SuperPoint: Self-Supervised Interest Point Detection and Description

Daniel DeTone
Magic Leap
Sunnyvale, CA
ddetone@magicleap.com

Tomasz Malisiewicz
Magic Leap
Sunnyvale, CA
tmalisiewicz@magicleap.com

Andrew Rabinovich
Magic Leap
Sunnyvale, CA
arabinovich@magicleap.com

Abstract

*This paper presents a self-supervised framework for training interest point detectors and descriptors suitable for a large number of multiple-view geometry problems in computer vision. As opposed to patch-based neural networks, our fully-convolutional model operates on full-sized images and jointly computes pixel-level interest point locations and associated descriptors in one forward pass. We introduce Homographic Adaptation, a multi-scale, multi-homography approach for boosting interest point detection repeatability and performing cross-domain adaptation (*e.g.*, synthetic-to-real). Our model, when trained on the MS-COCO generic image dataset using Homographic Adaptation, is able to repeatedly detect a much richer set of interest points than the initial pre-adapted deep model and any other traditional corner detector. The final system gives rise to state-of-the-art homography estimation results on HPatches when compared to LIFT, SIFT and ORB.*

1. Introduction

The first step in geometric computer vision tasks such as Simultaneous Localization and Mapping (SLAM), Structure-from-Motion (SfM), camera calibration, and image matching is to extract interest points from images. Interest points are 2D locations in an image which are stable and repeatable from different lighting conditions and viewpoints. The subfield of mathematics and computer vision known as Multiple View Geometry [9] consists of theorems and algorithms built on the assumption that interest points can be reliably extracted and matched across images. However, the inputs to most real-world computer vision systems are raw images, not idealized point locations.

Convolutional neural networks have been shown to be superior to hand-engineered representations on almost all tasks requiring images as input. In particular, fully-convolutional neural networks which predict 2D "keypoints" or "landmarks" are well-studied for a variety of tasks such as human pose estimation [31], object detection [14], and room layout estimation [12]. At the heart of these techniques is a large dataset of 2D ground truth locations labeled by human annotators.

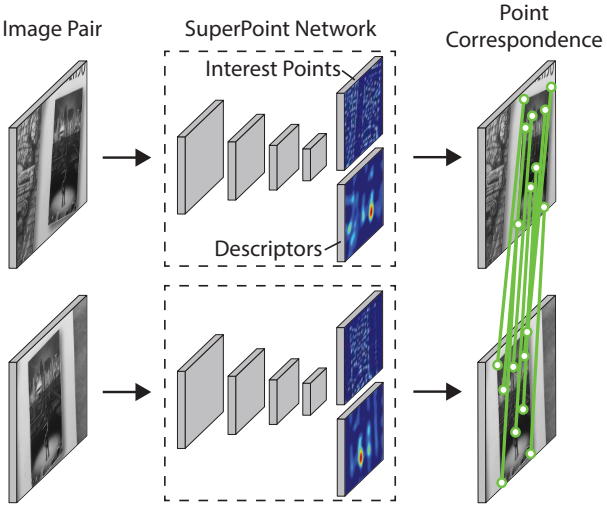

Figure 1. **SuperPoint for Geometric Correspondences.** We present a fully-convolutional neural network that computes SIFT-like 2D interest point locations and descriptors in a single forward pass and runs at 70 FPS on 480×640 images with a Titan X GPU.

It seems natural to similarly formulate interest point detection as a large-scale supervised machine learning problem and train the latest convolutional neural network architecture to detect them. Unfortunately, when compared to semantic tasks such as human-body keypoint estimation, where a network is trained to detect body parts such as the corner of the mouth or left ankle, the notion of interest point detection is semantically ill-defined. Thus training convolution neural networks with strong supervision of interest points is non-trivial.

Instead of using human supervision to define interest points in real images, we present a self-supervised solution using self-training. In our approach, we create a large dataset of pseudo-ground truth interest point locations in real images, supervised by the interest point detector itself, rather than a large-scale human annotation effort.

To generate the pseudo-ground truth interest points, we first train a fully-convolutional neural network on millions

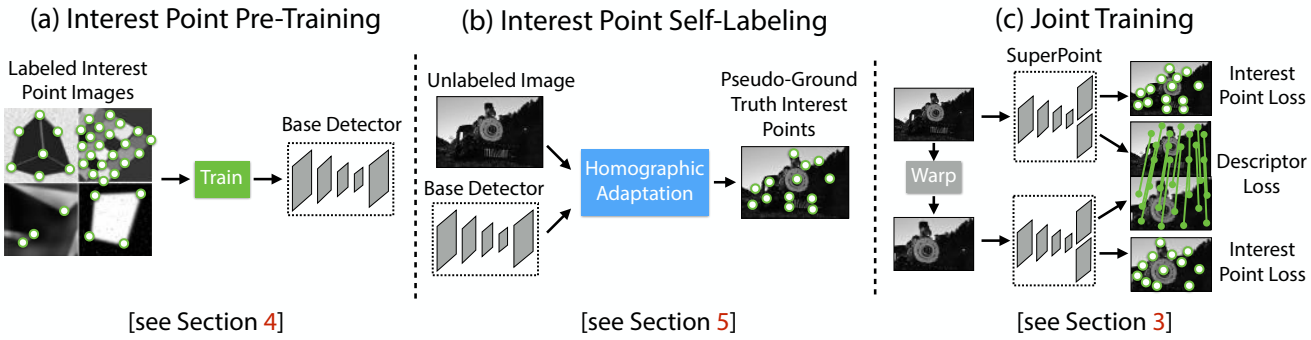

Figure 2. **Self-Supervised Training Overview.** In our self-supervised approach, we (a) pre-train an initial interest point detector on synthetic data and (b) apply a novel Homographic Adaptation procedure to automatically label images from a target, unlabeled domain. The generated labels are used to (c) train a fully-convolutional network that jointly extracts interest points and descriptors from an image.

of examples from a synthetic dataset we created called *Synthetic Shapes* (see Figure 2a). The synthetic dataset consists of simple geometric shapes with no ambiguity in the interest point locations. We call the resulting trained detector *MagicPoint*—it significantly outperforms traditional interest point detectors on the synthetic dataset (see Section 4). MagicPoint performs surprising well on real images despite domain adaptation difficulties [7]. However, when compared to classical interest point detectors on a diverse set of image textures and patterns, MagicPoint misses many potential interest point locations. To bridge this gap in performance on real images, we developed a multi-scale, multi-transform technique — *Homographic Adaptation*.

Homographic Adaptation is designed to enable self-supervised training of interest point detectors. It warps the input image multiple times to help an interest point detector see the scene from many different viewpoints and scales (see Section 5). We use Homographic Adaptation in conjunction with the MagicPoint detector to boost the performance of the detector and generate the pseudo-ground truth interest points (see Figure 2b). The resulting detections are more repeatable and fire on a larger set of stimuli; thus we named the resulting detector *SuperPoint*.

The most common step after detecting robust and repeatable interest points is to attach a fixed dimensional descriptor vector to each point for higher level semantic tasks, *e.g.*, image matching. Thus we lastly combine SuperPoint with a descriptor subnetwork (see Figure 2c). Since the SuperPoint architecture consists of a deep stack of convolutional layers which extract multi-scale features, it is straightforward to then combine the interest point network with an additional subnetwork that computes interest point descriptors (see Section 3). The resulting system is shown in Figure 1.

2. Related Work

Traditional interest point detectors have been thoroughly evaluated [24, 16]. The FAST corner detector [21] was the first system to cast high-speed corner detection as a machine learning problem, and the Scale-Invariant Feature Transform, or SIFT [15], is still probably the most well-known traditional local feature descriptor in computer vision.

Our SuperPoint architecture is inspired by recent advances in applying deep learning to interest point detection and descriptor learning. At the ability to match image substructures, we are similar to UCN [3] and to a lesser extent DeepDesc [6]; however, both do not perform any interest point detection. On the other end, LIFT [32], a recently introduced convolutional replacement for SIFT stays close to the traditional patch-based detect then describe recipe. The LIFT pipeline contains interest point detection, orientation estimation and descriptor computation, but additionally requires supervision from a classical SfM system. These differences are summarized in Table 1.

	Interest Points?	Descriptors?	Full Image Input?	Single Network?	Real Time?
SuperPoint (ours)	✓	✓	✓	✓	✓
LIFT [32]	✓	✓			
UCN [3]		✓	✓	✓	
TILDE [29]	✓			✓	
DeepDesc [6]		✓		✓	
SIFT	✓	✓			
ORB	✓	✓			✓

Table 1. **Qualitative Comparison to Relevant Methods.** Our SuperPoint method is the only one to compute both interest points and descriptors in a single network in real-time.

On the other extreme of the supervision spectrum, Quad-Networks [23] tackles the interest point detection problem from an unsupervised approach; however, their system is patch-based (inputs are small image patches) and relatively shallow 2-layer network. The TILDE [29] interest point detection system used a principle similar to Homographic Adaptation; however, their approach does not benefit from the power of large fully-convolutional neural networks.

Our approach can also be compared to other self-supervised methods, synthetic-to-real domain-adaptation methods. A similar approach to Homographic Adaptation is by Honari *et al*. [10] under the name "equivariant landmark transform." Also, Geometric Matching Networks [20] and Deep Image Homography Estimation [4] use a similar self-supervision strategy to create training data for estimating global transformations. However, these methods lack interest points and point correspondences, which are typically required for doing higher level computer vision tasks such as SLAM and SfM. Joint pose and depth estimation models also exist [33, 30, 28], but do not use interest points.

3. SuperPoint Architecture

We designed a fully-convolutional neural network architecture called SuperPoint which operates on a full-sized image and produces interest point detections accompanied by fixed length descriptors in a single forward pass (see Figure 3). The model has a single, shared encoder to process and reduce the input image dimensionality. After the encoder, the architecture splits into two decoder "heads", which learn task specific weights – one for interest point detection and the other for interest point description. Most of the network's parameters are shared between the two tasks, which is a departure from traditional systems which first detect interest points, then compute descriptors and lack the ability to share computation and representation across the two tasks.

3.1. Shared Encoder

Our SuperPoint architecture uses a VGG-style [27] encoder to reduce the dimensionality of the image. The encoder consists of convolutional layers, spatial downsampling via pooling and non-linear activation functions. Our encoder uses three max-pooling layers, letting us define $H_c = H/8$ and $W_c = W/8$ for an image sized $H \times W$. We refer to the pixels in the lower dimensional output as "cells," where three 2×2 non-overlapping max pooling operations in the encoder result in 8×8 pixel cells. The encoder maps the input image $I \in \mathbb{R}^{H \times W}$ to an intermediate tensor $\mathcal{B} \in \mathbb{R}^{H_c \times W_c \times F}$ with smaller spatial dimension and greater channel depth (*i.e.*, $H_c < H$, $W_c < W$ and $F > 1$).

3.2. Interest Point Decoder

For interest point detection, each pixel of the output corresponds to a probability of "point-ness" for that pixel in the input. The standard network design for dense prediction involves an encoder-decoder pair, where the spatial resolution is decreased via pooling or strided convolution, and then upsampled back to full resolution via upconvolution operations, such as done in SegNet [1]. Unfortunately, upsampling layers tend to add a high amount of computation and can introduce unwanted checkerboard artifacts [18], thus we designed the interest point detection head with an explicit decoder[1] to reduce the computation of the model.

The interest point detector head computes $\mathcal{X} \in \mathbb{R}^{H_c \times W_c \times 65}$ and outputs a tensor sized $\mathbb{R}^{H \times W}$. The 65 channels correspond to local, non-overlapping 8×8 grid regions of pixels plus an extra "no interest point" dustbin. After a channel-wise softmax, the dustbin dimension is removed and a $\mathbb{R}^{H_c \times W_c \times 64} \Rightarrow \mathbb{R}^{H \times W}$ reshape is performed.

[1]This decoder has no parameters, and is known as "sub-pixel convolution" [26] or "depth to space" in TensorFlow or "pixel shuffle" in PyTorch.

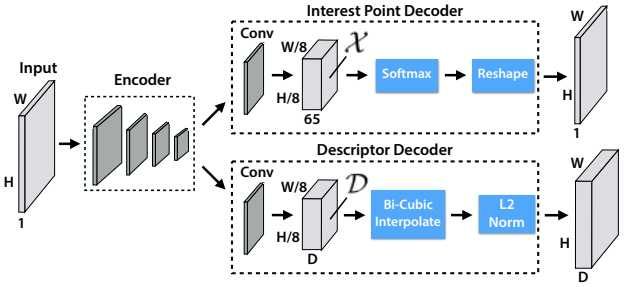

Figure 3. **SuperPoint Decoders**. Both decoders operate on a shared and spatially reduced representation of the input. To keep the model fast and easy to train, both decoders use non-learned upsampling to bring the representation back to $\mathbb{R}^{H \times W}$.

3.3. Descriptor Decoder

The descriptor head computes $\mathcal{D} \in \mathbb{R}^{H_c \times W_c \times D}$ and outputs a tensor sized $\mathbb{R}^{H \times W \times D}$. To output a dense map of L2-normalized fixed length descriptors, we use a model similar to UCN [3] to first output a semi-dense grid of descriptors (*e.g.*, one every 8 pixels). Learning descriptors semi-densely rather than densely reduces training memory and keeps the run-time tractable. The decoder then performs bi-cubic interpolation of the descriptor and then L2-normalizes the activations to be unit length. This fixed, non-learned descriptor decoder is shown in Figure 3.

3.4. Loss Functions

The final loss is the sum of two intermediate losses: one for the interest point detector, $\mathcal{L}_p$, and one for the descriptor, $\mathcal{L}_d$. We use pairs of synthetically warped images which have both (a) pseudo-ground truth interest point locations and (b) the ground truth correspondence from a randomly generated homography $\mathcal{H}$ which relates the two images. This allows us to optimize the two losses simultaneously, given a pair of images, as shown in Figure 2c. We use λ to balance the final loss:

$$\begin{aligned} &\mathcal{L}(\mathcal{X}, \mathcal{X}', \mathcal{D}, \mathcal{D}'; Y, Y', S) = \\ &\qquad \mathcal{L}_p(\mathcal{X}, Y) + \mathcal{L}_p(\mathcal{X}', Y') + \lambda \mathcal{L}_d(\mathcal{D}, \mathcal{D}', S). \end{aligned} \tag{1}$$

The interest point detector loss function $\mathcal{L}_p$ is a fully-convolutional cross-entropy loss over the cells $\mathbf{x}_{hw} \in \mathcal{X}$. We call the set of corresponding ground-truth interest point labels[2] Y and individual entries as y_{hw}. The loss is:

$$\mathcal{L}_p(\mathcal{X}, Y) = \frac{1}{H_c W_c} \sum_{\substack{h=1 \\ w=1}}^{H_c, W_c} l_p(\mathbf{x}_{hw}; y_{hw}), \tag{2}$$

where

$$l_p(\mathbf{x}_{hw}; y) = -\log\left(\frac{\exp(\mathbf{x}_{hwy})}{\sum_{k=1}^{65} \exp(\mathbf{x}_{hwk})}\right). \tag{3}$$

[2]If two ground truth corner positions land in the same bin then we randomly select one ground truth corner location.

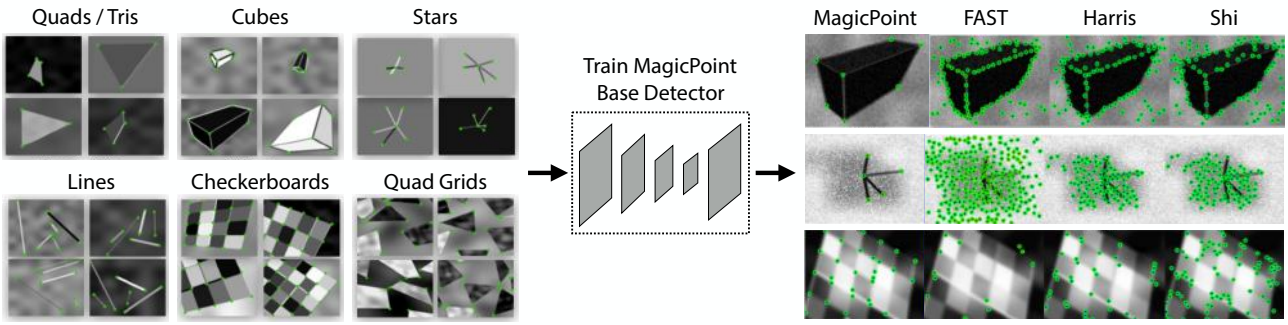

Figure 4. **Synthetic Pre-Training.** We use our Synthetic Shapes dataset consisting of rendered triangles, quadrilaterals, lines, cubes, checkerboards, and stars each with ground truth corner locations. The dataset is used to train the MagicPoint convolutional neural network, which is more robust to noise when compared to classical detectors.

The descriptor loss is applied to all pairs of descriptor cells, $\mathbf{d}_{hw} \in \mathcal{D}$ from the first image and $\mathbf{d}'_{h'w'} \in \mathcal{D}'$ from the second image. The homography-induced correspondence between the (h, w) cell and the (h', w') cell can be written as follows:

$$s_{hwh'w'} = \begin{cases} 1, & \text{if } ||\widehat{\mathcal{H}\mathbf{p}_{hw}} - \mathbf{p}_{h'w'}|| \leq 8 \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

where $\mathbf{p}_{hw}$ denotes the location of the center pixel in the (h, w) cell, and $\widehat{\mathcal{H}\mathbf{p}_{hw}}$ denotes multiplying the cell location $\mathbf{p}_{hw}$ by the homography $\mathcal{H}$ and dividing by the last coordinate, as is usually done when transforming between Euclidean and homogeneous coordinates. We denote the entire set of correspondences for a pair of images with S.

We also add a weighting term λ_d to help balance the fact that there are more negative correspondences than positive ones. We use a hinge loss with positive margin m_p and negative margin m_n. The descriptor loss is defined as:

$$\mathcal{L}_d(\mathcal{D}, \mathcal{D}', S) = \frac{1}{(H_c W_c)^2} \sum_{\substack{h=1 \\ w=1}}^{H_c, W_c} \sum_{\substack{h'=1 \\ w'=1}}^{H_c, W_c} l_d(\mathbf{d}_{hw}, \mathbf{d}'_{h'w'}; s_{hwh'w'}), \quad (5)$$

where

$$l_d(\mathbf{d}, \mathbf{d}'; s) = \lambda_d * s * \max(0, m_p - \mathbf{d}^T \mathbf{d}') + (1 - s) * \max(0, \mathbf{d}^T \mathbf{d}' - m_n). \quad (6)$$

4. Synthetic Pre-Training

In this section, we describe our method for training a base detector (shown in Figure 2a) called MagicPoint which is used in conjunction with Homographic Adaptation to generate pseudo-ground truth interest point labels for unlabeled images in a self-supervised fashion.

4.1. Synthetic Shapes

There is no large database of interest point labeled images that exists today. Thus to bootstrap our deep interest point detector, we first create a large-scale synthetic dataset called *Synthetic Shapes* that consists of simplified 2D geometry via synthetic data rendering of quadrilaterals, triangles, lines and ellipses. Examples of these shapes are shown in Figure 4. In this dataset, we are able to remove label ambiguity by modeling interest points with simple Y-junctions, L-junctions, T-junctions as well as centers of tiny ellipses and end points of line segments.

Once the synthetic images are rendered, we apply homographic warps to each image to augment the number of training examples. The data is generated on-the-fly and no example is seen by the network twice. While the types of interest points represented in Synthetic Shapes represents only a subset of all potential interest points found in the real world, we found it to work reasonably well in practice when used to train an interest point detector.

4.2. MagicPoint

We use the detector pathway of the SuperPoint architecture (ignoring the descriptor head) and train it on Synthetic Shapes. We call the resulting model *MagicPoint*.

Interestingly, when we evaluate MagicPoint against other traditional corner detection approaches such as FAST [21], Harris corners [8] and Shi-Tomasi's "Good Features To Track" [25] on the Synthetic Shapes dataset, we discovered a large performance gap in our favor. We measure the mean Average Precision (mAP) on 1000 held-out images of the Synthetic Shapes dataset, and report the results in Table 2. The classical detectors struggle in the presence of imaging noise – qualitative examples of this are shown in Figure 4. More detailed experiments can be found in Appendix B.

		MagicPoint	FAST	Harris	Shi
mAP	no noise	**0.979**	0.405	0.678	0.686
mAP	noise	**0.971**	0.061	0.213	0.157

Table 2. **Synthetic Shapes Detector Performance**. The MagicPoint model outperforms classical detectors in detecting corners of simple geometric shapes and is robust to added noise.

The MagicPoint detector performs very well on Syn-

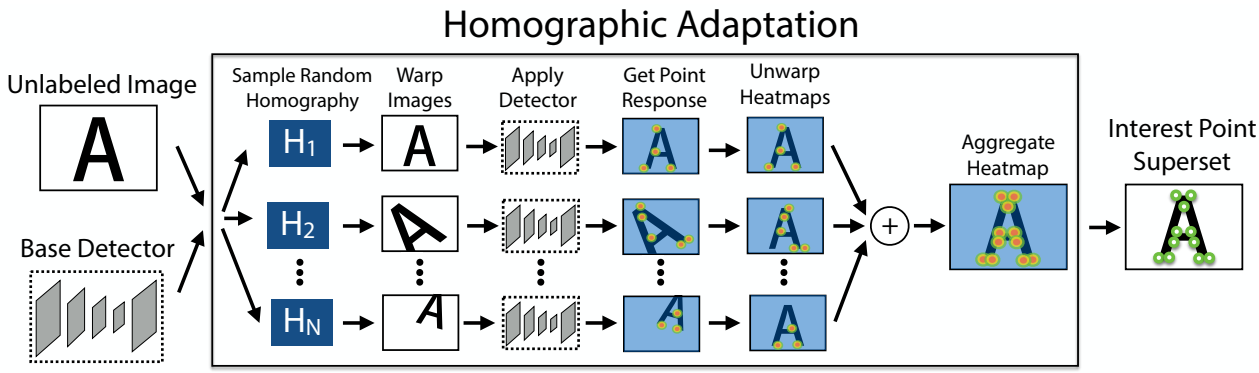

Figure 5. **Homographic Adaptation.** Homographic Adaptation is a form of self-supervision for boosting the geometric consistency of an interest point detector trained with convolutional neural networks. The entire procedure is mathematically defined in Equation 10.

thetic Shapes, but does it generalize to real images? To summarize a result that we later present in Section 7.2, the answer is *yes*, but not as well as we hoped. We were surprised to find that MagicPoint performs reasonably well on real world images, especially on scenes which have strong corner-like structure such as tables, chairs and windows. Unfortunately in the space of all natural images, it underperforms when compared to the same classical detectors on repeatability under viewpoint changes. This motivated our self-supervised approach for training on real-world images which we call Homographic Adaptation.

5. Homographic Adaptation

Our system bootstraps itself from a base interest point detector and a large set of unlabeled images from the target domain (*e.g.*, MS-COCO). Operating in a self-supervised paradigm (also known as self-training), we first generate a set of pseudo-ground truth interest point locations for each image in the target domain, then use traditional supervised learning machinery. At the core of our method is a process that applies random homographies to warped copies of the input image and combines the results – a process we call *Homographic Adaptation* (see Figure 5).

5.1. Formulation

Homographies give exact or almost exact image-to-image transformations for camera motion with only rotation around the camera center, scenes with large distances to objects, and planar scenes. Moreover, because most of the world is reasonably planar, a homography is good model for what happens when the same 3D point is seen from different viewpoints. Because homographies do not require 3D information, they can be randomly sampled and easily applied to any 2D image – involving little more than bilinear interpolation. For these reasons, *homographies are at the core of our self-supervised approach*.

Let $f_\theta(\cdot)$ represent the initial interest point function we wish to adapt, I the input image, $\mathbf{x}$ the resulting interest points and $\mathcal{H}$ a random homography, so that:

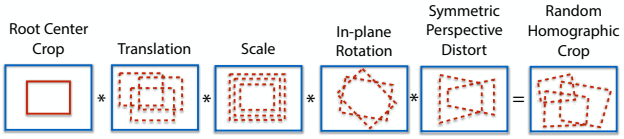

Figure 6. **Random Homography Generation.** We generate random homographies as the composition of less expressive, simple transformations.

$$\mathbf{x} = f_\theta(I). \tag{7}$$

An ideal interest point operator should be covariant with respect to homographies. A function $f_\theta(\cdot)$ is covariant with $\mathcal{H}$ if the output transforms with the input. In other words, a covariant detector will satisfy, for all $\mathcal{H}$ [3]:

$$\mathcal{H}\mathbf{x} = f_\theta(\mathcal{H}(I)), \tag{8}$$

moving homography-related terms to the right, we get:

$$\mathbf{x} = \mathcal{H}^{-1} f_\theta(\mathcal{H}(I)). \tag{9}$$

In practice, a detector will not be perfectly covariant – different homographies in Equation 9 will result in different interest points $\mathbf{x}$. The basic idea behind Homographic Adaptation is to perform an empirical sum over a sufficiently large sample of random $\mathcal{H}$'s (see Figure 5). The resulting aggregation over samples thus gives rise to a new and improved, super-point detector, $\hat{F}(\cdot)$:

$$\hat{F}(I; f_\theta) = \frac{1}{N_h} \sum_{i=1}^{N_h} \mathcal{H}_i^{-1} f_\theta(\mathcal{H}_i(I)). \tag{10}$$

5.2. Choosing Homographies

Not all 3x3 matrices are good choices for Homographic Adaptation. To sample good homographies which represent

[3] For clarity, we slightly abuse notation and allow $\mathcal{H}\mathbf{x}$ to denote the homography matrix $\mathcal{H}$ being applied to the resulting interest points, and $\mathcal{H}(I)$ to denote the entire image I being warped by $\mathcal{H}$.

plausible camera transformations, we decompose a potential homography into more simple, less expressive transformation classes. We sample within pre-determined ranges for translation, scale, in-plane rotation, and symmetric perspective distortion using a truncated normal distribution. These transformations are composed together with an initial root center crop to help avoid bordering artifacts. This process is shown in Figure 6.

When applying Homographic Adaptation to an image, we use the average response across a large number of homographic warps of the input image. The number of homographic warps N_h is a hyper-parameter of our approach. We typically enforce the first homography to be equal to identity, so that N_h=1 in our experiments corresponds to doing no adaptation. We performed an experiment to determine the best value for N_h, varying the range of N_h from small $N_h = 10$, to medium $N_h = 100$, and large $N_h = 1000$. Our experiments suggest that there is diminishing returns when performing more than 100 homographies. On a held-out set of images from MS-COCO, we obtain a repeatability score of .67 without any Homographic Adaptation, a repeatability boost of 21% when performing $N_h = 100$ transforms, and a repeatability boost of 22% when $N_h = 1000$, thus the added benefit of using more than 100 homographies is minimal. For a more detailed analysis and discussion of this experiment see Appendix C.

5.3. Iterative Homographic Adaptation

We apply the Homographic Adaptation technique at training time to improve the generalization ability of the base MagicPoint architecture on real images. The process can be repeated iteratively to continually self-supervise and improve the interest point detector. In all of our experiments, we call the resulting model, after applying Homographic Adaptation, *SuperPoint* and show the qualitative progression on images from HPatches in Figure 7.

6. Experimental Details

In this section we provide some implementation details for training the MagicPoint and SuperPoint models. This encoder has a VGG-like [27] architecture that has eight 3x3 convolution layers sized 64-64-64-64-128-128-128-128. Every two layers there is a 2x2 max pool layer. Each decoder head has a single 3x3 convolutional layer of 256 units followed by a 1x1 convolution layer with 65 units and 256 units for the interest point detector and descriptor respectively. All convolution layers in the network are followed by ReLU non-linear activation and BatchNorm normalization.

To train the fully-convolutional SuperPoint model, we start with a base MagicPoint model trained on Synthetic Shapes. The MagicPoint architecture is the SuperPoint architecture without the descriptor head. The MagicPoint

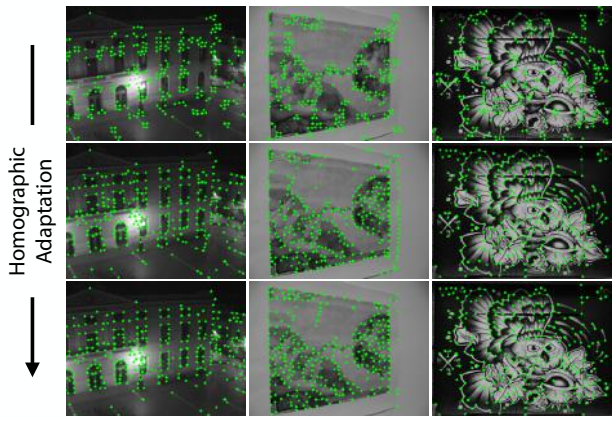

Figure 7. **Iterative Homographic Adaptation**. Top row: initial base detector (MagicPoint) struggles to find repeatable detections. Middle and bottom rows: further training with Homographic Adaption improves detector performance.

model is trained for 200,000 iterations of synthetic data. Since the synthetic data is simple and fast to render, the data is rendered on-the-fly, thus no single example is seen twice by the network.

We generate pseudo-ground truth labels using the MS-COCO 2014 [13] training dataset split which has 80,000 images and the MagicPoint base detector. The images are sized to a resolution of 240×320 and converted to grayscale. The labels are generated using Homographic Adaptation with $N_h = 100$, as motivated by our results from Section 5.2. We repeat the Homographic Adaptation a second time, using the resulting model trained from the first round of Homographic Adaptation.

The joint training of SuperPoint is also done on 240×320 grayscale COCO images. For each training example, a homography is randomly sampled. It is sampled from a more restrictive set of homographies than during Homographic Adaptation to better model the target application of pairwise matching (*e.g.*, we avoid sampling extreme in-plane rotations as they are rarely seen in HPatches). The image and corresponding pseudo-ground truth are transformed by the homography to create the needed inputs and labels. The descriptor size used in all experiments is $D = 256$. We use a weighting term of $\lambda_d = 250$ to keep the descriptor learning balanced. The descriptor hinge loss uses a positive margin $m_p = 1$ and negative margin $m_n = 0.2$. We use a factor of $\lambda = 0.0001$ to balance the two losses.

All training is done using PyTorch [19] with mini-batch sizes of 32 and the ADAM solver with default parameters of $lr = 0.001$ and $\beta = (0.9, 0.999)$. We also use standard data augmentation techniques such as random Gaussian noise, motion blur, brightness level changes to improve the network's robustness to lighting and viewpoint changes.

	57 Illumination Scenes		59 Viewpoint Scenes	
	NMS=4	NMS=8	NMS=4	NMS=8
SuperPoint	**.652**	**.631**	.503	**.484**
MagicPoint	.575	.507	.322	.260
FAST	.575	.472	.503	.404
Harris	.620	.533	**.556**	.461
Shi	.606	.511	.552	.453
Random	.101	.103	.100	.104

Table 3. **HPatches Detector Repeatability**. SuperPoint is the most repeatable under illumination changes, competitive on viewpoint changes, and outperforms MagicPoint in all scenarios.

	Homography Estimation			Detector Metrics		Descriptor Metrics	
	$\epsilon = 1$	$\epsilon = 3$	$\epsilon = 5$	Rep.	MLE	NN mAP	M. Score
SuperPoint	.310	**.684**	**.829**	.581	1.158	**.821**	**.470**
LIFT	.284	.598	.717	.449	1.102	.664	.315
SIFT	**.424**	.676	.759	.495	**0.833**	.694	.313
ORB	.150	.395	.538	**.641**	1.157	.735	.266

Table 4. **HPatches Homography Estimation.** SuperPoint outperforms LIFT and ORB and performs comparably to SIFT using various ϵ thresholds of correctness. We also report related metrics which measure detector and descriptor performance individually.

7. Experiments

In this section we present quantitative results of the methods presented in the paper. Evaluation of interest points and descriptors is a well-studied topic, thus we follow the evaluation protocol of Mikołajczyk *et al*. [16]. For more details on our evaluation metrics, see Appendix A.

7.1. System Runtime

We measure the run-time of the SuperPoint architecture using a Titan X GPU and the timing tool that comes with the Caffe [11] deep learning library. A single forward pass of the model runs in approximately 11.15 ms with inputs sized 480×640, which produces the point detection locations and a semi-dense descriptor map. To sample the descriptors at the higher 480×640 resolution from the semi-dense descriptor, it is not necessary to create the entire dense descriptor map – we can just sample from the 1000 detected locations, which takes about 1.5 ms on a CPU implementation of bi-cubic interpolation followed by L2 normalization. Thus we estimate the total runtime of the system on a GPU to be about 13 ms or **70 FPS**.

7.2. HPatches Repeatability

In our experiments we train SuperPoint on the MS-COCO images, and evaluate using the HPatches dataset [2]. HPatches contains 116 scenes with 696 unique images. The first 57 scenes exhibit large changes in illumination and the other 59 scenes have large viewpoint changes.

To evaluate the interest point detection ability of the SuperPoint model, we measure repeatability on the HPatches dataset. We compare it to the MagicPoint model (before Homographic Adaptation), as well as FAST [21], Harris [8] and Shi [25], all implemented using OpenCV. Repeatability is computed at 240×320 resolution with 300 points detected in each image. We also vary the Non-Maximum Suppression (NMS) applied to the detections. We use a correct distance of $\epsilon = 3$ pixels. Applying larger amounts of NMS helps ensure that the points are evenly distributed in the image, useful for certain applications such as ORB-SLAM [17], where a minimum number of FAST corner detections is forced in each cell of a coarse grid.

In summary, the Homographic Adaptation technique used to transform MagicPoint into SuperPoint gives a large boost in repeatability, especially under large viewpoint changes. Results are shown in Table 3. The SuperPoint model outperforms classical detectors under illumination changes and performs on par with classical detectors under viewpoint changes.

7.3. HPatches Homography Estimation

To evaluate the performance of the SuperPoint interest point detector and descriptor network, we compare matching ability on the HPatches dataset. We evaluate SuperPoint against three well-known detector and descriptor systems: LIFT [32], SIFT [15] and ORB [22]. For LIFT we use the pre-trained model (Picadilly) provided by the authors. For SIFT and ORB we use the default OpenCV implementations. We use a correct distance of $\epsilon = 3$ pixels for Rep, MLE, NN mAP and MScore. We compute a maximum of 1000 points for all systems at a 480×640 resolution and compute a number of metrics for each image pair. To estimate the homography, we perform nearest neighbor matching from all interest points+descriptors detected in the first image to all the interest points+descriptors in the second. We use an OpenCV implementation (`findHomography()` with RANSAC) with all the matches to compute the final homography estimate.

The homography estimation results are shown in Table 4. SuperPoint outperforms LIFT and ORB and performs comparably to SIFT for homography estimation on HPatches using various ϵ thresholds of correctness. Qualitative examples of SuperPoint versus LIFT, SIFT and ORB are shown in Figure 8. Please see Appendix D for even more homography estimation example pairs. SuperPoint tends to produce a larger number of correct matches which densely cover the image, and is especially effective against illumination changes.

Quantitatively we outperform LIFT in almost all metrics. LIFT is also outperformed by SIFT in most metrics. This may be due to the fact that HPatches includes indoor sequences and LIFT was trained on a single outdoor se-

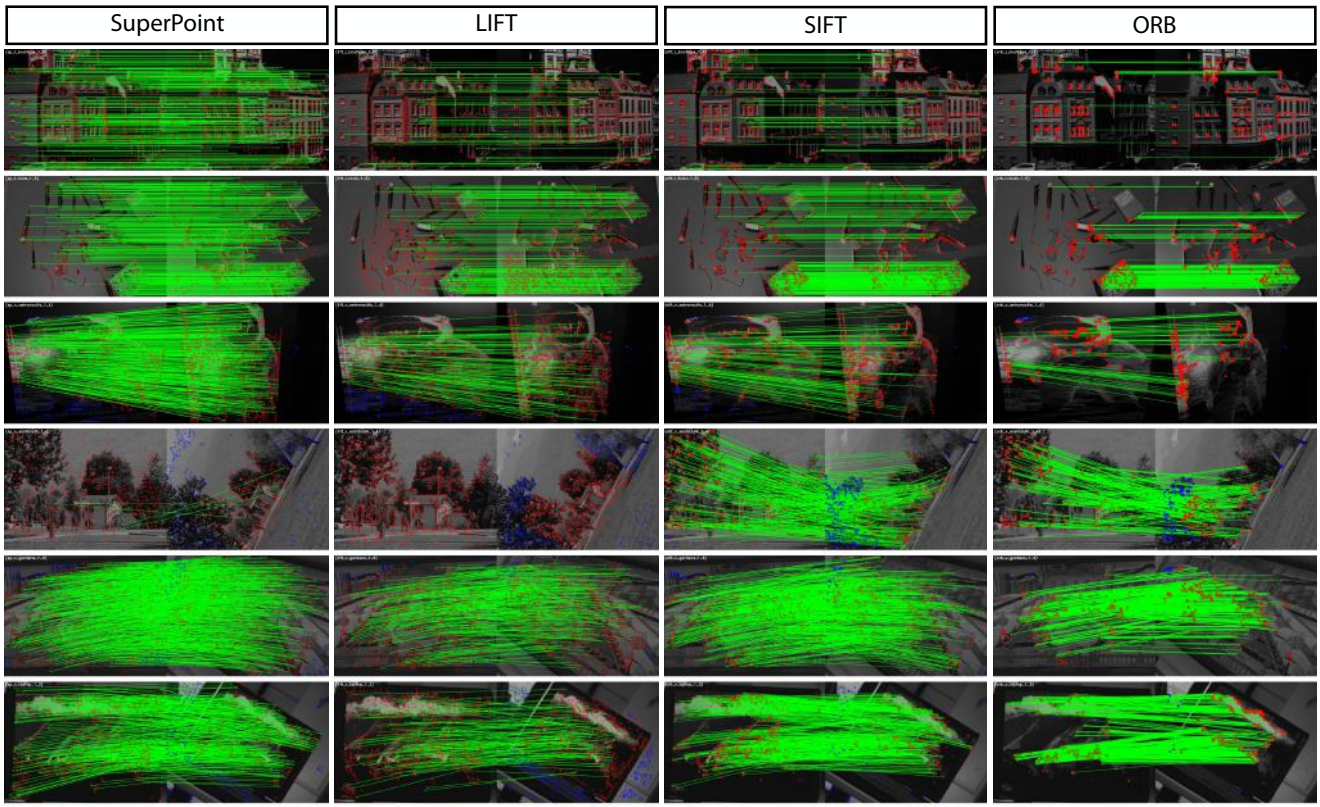

Figure 8. **Qualitative Results on HPatches**. The green lines show correct correspondences. SuperPoint tends to produce more dense and correct matches compared to LIFT, SIFT and ORB. While ORB has the highest average repeatability, the detections cluster together and generally do not result in more matches or more accurate homography estimates (see 4). Row 4: Failure case of SuperPoint and LIFT due to extreme in-plane rotation not seen in the training examples. See Appendix D for additional homography estimation example pairs.

quence. Our method was trained on hundreds of thousands of warped MS-COCO images that exhibit a much larger diversity and more closely match the diversity in HPatches.

SIFT performs well for sub-pixel precision homographies $\epsilon = 1$ and has the lowest mean localization error (MLE). This is likely due to the fact that SIFT performs extra sub-pixel localization, while other methods do not perform this step.

ORB achieves the highest repeatability (Rep.); however, its detections tend to form sparse clusters throughout the image as shown in Figure 8, thus scoring poorly on the final homography estimation task. This suggests that *optimizing solely for repeatability does not result in better matching or estimation further up the pipeline*.

SuperPoint scores strongly in descriptor-focused metrics such as nearest neighbor mAP (NN mAP) and matching score (M. Score), which confirms findings from both Choy *et al*. [3] and Yi *et al*. [32] which show that learned representations for descriptor matching outperform hand-tuned representations.

8. Conclusion

We have presented a fully-convolutional neural network architecture for interest point detection and description trained using a self-supervised domain adaptation framework called Homographic Adaptation. Our experiments demonstrate that (1) it is possible to transfer knowledge from a synthetic dataset onto real-world images, (2) sparse interest point detection and description can be cast as a single, efficient convolutional neural network, and (3) the resulting system works well for geometric computer vision matching tasks such as Homography Estimation.

Future work will investigate whether Homographic Adaptation can boost the performance of models such as those used in semantic segmentation (*e.g.*, SegNet [1]) and object detection (*e.g.*, SSD [14]). It will also carefully investigate the ways that interest point detection and description (and potentially other tasks) benefit each other.

Lastly, we believe that our SuperPoint network can be used to tackle all visual data-association in 3D computer vision problems like SLAM and SfM, and that a learning-based Visual SLAM front-end will enable more robust applications in robotics and augmented reality.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *PAMI*, 2017. 3, 8

[2] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *CVPR*, 2017. 7

[3] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal Correspondence Network. In *NIPS*. 2016. 2, 3, 8

[4] D. DeTone, T. Malisiewicz, and A. Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016. 2

[5] D. DeTone, T. Malisiewicz, and A. Rabinovich. Toward geometric deepslam. *arXiv preprint arXiv:1707.07410*, 2017. 10

[6] L. F. I. K. P. F. F. M.-N. Edgar Simo-Serra, Eduard Trulls. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 2

[7] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 2

[8] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Manchester, UK, 1988. 4, 7, 10

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision*. 2003. 1

[10] S. Honari, P. Molchanov, S. Tyree, P. Vincent, C. Pal, and J. Kautz. Improving landmark localization with semi-supervised learning. *arXiv preprint arXiv:1709.01591*, 2017. 2

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 7

[12] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich. RoomNet: End-to-end room layout estimation. In *ICCV*, 2017. 1

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and L. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 6

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 1, 8

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 2, 7

[16] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 2005. 2, 7, 10

[17] R. Mur-Artal, J. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 2015. 7

[18] A. Odena, V. Dumoulin, and C. Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 3

[19] A. Paszke, S. Gross, S. Chintala, and G. Chanan. PyTorch. https://github.com/pytorch/pytorch. 6

[20] I. Rocco, R. Arandjelović, and J. Sivic. Convolutional neural network architecture for geometric matching. In *CVPR*, 2017. 2

[21] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *ECCV*, 2006. 2, 4, 7, 10

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011. 7

[23] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys. Quad-networks: unsupervised learning to rank for interest point detection. In *CVPR*. 2017. 2

[24] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *IJCV*, 2000. 2

[25] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994. 4, 7, 10

[26] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*, 2016. 3

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 6

[28] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. DeMoN: Depth and motion network for learning monocular stereo. In *CVPR*, 2017. 2

[29] Y. Verdie, K. Yi, P. Fua, and V. Lepetit. TILDE: A Temporally Invariant Learned DEtector. In *CVPR*, 2015. 2

[30] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. SfM-Net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017. 2

[31] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016. 1

[32] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. In *ECCV*, 2016. 2, 7, 8

[33] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 2

APPENDIX

A. Evaluation Metrics

In this section we present more details on the metrics used for evaluation. In our experiments we follow the protocol of [16], with one exception. Since our fully-convolutional model does not use local patches, we instead compare detection distances by measuring the distance between the 2D detection centers, rather than measure patch overlap. For multi-scale methods such as SIFT and ORB, we compare distances at the highest resolution scale.

Corner Detection Average Precision. We compute Precision-Recall curves and the corresponding Area-Under-Curve (also known as Average Precision), the pixel location error for correct detections, and the repeatability rate. For corner detection, we use a threshold ε to determine if a returned point location $\mathbf{x}$ is correct relative to a set of K ground-truth corners $\{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_K\}$. We define the correctness as follows:

$$\texttt{Corr}(\mathbf{x}) = (\min_j ||\mathbf{x} - \hat{\mathbf{x}}_j||) \leq \varepsilon. \quad (11)$$

The precision recall curve is created by varying the detection confidence and summarized with a single number, namely the Average Precision (which ranges from 0 to 1), and larger AP is better.

Localization Error. To complement the AP analysis, we compute the corner localization error, but solely for the correct detections. We define the Localization Error as follows:

$$\texttt{LE} = \frac{1}{N} \sum_{i:\texttt{Corr}(\mathbf{x}_i)} \min_{j \in \{1,\ldots,K\}} ||\mathbf{x}_i - \hat{\mathbf{x}}_j||. \quad (12)$$

The Localization Error is between 0 and ε, and lower LE is better.

Repeatability. We compute the repeatability rate for an interest point detector on a pair of images. Since the SuperPoint architecture is fully-convolutional and does not rely on patch extraction, we cannot compute patch overlap and instead compute repeatability by measuring the distance between the extracted 2D point centers. We use ε to represent the correct distance threshold between two points. More concretely, let us assume we have N_1 points in the first image and N_2 points in the second image. We define correctness for repeatability experiments as follows:

$$\texttt{Corr}(\mathbf{x}_i) = (\min_{j \in \{1,\ldots,N_2\}} ||\mathbf{x}_i - \hat{\mathbf{x}}_j||) \leq \varepsilon. \quad (13)$$

Repeatability simply measures the probability that a point is detected in the second image.

$$\texttt{Rep} = \frac{1}{N_1 + N_2} (\sum_i \texttt{Corr}(\mathbf{x}_i) + \sum_j \texttt{Corr}(\mathbf{x}_j)). \quad (14)$$

Nearest Neighbor mean Average Precision. This metric captures how discriminating the descriptor is by evaluating it at multiple descriptor distance thresholds. It is computed by measuring Area Under Curve (AUC) of the Precision-Recall curve, using the Nearest Neighbor matching strategy. This metric is computed symmetrically across the pair of images and averaged.

Matching Score. This metric measures the overall performance of the interest point detector and descriptor combined. It measures the ratio of ground truth correspondences that can be recovered by the whole pipeline over the number of features proposed by the pipeline in the shared viewpoint region. This metric is computed symmetrically across the pair of images and averaged.

Homography Estimation. We measure the ability of an algorithm to estimate the homography relating a pair of images by comparing the estimated homography $\hat{\mathcal{H}}$ to the ground truth homography $\mathcal{H}$. It is not straightforward to compare the 3×3 $\mathcal{H}$ matrices directly, since different entries in the matrix have different scales. Instead we compare how well the homography transforms the four corners of one image onto the other. We define the four corners of the first image as $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4$. We then apply the ground truth $\mathcal{H}$ to get the ground truth corners in the second image $\mathbf{c}'_1, \mathbf{c}'_2, \mathbf{c}'_3, \mathbf{c}'_4$ and the estimated homography $\hat{\mathcal{H}}$ to get $\hat{\mathbf{c}}'_1, \hat{\mathbf{c}}'_2, \hat{\mathbf{c}}'_3, \hat{\mathbf{c}}'_4$. We use a threshold ε to denote a correct homography.

$$\texttt{CorrH} = \frac{1}{N} \sum_{i=1}^{N} \left(\left(\frac{1}{4} \sum_{j=1}^{4} ||\mathbf{c'}_{ij} - \hat{\mathbf{c}}'_{ij}|| \right) \leq \varepsilon \right). \quad (15)$$

The scores range between 0 and 1, higher is better.

B. Additional Synthetic Shapes Experiments

We present the full results of the SuperPoint interest point detector (ignoring the descriptor head) trained and evaluated on the Synthetic Shapes dataset.[4] We call this detector *MagicPoint*. The data consists of simple synthetic geometry that a human could easily label with the ground truth corner locations. We expect a good point detector to easily detect the correct corners in these scenarios. In fact, we were surprised at how difficult the simple geometries were for the classical point detectors such as FAST [21], Harris [8] and the Shi-Tomasi "Good Features to Track" [25].

We evaluated two models: *MagicPointL* and *MagicPointS*. Both models share the same encoder architecture, but differ in the number of neurons per layer. MagicPointL and MagicPointS have 64-64-64-64-128-128-128-128-128 and 9-9-16-16-32-32-32-32-32 respectively.

We created an evaluation dataset with our Synthetic Shapes generator to determine how well our detector is able

[4] An earlier version of our MagicPoint experiments can be found in our "Toward Geometric DeepSLAM" paper [5].

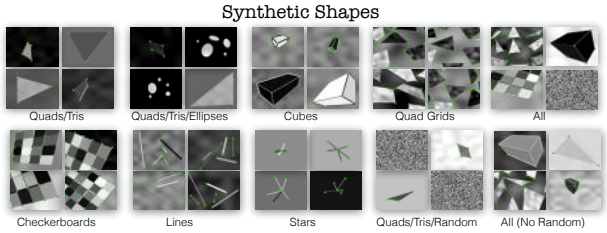

Figure 9. **Synthetic Shapes Dataset.** The Synthetic Shapes dataset consists of rendered triangles, quadrilaterals, lines, cubes, checkerboards, and stars each with ground truth corner locations. It also includes some negative images with no ground truth corners, such as ellipses and random noise images.

Metric	Noise	MagicPointL	MagicPointS	FAST	Harris	Shi
mAP	no noise	0.979	**0.980**	0.405	0.678	0.686
mAP	noise	**0.971**	0.939	0.061	0.213	0.157
MLE	no noise	**0.860**	0.922	1.656	1.245	1.188
MLE	noise	**1.012**	1.078	1.766	1.409	1.383

Table 5. **Synthetic Shapes Results Table.** Reports the mean Average Precision (mAP, higher is better) and Mean Localization Error (MLE, lower is better) across the 10 categories of images on the Synthetic Shapes dataset. Note that MagicPointL and MagicPointS are relatively unaffected by imaging noise.

to localize simple corners. There are 10 categories of images, shown in Figure 9.

Mean Average Precision and Mean Localization Error. For each category, there are 1000 images sampled from the Synthetic Shapes generator. We compute Average Precision and Localization Error with and without added imaging noise. A summary of the per category results are shown in Figure 10 and the mean results are shown in Table 5. The MagicPoint detectors outperform the classical detectors in all categories. There is a significant performance gap in mAP in all categories in the presence of noise.

Effect of Noise Magnitude. Next we study the effect of noise more carefully by varying its magnitude. We were curious if the noise we add to the images is too extreme and unreasonable for a point detector. To test this hypothesis, we linearly interpolate between the clean image ($s = 0$) and the noisy image ($s = 1$). To push the detectors to the extreme, we also interpolate between the noisy image and random noise ($s = 2$). The random noise images contain no geometric shapes, and thus produce an mAP score of 0.0 for all detectors. An example of the varying degree of noise and the plots are shown in Figure 11.

Effect of Noise Type. We categorize the noise into eight categories. We study the effect of these noise types individually to better understand which has the biggest effect on the point detectors. Speckle noise is particularly difficult for traditional detectors. Results are summarized in Figure 12.

Blob Detection. We experimented with our model's ability to detect the centers of shapes such as quadrilaterals and ellipses. We used the MagicPointL architecture (as

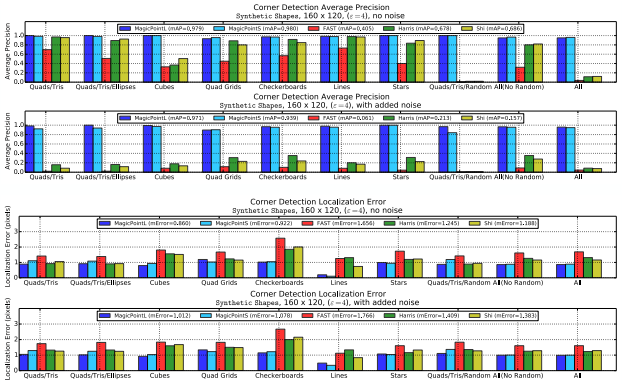

Figure 10. **Per Shape Category Results.** These plots report Average Precision and Corner Localization Error for each of the 10 categories in the Synthetic Shapes dataset with and without noise. The sequences with "Random" inputs are especially difficult for the classical detectors.

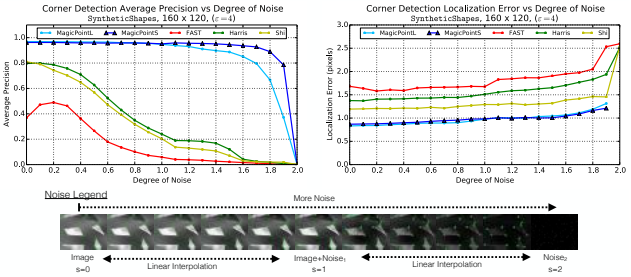

Figure 11. **Effect of Noise Magnitude.** Two versions of MagicPoint are compared to three classical point detectors on the Synthetic Shapes dataset (shown in Figure 9). The MagicPoint models outperform the classical techniques in both metrics, especially in the presence of image noise.

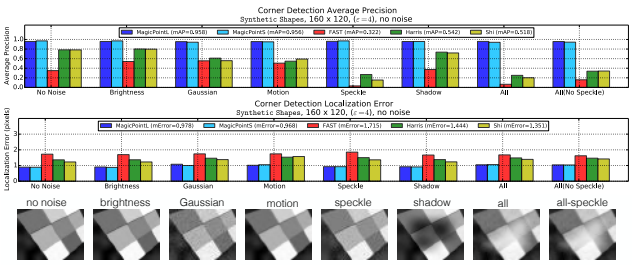

Figure 12. **Effect of Noise Type.** The detector performance is broken down by noise category. Speckle noise is particularly difficult for traditional detectors.

described above) and augmented the Synthetic Shapes training set to include blob centers in addition to corners. We observed that our model was able to detect such blobs as long as the entire shape was not too large. However, the confidences produced for such "blob detection" are typically lower than those for corners, making it somewhat cumbersome to integrate both kinds of detections into a single system. For the main experiments in the paper, we omit training with blobs, except the following experiment.

We created a sequence of 96×96 images of a black square on a white background. We vary the square's width to range from 3 to 91 pixels and report MagicPoint's confidence for two special pixels in the output heatmap: the center pixel (location of the blob) and the square's top-left pixel (an easy-to-detect corner). The MagicPoint blob+corner confidence plot for this experiment can be seen in Figure 13. We observe that we can confidently detect the center of the blob when the square is between 11 and 43 pixels wide (red region in Figure 13), detect with lower confidence when the square is between 43 and 71 pixels wide (yellow region in Figure 13), and unable to detect the center blob when the square is larger than 71 (blue regions in Figure 13).

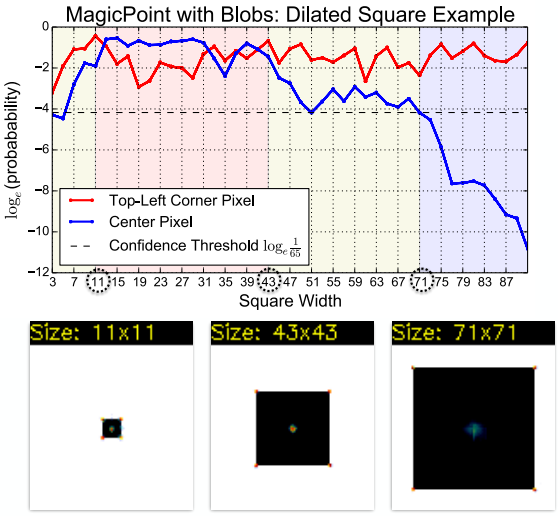

Figure 13. **MagicPoint: Blob Center Detection** Top: we experimented with MagicPoint's ability to detect the centers of shapes and plot detection confidences for both the top-left (TL) corner and the center blob. Bottom: point detection heatmaps (MagicPoint outputs) superimposed on the black rectangle images. Notice that our model is able to detect centers of 71 pixel rectangles, meaning that our network's receptive field is at least 71 pixels.

C. Homographic Adaptation Experiment

When combining interest point response maps, it is important to differentiate between within-scale aggregation and across-scale aggregation. Real-world images typically contain features at different scales, as some points which would be deemed interesting in a high-resolution images, are often not even visible in coarser, lower resolution images. However, within a single-scale, transformations of the image such as rotations and translations should not make interest points appear/disappear. This underlying multi-scale nature of images has different implications for within-scale and across-scale aggregation strategies. Within-scale aggregation should be similar to computing the intersection of a set and across-scale aggregation should be similar to the union of a set. In other words, it is the average response within-scale that we really want, and the maximum response across-scale. We can additionally use the average response across scale as a multi-scale measure of interest point confidence. The average response across scales will be maximized when the interest point is visible across all scales, and these are likely to be the most robust interest points for tracking applications.

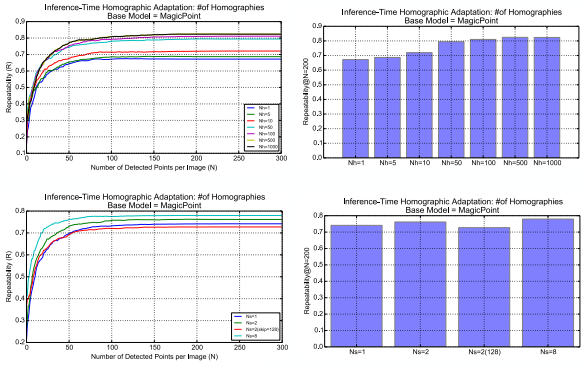

Figure 14. **Homographic Adaptation.** Top: we vary the number of homographies applied during Homographic Adaptation and report repeatability. Bottom: we isolate the effect of scale.

Within-scale aggregation. We use the average response across a large number of Homographic warps of the input image. Care should be taken in choosing random homographies because not all homographies are realistic image transformations. The number of homographic warps N_h is a hyper-parameter of our approach. We typically enforce the first homography to be equal to identity, so that $N_h = 1$ in our experiments corresponds to doing no homographies (or equivalently, applying the identity Homography). Our experiments range from "small" $N_h = 10$, to "medium" $N_h = 100$, and "large" $N_h = 1000$.

Across-scale aggregation. When aggregating across scales, the number of scales considered N_s is a hyper-parameter of our approach. The setting of $N_s = 1$ corresponds to no multi-scale aggregation (or simply aggregating across the large possible image size only). For $N_s > 1$, we refer to the multi-scale set of images being processed as "the multi-scale image pyramid." We consider weighting schemes that weigh levels of the pyramid differently, giving higher-resolution images a larger weight. This is important because interest points detected at lower resolutions have poorer localization ability, and we want the final aggregated points to be localized as well as possible.

We experimented with within-scale and across-scale aggregation on a held out test of MS-COCO images. The results are summarized in Figure 14. We find that within-scale aggregation has the biggest effect on repeatability.

D. Extra Qualitative Examples

We show extra qualitative examples of SuperPoint, LIFT, SIFT and ORB on HPatches matching in Figure 15.

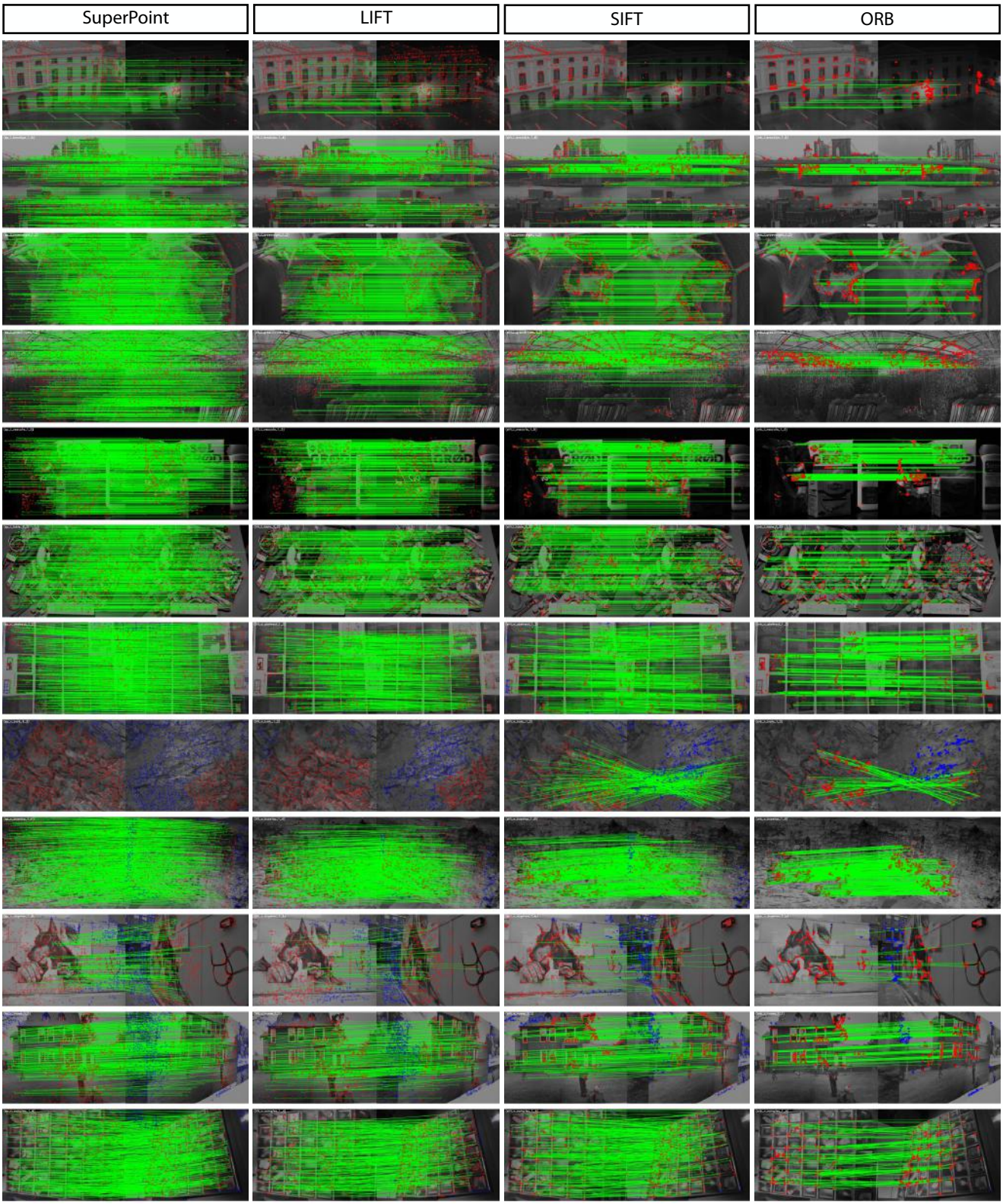

Figure 15. **Extra Qualitative Results on HPatches**. More examples like in Figure 8. The green lines show correct correspondences, green dots show matched points, red dots show mis-matched points, blue dots show points outside of the shared viewpoint region.