
DGR [9]	81.1	89.3	94.8	9.4	1.8	54.5	76.2	88.7	18.4	4.5	70.5	85.5	89.0	13.7	0.4
3D MV Reg [19]	87.7	93.2	97.0	6.0	1.2	69.0	83.1	91.8	11.7	2.9	78.9	89.2	91.8	10.2	0.2
LoFTR(MegaDepth) [47]	91.7	96.8	99.4	2.8	1.2	65.9	85.2	97.1	6.0	3.3	-	-	-	-	-
Dense feature matching															
SuperPoint [16]	93.0	98.4	**99.8**	2.5	1.6	56.8	84.7	98.2	6.5	4.3	77.3	96.1	98.4	4.5	0.3
URR [18]	92.7	95.8	98.5	3.4	**0.8**	77.2	89.6	96.1	7.3	2.3	86.0	94.6	96.1	5.9	**0.1**
SiLK (VGGnp-4)	**98.1**	**99.0**	99.6	**1.7**	**0.8**	**82.9**	**94.8**	**99.0**	**4.1**	**2.1**	**92.8**	**98.3**	**99.1**	**4.3**	**0.1**

Table 6: **SiLK achieves state-of-the-art on camera pose estimation and point cloud registration on ScanNet.**

	HPatches			ScanNet			Model		
	R	HAc	MMA	RA	TA	C	Param	FPS	GFLOP
VGGnp-4	0.62	**0.62**	**0.59**	**98.1**	82.9	92.8	942k	12.2	370
VGGnp-3	0.63	0.61	0.58	98.0	**84.2**	**93.5**	868k	12.5	330
VGGnp-2	0.63	0.57	0.55	97.3	83.5	92.4	757k	14.3	268
VGGnp-1	0.63	0.57	0.44	94.6	82.1	90.0	461k	18.9	90
VGGnp-μ	**0.64**	0.56	0.40	93.5	81.1	89.0	**76k**	**36.5**	**23**
FPN [47]	0.58	0.55	0.52	-	-	-	6.6M	17.5	298
UNet [50]	0.60	0.41	0.58	-	-	-	1.3M	25.9	198

Table 7: **SiLK is backbone agnostic.** Backbones from existing methods are trained and evaluated. Low-capacity model perform well on Repeatability, Homography Estimation Accuracy, Camera Pose Estimation and Point Cloud Registration, but drops on Mean Matching Accuracy. FPS and GFLOPs measured on 480×640 images with NVIDIA Quadro GP100 GPU.

tion only drop mildly as we shrink the model. In particular, SiLK (VGGnp-μ in Tab. 7) achieves very competitive performance vs SOTA (Tab. 2&Tab. 6). On the other hand, keypoint matching (MMA) scores drop signficantly. We suggest two possible reasons: first, pointwise matching may benefit from the larger receptive field of deeper models. Second, homography estimation aggregates numerous pointwise measurements; homographies will improve if the noises cancel out, or if the worst estimates are removed by RANSAC.

4.5.2 Fast training on tiny images

By default, SiLK uses 146x146 descriptor feature map resolution during training. In areas like object detection or image recognition, higher resolution typically leads to stronger results. In SiLK, higher resolution provides more points. This benefits the contrastive loss with more negatives, but also increases training time and GPU memory usage.

Surprisingly, performance changes very little when varying resolution during training (Tab. 8), especially on ScanNet. Tiny feature maps (82x82) remain competitive on both HPatches and ScanNet, and trains in 1.7 hours on two GPUs. This enables applications like test-time finetuning, on-device finetuning, and rapid experimental iteration.

		HPatches			ScanNet		
Size	Time	R	HAc	MMA	RA	TA	C
82^2	**1.7h**	0.60	0.58	0.56	98.1	**83.5**	92.5
114^2	2.7h	0.62	**0.62**	0.59	98.1	82.9	92.8
146^2	5h	**0.63**	0.59	0.59	**98.2**	83.3	**92.9**
178^2	9.5h	**0.63**	**0.62**	0.59	98.1	**83.5**	**92.9**
210^2	18h	**0.63**	0.61	**0.60**	98.1	83.4	92.8

Table 8: **Decreasing training image size has minimal impact.** This suggest SiLK can be trained under 3h, with little performance drop.

4.5.3 Robustness to training data

Existing methods use various training sets (Tab. 1); empirically we observe cases of poor generalization across datasets. For example, LoFTR[47], trained on indoor ScanNet data, drops significantly vs LoFTR trained on outdoor MegaDepth data (Tab. 3) and vice-versa (Tab. 5). This overfitting may be exacerbated by the high-capacity machinery used by these methods, e.g. LoFTR's Transformer contexualizer. We measure SiLK's robustness on training data choices, by using different images from COCO[29], ImageNet[15], MegaDepth[28] and ScanNet[12]. We also combine them to formulate a diversified set of training data.

SiLK is quite robust to change in training set, with the exception of ScanNet (Tab. 9). We hypothesize this is due to the significant amount of uniform surfaces (e.g. walls, doors) present in ScanNet. These featureless areas contain few keypoints to learn from. SiLK's drop agrees directionally with LoFTR's drop observed in Tab. 3, but the magnitude is much smaller. This may be because SiLK's VGGnp backbone has much lower capacity than LoFTR (FPN+Transformer), and hence is less susceptible to over-

	HPatches			ScanNet		
	R	HAc	MMA	RA	TA	C
COCO	0.62	**0.62**	**0.59**	**98.1**	82.9	92.8
ImageNet	0.63	0.6	**0.59**	**98.1**	**83.5**	**93.0**
MegaDep.	0.62	0.61	0.57	97.9	**83.5**	92.9
ScanNet	0.60	0.55	0.48	97.6	82.8	92.7
C+I+M+S	0.61	0.59	0.54	97.7	83.0	92.6
C+I+M	**0.64**	0.6	**0.59**	98.0	82.9	92.8

Table 9: **SiLK is robust to different training sets.** A noticeable drop is observed only when training on ScanNet.

fitting. Finally, we remark that SiLK trained with COCO is used for comparisons in Sec. 4.2, Sec. 4.3 and Sec. 4.4 for HPatches, IMC and ScanNet, whereas LoFTR requires different training data (MegaDepth or ScanNet) to achieve strong performance.

5. Conclusion

This paper presents SiLK, a simple and flexible framework for keypoint detection and descriptors. SiLK is designed from the principles of distinctiveness and invariance, and achieves or advances SOTA on key low-level tasks for 3D visual perception. SiLK's simplicity questions the need for complex machinery for good keypoint detection in low-level applications. In addition, extensive ablations reveal SiLK's robustness to backbone, training data and training input size. These findings lead to a tiny version of SiLK that is lightweight, accurate, and trains quickly. We view this "tiny and learned" regime as very promising for applications where runtime and/or power consumption is critical. We hope SiLK can draw attention to the field and facilitate stronger solutions.

References

[1] Ebtsam Adel, Mohammed Elmogy, and Hazem Elbakry. Image stitching based on feature extraction techniques: a survey. International Journal of Computer Applications, 99(6):1–8, 2014. 1

[2] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In 2012 IEEE conference on computer vision and pattern recognition, pages 510–517. Ieee, 2012. 2

[3] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5173–5182, 2017. 5

[4] Daniel Barath, Jiri Matas, and Jana Noskova. MAGSAC: marginalizing sample consensus. In Conference on Computer Vision and Pattern Recognition, 2019. 5

[5] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). Computer vision and image understanding, 110(3):346–359, 2008. 2

[6] Eric Brachmann and Carsten Rother. Neural-guided ransac: Learning where to sample model hypotheses. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 4321–4330, 2019. 6

[7] Matthew A. Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. International Journal of Computer Vision, 74:59–73, 2006. 1

[8] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In European conference on computer vision, pages 778–792. Springer, 2010. 2

[9] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In CVPR, 2020. 7, 8

[10] Christopher Choy, Jaesik Park, and Vladlen Koltun. Fully convolutional geometric features. In 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pages 8957–8965, 2019. 7, 8

[11] 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

[12] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5828–5839, 2017. 6, 8

[13] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), volume 1, pages 886–893. Ieee, 2005. 2

[14] Andrew J. Davison, Ian D. Reid, Nicholas Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29:1052–1067, 2007. 1

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition, pages 248–255. Ieee, 2009. 8

[16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pages 224–236, 2018. 2, 3, 4, 5, 7, 8

[17] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. arXiv preprint arXiv:1905.03561, 2019. 2, 3, 5, 7, 13

[18] Mohamed El Banani, Luya Gao, and Justin Johnson. Unsupervisedr&r: Unsupervised point cloud registration via differentiable rendering. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 7129–7139, 2021. 6, 7, 8

[19] Z. Gojcic, C. Zhou, J. D. Wegner, L. J. Guibas, and T. Birdal. Learning multiview 3d point cloud registration. In 2020 IEEE/CVF Conference on Computer Vision and

Pattern Recognition (CVPR), pages 1756–1766, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society. 6, 7, 8

[20] Chris Harris, Mike Stephens, et al. A combined corner and edge detector. In Alvey vision conference, volume 15, pages 10–5244. Manchester, UK, 1988. 2

[21] Wilfried Hartmann, Michal Havlena, and Konrad Schindler. Predicting matchability. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 9–16, 2014. 2

[22] 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. International Journal of Computer Vision, 2020. 5, 6

[23] W. Kabsch. A solution for the best rotation to relate two sets of vectors. Acta Crystallographica Section A, 32(5):922–923, Sep 1976. 7

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2015. 5

[25] Benjamin Lefaudeux, Francisco Massa, Diana Liskovich, Wenhan Xiong, Vittorio Caggiano, Sean Naren, Min Xu, Jieru Hu, Marta Tintore, Susan Zhang, Patrick Labatut, and Daniel Haziza. xformers: A modular and hackable transformer modelling library. https://github.com/facebookresearch/xformers, 2022. 4

[26] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In 2011 International conference on computer vision, pages 2548–2555. Ieee, 2011. 2

[27] Xinghui Li, Kai Han, Shuda Li, and Victor Prisacariu. Dual-resolution correspondence networks. In Conference on Neural Information Processing Systems (NeurIPS), 2020. 6, 7

[28] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2041–2050, 2018. 8

[29] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer, 2014. 5, 8

[30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, Computer Vision – ECCV 2016, pages 21–37, Cham, 2016. Springer International Publishing. 1

[31] David G Lowe. Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2):91–110, 2004. 2, 5, 8, 13

[32] Jiayi Ma, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. Image matching from handcrafted to deep features: A survey. International Journal of Computer Vision, 129(1):23–79, 2021. 1

[33] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(10):1615–1630, 2005. 4

[34] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. IEEE Transactions on Robotics, 31:1147–1163, 2015. 1, 2, 5

[35] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018. 2

[36] Markus N Rabe and Charles Staats. Self-attention does not need $o(n^2)$ memory. arXiv preprint arXiv:2112.05682, 2021. 4

[37] Umer Rafi, Andreas Doering, Bastian Leibe, and Juergen Gall. Self-supervised keypoint correspondences for multi-person pose estimation and tracking in videos. In Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX, page 36–52, Berlin, Heidelberg, 2020. Springer-Verlag. 1, 5

[38] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotný. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pages 10881–10891, 2021. 5

[39] Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2d2: repeatable and reliable detector and descriptor. arXiv preprint arXiv:1906.06195, 2019. 2, 3, 5, 13

[40] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In European conference on computer vision, pages 430–443. Springer, 2006. 2

[41] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In 2011 International conference on computer vision, pages 2564–2571. Ieee, 2011. 2

[42] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 4938–4947, 2020. 2, 4, 5, 6, 7

[43] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. IEEE transactions on neural networks, 20(1):61–80, 2008. 2

[44] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4104–4113, 2016. 1, 2

[45] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2015. 7

[46] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. Pacific Journal of Mathematics, 21(2):343–348, 1967. 2

[47] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. Loftr: Detector-free local feature matching with transformers. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 8922–8931, 2021. 2, 4, 5, 6, 8, 13

[48] Jiexiong Tang, Hanme Kim, Vitor Guizilini, Sudeep Pillai, and Rares Ambrus. Neural outlier rejection

for self-supervised keypoint learning. arXiv preprint arXiv:1912.10615, 2019. 2

[49] Prune Truong, Stefanos Apostolopoulos, Agata Mosinska, Samuel Stucky, Carlos Ciller, and Sandro De Zanet. Glampoints: Greedily learned accurate match points. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 10732–10741, 2019. 2, 4

[50] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. Advances in Neural Information Processing Systems, 33:14254–14265, 2020. 2, 3, 4, 5, 8, 13

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017. 2

[52] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal V. Fua. Learning to find good correspondences. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2666–2674, 2017. 6, 7

[53] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning two-view correspondences and geometry using order-aware network. International Conference on Computer Vision (ICCV), 2019. 6, 7

Appendix

A. Maths

In this section, we provide a short mathematical description of our method (probabilistic modeling and losses).

A.1. Definitions

- I, I' is the pair of images to match.
- q_i, $q'_j \in \mathbb{R}$ are the ith and jth keypoint logits from I and I' respectively (M in total).
- d_i, $d'_j \in \mathbb{R}^{128}$ are the ith and jth descriptors from I and I' respectively (M in total).
- $s_{ij} \in [-1, +1]$ is the similarity score between keypoint i from I and keypoint j from I'. s is therefore a $M \times M$ matrix.
- c_i, c'_j are the ith and jth correspondence indices from I and I' respectively (N in total, with $N \leq M$). Such that c_i and c'_i represent the correspondence between descriptor d_{c_i} and $d'_{c'_i}$ with similarity $s_{c_i c'_i}$.

A.2. Matching Probabilities

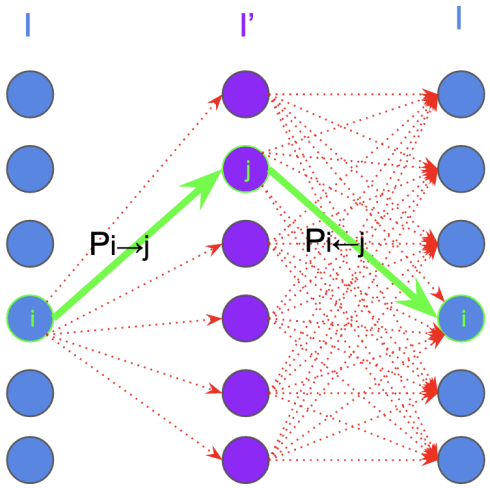


Figure 4: Visualization of the cycle-consistent probabilistic path. The probability $P_{i \leftrightarrow j}$ is the probability of following the green path, over the set of all possible red paths; from image I to I' and back.

The matching probabilities are modeled by a double-softmax, enforcing the cycle-consistency property (cf. Fig. 4).

$$P_{i \leftrightarrow j} = P_{i \rightarrow j} P_{i \leftarrow j}$$

- $P_{i \rightarrow j} = \frac{e^{\frac{s_{ij}}{\tau}}}{\sum_k e^{\frac{s_{ik}}{\tau}}}$ is the directional probability of matching d_i to d'_j
- $P_{i \leftarrow j} = \frac{e^{\frac{s_{ij}}{\tau}}}{\sum_k e^{\frac{s_{kj}}{\tau}}}$ is the directional probability of matching d'_j to d_i

where τ is the temperature, and s the pairwise similarity matrix; obtained using a standard cosine similarity function.

$$s_{ij} = \text{cosim}(d_i, d'_j) = \frac{\langle d_i, d'_j \rangle}{\sqrt{\langle d_i, d_i \rangle \langle d'_j, d'_j \rangle}}$$

where $\langle ., . \rangle$ is the dot product.

A.3. Keypoint Probabilities

All keypoints probabilities are obtained using a simple sigmoid.

$$\sigma(q_i) = \frac{1}{1 + e^{-q_i}}$$

A.4. Matching Loss

The matching loss of a single image pair is the negative log likelihood loss, summed over the entire set of correspondences.

$$\begin{aligned} \mathcal{L}_{desc} &= \text{NLL}(s, c, c') \\ &= -\frac{1}{N} \sum_{i=0}^{N-1} \log P_{c_i \leftrightarrow c'_i} \\ &= -\frac{1}{N} \sum_{i=0}^{N-1} \left[\log P_{c_i \rightarrow c'_i} + \log P_{c'_i \leftarrow c_i} \right] \end{aligned}$$

A.5. Keypoint Loss

Once the matching success of descriptors as been measured (and stored in variable $y \in \{0, 1\}^N$, cf. Appendix A.6), we can learn the keypoint probabilities using a standard binary cross-entropy loss; using the keypoint logits extracted from both I and I'.

$$\mathcal{L}_{key} = \text{BCE}(q, y, c) + \text{BCE}(q', y, c')$$

where

$$\text{BCE}(q, y, c) = -\frac{1}{N} \sum_{i=0}^{N-1} \left[y_i \log \sigma(+q_{c_i}) + (1 - y_i) \log \sigma(-q_{c_i}) \right]$$

A.6. Matching Success

The matching success is the process of measuring whether or not matching currently learned descriptors (using a simple mutual nearest neighbor) would produce correct matches. It can be expressed mathematically as verifying whether or not the similarity of a ground truth correspondence is the row and column maximum in s.

$$y_i = \mathbf{1}\big[s_{c_i c'_i} \geq \max_k\{s_{c_i k}\}\big]\mathbf{1}\big[s_{c_i c'_i} \geq \max_k\{s_{k c'_i}\}\big]$$

where $\mathbf{1}\big[.\big]$ is the indicator function.

B. Additional Experiments

In this section, we present additional experiments as evidence of the robustness of SiLK under varying, but realistic, conditions. We also hope this comprehensive set of data points can be used by the community to tune their own version of SiLK to a specific use case or task. For example, if a task is sensitive to false positive matching, one might consider using the ratio-test filtering, as indicated in Tab. 12.

SiLK's robustness is tested against three important types of variations, as we aim to answer the following.

1) Are SiLK's results robust across different image resolutions ? (cf. Tab. 10 and Fig. 5)

2) Can we improve SiLK by simply selecting more keypoints ? Or do we reach saturation for a certain value of k ? (cf. Tab. 11 and Fig. 6)

3) Do existing false-positive removal techniques work on SiLK? And how is performance affected by it ?(cf. Tab. 12).

Additionally, we empirically demonstrate the importance of **not** using zero-padding when learning keypoints (cf. Tab. 13). This is an often under-emphasized point we shed light on here.

B.1. Downsizing images is better than increasing ϵ

All of the HPatches metrics reported in this paper use an ϵ-distance error threshold to determine whether or not a pixel position is considered close-enough to its ground truth. A low ϵ means the metrics are reported in a highly accurate regime, while a high value of ϵ allows for some local mistakes to occur. However, ϵ is an absolute pixel distance, which means that metrics might vary in non-trivial ways as the input resolution changes during inference.

Existing methods tend to report metrics using high values of ϵ. For example, D2-Net [17], R2D2 [39] and DISK [50] all report MMA with ϵ-thresholds up to 10. We argue this is unnecessary. Tasks that do not require accurate keypoints (i.e. high values of ϵ) might want to reconsider running their keypoint model on lower resolution images (to reduce computational cost). As an input image is downscaled by a factor γ, ϵ should also be downscaled by the same factor in order to keep its relative size constant. So in theory, a metric reported on resolution α with error-threshold ϵ should be roughly equivalent to the same metric reported on resolution $\frac{\alpha}{\gamma}$ with error threshold $\frac{\epsilon}{\gamma}$.

In Tab. 10, we show this initial intuition is not correct. When looking at resolutions of 720 and 240 (i.e. downscaling of γ=3), existing methods all underperform their expected theoretical values. SiLK is the only model that consistently gain from running at lower resolutions. This is likely caused by SiLK's ability to obtain high number of keypoints on low resolution images while other methods are limited by their sparsity constraints (i.e. cell-based keypoint detection and NMS).

Additionally, we show in Fig. 5 that SiLK's performance against sparse keypoint methods is robust across a wide range of resolutions; with the exception of SIFT on large image resolution, using Homography Estimation Accuracy metric. One can also observe large performance gain (except on MMA) versus LoFTR [47](dense keypoint method) in the low resolution regime.

B.2. Increasing top-k improves SiLK, but saturates early

Increasing top-k has shown multiple times (cf. Tab. 10) to improve overall results. In this experiment, we simply vary the parameter k in order to monitor SiLK's performance.

As can be observed in Fig. 6 and Tab. 11, results start to saturate after $k = 10,000$; on Homography Estimation and MMA. Repeatability continuously increases as we increase k, but that is simply a consequences of getting more keypoints (i.e. the keypoint overlaps become more likely).

B.3. Improving MMA using ratio-test or double-softmax filtering

All previously reported results have been computed using MNN matching on unprocessed cosine distances. There are, however, known distance post-processing techniques used to reduce false positive matching. The ratio-test [31] is one of those. The distance of the best match is divided by the distance of the second best match. A low value indicates a large difference between the two best distances, which indicates a measure of relative distinctiveness. Therefore, filtering out matches with high ratio values do tend to reduce matching errors caused by repeated similar keypoints (e.g. window corners of a building).

More recently[47], a similar idea has emerged from the probabilistic formulation of the matching problem: Filtering out low-probability matches seems like a natural way to reduce false positive matches.

In Fig. 7 and Tab. 12, we show that using either ratio-test or double-softmax filtering can help SiLK trade Homography Estimation for MMA.

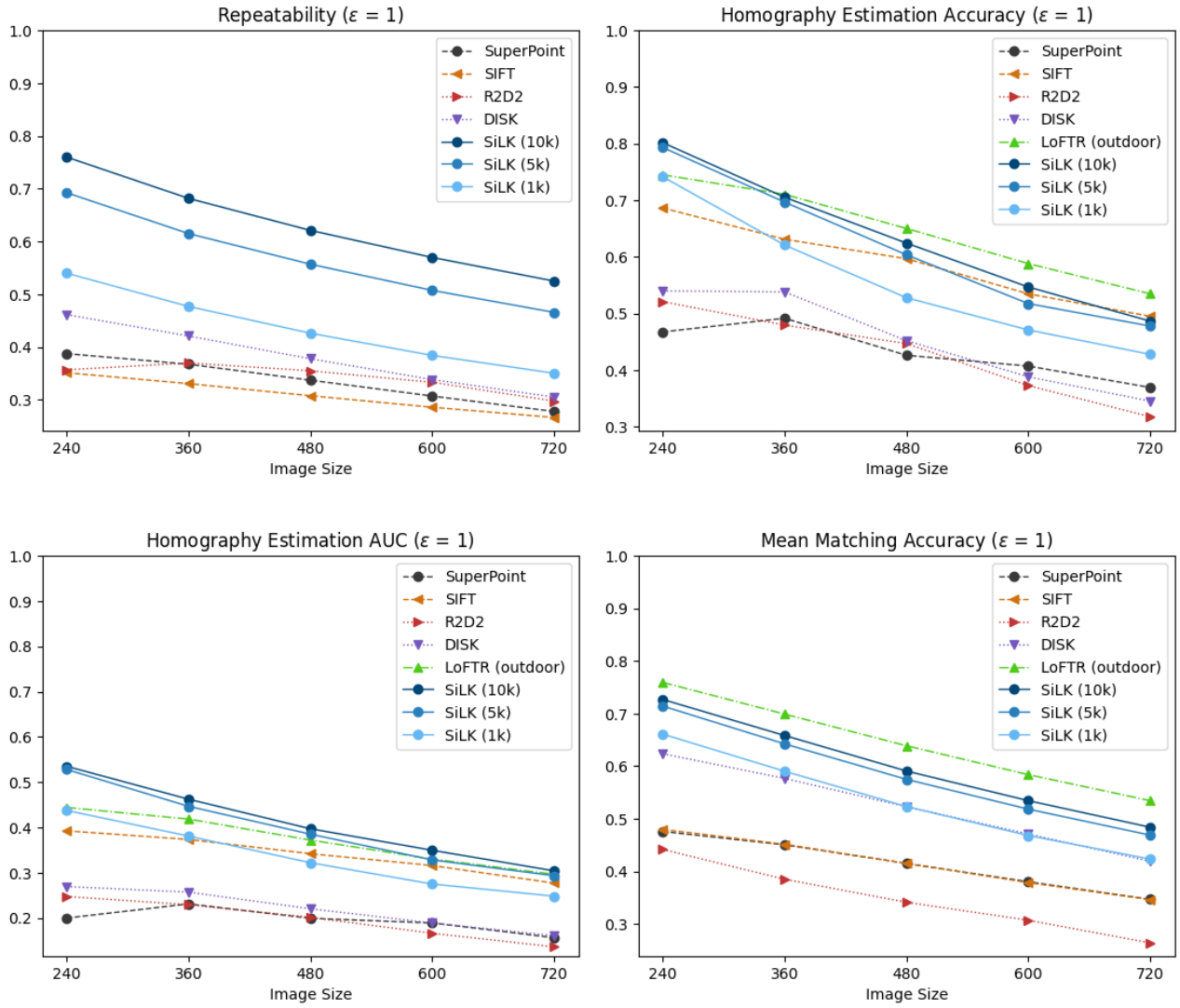


Figure 5: **SiLK's performance is robust across different input scales.** All sparse methods are outperformed by SiLK on all metrics and all scales (except SIFT on Homography Estimation Accuracy and large image resolution). Notice how R2D2, DISK and SuperPoint rankings differ across resolutions.

B.4. Use NO padding to learn keypoints.

Zero padding is commonly used in various models. It is the process of adding a 0-filled border to an image or dense feature map. A common example is when using 3x3 convolutions, adding a padding of 1 will ensure the spatial shape of the input is preserved, otherwise it would be reduced.

The use or lack of padding is rarely mentioned by existing keypoint methods. However, we find that most implementations do in fact avoid the use of zero padding. Other implementations might use it, but then compensate by removing an arbitrarily-sized border from the dense outputs.

The reason for not using padding in SiLK's case is because it creates easily detectable corners and edges on the image borders, therefore causing overfitting during training.

To show the importance of **not** using padding when learning keypoints, we provide two tables (cf. Tab. 13) as evidence of the adverse effects of using it.

C. Implementation details

C.1. Data augmentation

Here we detail the data augmentation used by SiLK during training, provided by Albumentation library (Fig. 8):

HPatches											
	Size	Repeatability		Hom. Est. Acc.		Hom. Est. AUC		MMA		# of keypoints	
		$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	pre-match	post-match
SuperPoint	240	0.39×	0.64	0.47×	0.83	0.20×	0.54	0.48×	0.76	305	187
	360	0.37	0.63	0.49	0.83	0.23	0.55	0.45	0.74	569	341
	480	0.34	0.61	0.43	0.80	0.20	0.51	0.41	0.72	847	499
	600	0.31	0.59	0.41	0.79	0.19	0.49	0.38	0.69	1141	655
	720	0.28	0.56✓	0.37	0.74✓	0.16	0.45✓	0.35	0.66✓	1460	813
SIFT	240	0.35×	0.54	0.69×	0.89	0.39×	0.68	0.48×	0.57	677	298
	360	0.33	0.53	0.63	0.85	0.37	0.65	0.45	0.56	1331	570
	480	0.31	0.52	0.60	0.84	0.34	0.61	0.41	0.55	2189	910
	600	0.29	0.51	0.53	0.80	0.32	0.58	0.38	0.53	3212	1297
	720	0.27	0.50✓	0.49	0.75✓	0.28	0.53✓	0.35	0.50✓	4273	1681
R2D2	240	0.36×	0.70	0.52×	0.81	0.25×	0.56	0.44×	0.79	1037	351
	360	0.37	0.73	0.48	0.82	0.23	0.54	0.38	0.77	3193	1042
	480	0.36	0.72	0.45	0.79	0.20	0.50	0.34	0.75	6088	1967
	600	0.33	0.71	0.37	0.76	0.17	0.46	0.31	0.72	9517	2994
	720	0.30	0.68✓	0.32	0.69✓	0.14	0.42✓	0.26	0.67✓	12036	3698
DISK	240	0.46×	0.72	0.54×	0.86	0.27×	0.60	0.62×	0.87	841	484
	360	0.42	0.71	0.54	0.85	0.26	0.58	0.58	0.86	1847	1030
	480	0.38	0.69	0.45	0.80	0.22	0.52	0.52	0.84	3349	1794
	600	0.34	0.67	0.39	0.75	0.19	0.47	0.47	0.81	5152	2647
	720	0.31	0.65✓	0.34	0.71✓	0.16	0.43✓	0.42	0.77✓	7417	3732
LoFTR (outdoor)	240	-	-	0.74×	0.90	0.44×	0.72	0.76×	0.93	1280	662
	360	-	-	0.71	0.90	0.42	0.70	0.70	0.92	2878	1533
	480	-	-	0.65	0.87	0.37	0.65	0.64	0.91	5109	2719
	600	-	-	0.59	0.83	0.33	0.61	0.58	0.89	7987	4166
	720	-	-	0.53	0.79✓	0.30	0.57✓	0.53	0.87✓	11490	5804
SiLK (top-10k)	240	0.76✓	0.90	0.80✓	0.93	0.54✓	0.78	0.73✓	0.79	10000	4816
	360	0.68	0.85	0.71	0.91	0.46	0.72	0.66	0.75	10000	4515
	480	0.62	0.81	0.62	0.87	0.40	0.66	0.59	0.71	10000	4283
	600	0.57	0.77	0.55	0.81	0.35	0.59	0.53	0.67	10000	4092
	720	0.53	0.73×	0.49	0.76×	0.30	0.53×	0.48	0.63×	10000	3945
SiLK (top-5k)	240	0.69✓	0.86	0.79✓	0.93	0.53✓	0.77	0.71✓	0.77	5000	2331
	360	0.62	0.80	0.70	0.90	0.45	0.70	0.64	0.73	5000	2181
	480	0.56	0.76	0.60	0.85	0.39	0.64	0.57	0.69	5000	2074
	600	0.51	0.71	0.52	0.80	0.33	0.57	0.52	0.65	5000	1983
	720	0.47	0.67×	0.48	0.74×	0.29	0.52×	0.47	0.61×	5000	1919
SiLK (top-1k)	240	0.54✓	0.73	0.74✓	0.91	0.44×	0.72	0.66✓	0.71	1000	429
	360	0.48	0.66	0.62	0.86	0.38	0.65	0.59	0.67	1000	408
	480	0.43	0.61	0.53	0.81	0.32	0.58	0.52	0.63	1000	389
	600	0.38	0.57	0.47	0.75	0.28	0.52	0.47	0.59	1000	376
	720	0.35	0.53×	0.43	0.69×	0.25	0.48✓	0.42	0.55×	1000	366

Table 10: **SiLK is the only model that benefits from running at lower resolutions.** On each metric and method, we compare the lowest (resolution=240, ϵ=1) pair, to the highest (resolution=720, ϵ=3) pair. Since the resolution / ϵ ratio is the same for both pairs, the measured level of accuracy is equivalent.

HPatches										
	Repeatability		Hom. Est. Acc.		Hom. Est. AUC		MMA		# of keypoints	
k	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	pre-match	post-match
1.0k	0.43	0.61	0.53	0.81	0.32	0.58	0.52	0.63	1000	389
2.5k	0.50	0.70	0.58	0.83	0.37	0.62	0.55	0.67	2500	1006
5.0k	0.56	0.76	0.60	0.85	0.39	0.64	0.57	0.69	5000	2074
7.5k	0.59	0.79	0.60	0.85	0.39	0.64	0.58	0.70	7500	3169
10.0k	0.62	0.81	0.62	0.87	0.40	0.66	0.59	0.71	10000	4283
12.5k	0.64	0.82	0.63	0.87	0.40	0.66	0.59	0.72	12500	5408
15.0k	0.66	0.83	0.62	0.86	0.40	0.66	0.60	0.72	15000	6541
17.5k	0.67	0.84	0.62	0.86	0.41	0.66	0.60	0.73	17500	7677
20.0k	0.69	0.85	0.63	0.86	0.41	0.66	0.60	0.73	20000	8821
22.5k	0.70	0.85	0.62	0.87	0.41	0.66	0.60	0.73	22500	9963
25.0k	0.71	0.86	0.64	0.87	0.42	0.67	0.60	0.73	25000	11106

Table 11: Numerical results used in Fig. 6.

HPatches									
		Hom. Est. Acc.		Hom. Est. AUC		MMA		# of keypoints	
	Threshold	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 1$	$\epsilon = 3$	pre-match	post-match
ratio-test	1.00	0.62	0.87	0.40	0.66	0.59	0.71	10000	4283
	0.95	0.61	0.87	0.40	0.65	0.62	0.75	10000	3771
	0.90	0.61	0.86	0.40	0.65	0.66	0.79	10000	3314
	0.85	0.59	0.84	0.40	0.64	0.69	0.83	10000	2926
	0.80	0.60	0.83	0.40	0.63	0.72	0.86	10000	2599
	0.75	0.58	0.82	0.40	0.62	0.74	0.88	10000	2314
	0.70	0.56	0.82	0.39	0.61	0.76	0.89	10000	2061
	0.65	0.54	0.78	0.39	0.59	0.77	0.90	10000	1830
	0.60	0.53	0.78	0.38	0.58	0.77	0.91	10000	1620
	0.55	0.51	0.75	0.37	0.57	0.77	0.91	10000	1427
	0.50	0.49	0.76	0.37	0.55	0.76	0.90	10000	1248
double-softmax	1.00	0.62	0.86	0.40	0.65	0.52	0.63	10000	5321
	0.95	0.59	0.86	0.39	0.64	0.68	0.83	10000	3650
	0.90	0.59	0.84	0.39	0.63	0.73	0.88	10000	3244
	0.85	0.55	0.83	0.39	0.62	0.75	0.91	10000	2923
	0.80	0.56	0.81	0.39	0.61	0.77	0.92	10000	2616
	0.75	0.56	0.80	0.39	0.61	0.77	0.92	10000	2319
	0.70	0.55	0.79	0.39	0.60	0.78	0.93	10000	2041
	0.65	0.55	0.79	0.38	0.59	0.78	0.93	10000	1782
	0.60	0.53	0.78	0.38	0.58	0.79	0.93	10000	1547
	0.55	0.53	0.76	0.38	0.57	0.79	0.92	10000	1335
	0.50	0.51	0.76	0.38	0.57	0.79	0.92	10000	1141

Table 12: Numerical results used in Fig. 7.

HPatches											
	Repeatability		Hom. Est. Acc.		Hom. Est. AUC		MMA		# of keypoints	
padding	0.59	0.79	0.59	0.84	0.39	0.63	0.57	0.68	10000	4222
no padding	**0.62**	**0.81**	**0.62**	**0.87**	**0.40**	**0.66**	**0.59**	**0.71**	10000	4283

	ScanNet														
	Rotation					Translation					Chamfer				
	Accuracy ↑			Error ↓		Accuracy ↑			Error ↓		Accuracy ↑			Error ↓	
	5°	10°	45°	Mean	Med.	5	10	25	Mean	Med.	1	5	10	Mean	Med.
padding	93.7	97.2	**99.7**	2.1	0.9	75.2	89.8	97.4	5.2	2.5	85.6	95.9	97.8	4.6	**0.1**
no padding	**98.1**	**99.0**	99.6	**1.7**	**0.8**	**82.9**	**94.8**	**99.0**	**4.1**	**2.1**	**92.8**	**98.3**	**99.1**	**4.3**	**0.1**

Table 13: **Avoid using padding to learn good keypoints.**

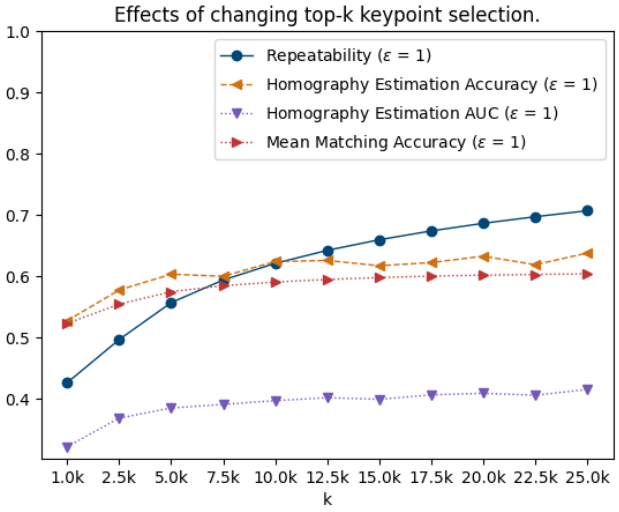


Figure 6: **Increasing top-k keypoint selection gives an initial boost in performance, but tend to get dimishing returns for $\mathbf{k} > 10,000$.**

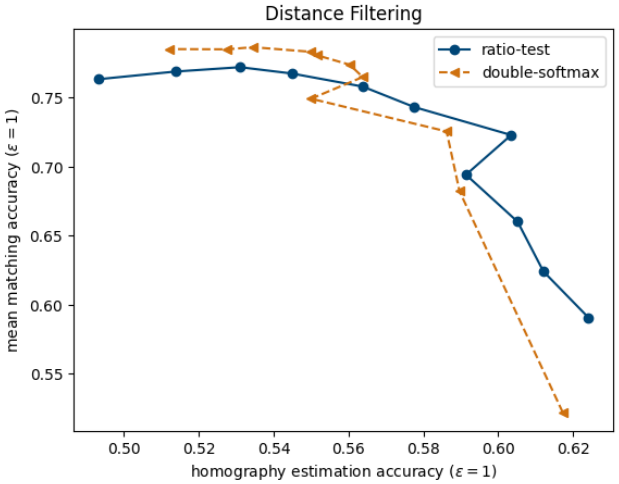


Figure 7: **MMA / Homography trade-off can be controlled with ratio-test and double-softmax filtering.** Different threshold values are tested (between 0.5 and 1) using both methods.

```python
import albumentations as A

silk_augmentation = A.Compose([
    A.RandomGamma(
      p=0.1, gamma_limit=(15, 65)
    ),
    A.HueSaturationValue(
      p=0.1, val_shift_limit=(-100, -40)
    ),
    A.Blur(
      p=0.1, blur_limit=(3, 9)
    ),
    A.MotionBlur(
      p=0.2, blur_limit=(3, 25)
    ),
    A.RandomBrightnessContrast(
      p=0.5,
      brightness_limit=(-0.3, 0.0),
      contrast_limit=(-0.5, 0.3)
    ),
    A.GaussNoise(p=0.5),
], p=0.95)
```

Figure 8: Pseudo-code: Data augmentation for SiLK.