SuperPoint: Self-Supervised Interest Point Detection and Description

Daniel DeTone, Tomasz Malisiewicz, Andrew Rabinovich

Research @ Magic Leap

CVPR 2018 Deep Learning for Visual SLAM Workshop
Main Ideas

- “SuperPoint”
  - A Deep SLAM Frontend
  - Multi-task fully convolutional network
  - Designed for Real-time
- “Homographic Adaptation”
  - Self-supervised recipe to train keypoints
  - Synthetic pre-training
  - Homography-inspired domain adaptation
2000-2015 Visual SLAM

- Great Visual SLAM Research
- Real-time systems emerge
- Very few learned components

Collage courtesy: Andrew Davison’s ICCV 2015 Future of Real-time SLAM workshop talk
Deep Learning excitement is very high

Simple end-to-end setups work across many computer vision tasks

Purely data-driven, powerful

Very few heuristics / little hand-tuning

Accuracy not yet competitive

Maybe due to lack of large-scale data
2017-2018: Splitting Up the Problem

- **Frontend**: Image inputs
  - Deep Learning success: Images + ConvNets
  - Most of current work “deep-ifys” the Frontend -> Focus of this talk
- **Backend**: Optimization over pose and map quantities
  - 2018: Early deep learning work -> Focus of other oral at 12:05pm

Photo Credit: Cadena et al 2016
We address the problem of determining correspondences between two images in agreement with a geometric model such as an affine or thin-plate spline transformation, and estimating its parameters. The contributions of this work are three-fold. First, we propose a convolutional neural network architecture for geometric matching. The architecture is based on three main components that mimic the standard steps of feature extraction, matching and simultaneous inlier detection and model parameter estimation, while being trainable end-to-end. Second, we demonstrate that the network parameters can be trained from synthetically generated imagery without the need for manual annotation and that our matching layer significantly increases generalization capabilities to never seen before images. Finally, we show that the same model can perform both instance-level and category-level matching giving state-of-the-art results on the challenging Proposal Flow dataset.

1. Introduction

Estimating correspondences between images is one of the fundamental problems in computer vision [19, 25] with applications ranging from large-scale 3D reconstruction [3] to image manipulation [21] and semantic segmentation [42]. Traditionally, correspondences consistent with a geometric model such as epipolar geometry or planar affine transformation, are computed by detecting and matching local features (such as SIFT [38] or HOG [12, 22]), followed by pruning incorrect matches using local geometric constraints [43, 47] and robust estimation of a global geometric transformation using algorithms such as RANSAC [18] or Hough transform [32, 34, 38]. This approach works well in many cases but fails in situations that exhibit (i) large changes of depicted appearance due to, e.g., intra-class variation [22], or (ii) large changes of scene layout or non-rigid deformations that require complex geometric models with many parameters which are hard to estimate in a manner robust to outliers.

In this work we build on the traditional approach and develop a convolutional neural network (CNN) architecture that mimics the standard matching process. First, we replace the standard local features with powerful trainable convolutional neural network features [31, 46], which allows us to handle large changes of appearance between the matched images. Second, we develop trainable matching and transformation estimation layers that can cope with noisy and incorrect matches in a robust way, mimicking the good practices in feature matching such as the second nearest neighbor test [38], neighborhood consensus [43, 47] and Hough transform-like estimation [32, 34, 38].

The outcome is a convolutional neural network architecture trainable for the end task of geometric matching, which can handle large appearance changes, and is therefore suitable for both instance-level and category-level matching problems.

2. Related work

The classical approach for finding correspondences involves identifying interest points and computing local descriptors around these points [10, 11, 24, 37, 38, 39, 43].
2017-2018 Deep Frontends: Sparse

Existing Patch-based Systems

- Sliding Window
- Input Patches
- Descriptors

Deep Network A
Deep Network B

- Most low-compute Visual SLAM built on sparse frontends
- Extract points -> “Backend Ready”
- Most learned systems patch-based
  - Two separate networks
  - Lack powerful matchability of dense methods

LIFT: Learned Invariant Feature Transform
QuadNetworks: Unsupervised Learning to Rank for Interest Point Detection

LF-Net: Learning Local Features from Images

- Most low-compute Visual SLAM built on sparse frontends
- Extract points -> “Backend Ready”
- Most learned systems patch-based
  - Two separate networks
  - Lack powerful matchability of dense methods
Question

- How can we get the power of dense matchability and the practicality of sparse output in a learnable framework?
SuperPoint: A Deep SLAM Front-end

- Powerful fully convolutional design
  - Points + descriptors computed jointly
  - Share VGG-like backbone
- Designed for real-time
  - Tasks share ~90% of compute
  - Two learning-free decoders: no deconvolution layers
Keypoint / Interest Point Decoder

- No deconvolution layers
- Each output cell responsible for local 8x8 region
Descriptor Decoder

- Also no deconvolution layers
- Interpolate using 2D keypoint into coarse descriptor map
How To Train SuperPoint?

Image → ConvNet → Keypoint 2D Locations, Keypoint Descriptors
Setting up the Training

- Siamese training -> pairs of images
- Descriptor trained via metric learning
- Keypoints trained via supervised keypoint labels
How to get Keypoint Labels for Natural Images?

- Need large-scale dataset of annotated images
- Too hard for humans to label
Self-Supervised Approach

Synthetic Shapes (has interest point labels)

First train on this

“Homographic Adaptation”

MS-COCO (no interest point labels)

Use resulting detector to label this
Synthetic Training

- Non-photorealistic shapes
- Heavy noise
- Effective and easy

- Quads/Tris
- Quads/Tris/Ellipses
- Cubes
- Checkerboards
- Lines
- Stars
Early Version of SuperPoint (MagicPoint)

“Toward Geometric Deep SLAM”
DeTone et al. 2017

Corner Detection Average Precision vs Degree of Noise
Synthetic Shapes, 160 x 120, (ε = 4)

Noise Legend

More Noise

Image s=0

Linear Interpolation

Image+Noise1 s=1

Linear Interpolation

Noise2 s=2

Metric Noise MagicPointL MagicPointS FAST Harris Shi

Effect of Noise Filters

Effect of Noise Type

Effect of Noise Magnitude

Mean Average Precision and Mean Localization Error with and without added images, shown in Figure 9.

For each category, there are 1000 images sampled from the Synthetic Shapes generator. We compute Average Precision and Corner Localization Error for each of the 10 categories of images.

We categorized the noise into eight categories. We study the effect of these noise types individually to better understand which has the biggest effect on performance. We were curious if the noise we add to the images is too extreme and that our model was able to detect such blobs as long as the entire shape was not too large. However, the confidence produced for such “blob detection” are typically unreasonable for a point detector. To test this hypothesis, we linearly interpolate between the clean image and the noisy image (image+noise). The random noise images contain no geometric shapes, and thus produce an mAP score of 0.979 for all detectors. An example of the varying degree of noise is shown in Figure 12.

Per Shape Category Results

Table 5.

<table>
<thead>
<tr>
<th>Shape Category</th>
<th>MagicPointL</th>
<th>MagicPointS</th>
<th>FAST</th>
<th>Harris</th>
<th>Shi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cubes</td>
<td>0.980</td>
<td>1.078</td>
<td>1.766</td>
<td>1.409</td>
<td>1.383</td>
</tr>
<tr>
<td>Quad Grids</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quads/Tris</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stars</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checkboard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quads/Tris/Ellipses</td>
<td>0.939</td>
<td>0.061</td>
<td>0.213</td>
<td>0.157</td>
<td></td>
</tr>
<tr>
<td>Lines/Ellipses</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines/Quads</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines/Quads/Random</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines/Quads/Random/All</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Metrics No MagicPointL MagicPointS FAST Harris Shi

Mean Average Precision

Effect of Noise Magnitude

The plots report Average Precision and Corner Localization Error for each of the 10 categories.

For each category, there are 1000 images sampled from the Synthetic Shapes train 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.
Generalizing to Real Data

- Synthetically trained detector
  - Works! Despite large domain gap
  - Worked well on geometric structures
  - Under performed on certain textures unseen during training
Homographic Adaptation

- Simulate planar camera motion with homographies
- Self-labelling technique
  - Suppress spurious detections
  - Enhance repeatable points
Iterative Homographic Adaptation

- Label, train, repeat, ...
- Resulting points:
  - Higher coverage
  - More repeatable
HPatches Evaluation

- Homography estimation task
- Dataset of 116 scenes each with 6 images = 696 images
- Indoor and outdoor planar scenes
- Compared against LIFT, SIFT and ORB

50% of dataset: **Illumination** Change

50% of dataset: **Viewpoint** Change
Qualitative Illumination Example

- SuperPoint -> denser set of correct matches
- ORB -> highly clustered matches
Qualitative Viewpoint Example #1

- Similar story
Qualitative Viewpoint Example #2

- In-plane rotation of ~35 degrees
# HPatches Evaluation

## Core Task

<table>
<thead>
<tr>
<th>Detector</th>
<th>Homography Estimation</th>
<th>SuperPoint</th>
<th>LIFT</th>
<th>SIFT</th>
<th>ORB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0.684</td>
<td>0.598</td>
<td>0.676</td>
<td>0.395</td>
</tr>
</tbody>
</table>

## Sub-metrics

<table>
<thead>
<tr>
<th></th>
<th>Descriptor Metrics</th>
<th>Detector Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NN mAP</td>
<td>M. Score</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SuperPoint</td>
<td>0.821</td>
<td>0.470</td>
</tr>
<tr>
<td>LIFT</td>
<td>0.664</td>
<td>0.315</td>
</tr>
<tr>
<td>SIFT</td>
<td>0.694</td>
<td>0.313</td>
</tr>
<tr>
<td>ORB</td>
<td>0.735</td>
<td>0.266</td>
</tr>
</tbody>
</table>
Timing SuperPoint vs LIFT

- Speed important for low-compute Visual SLAM
  - SuperPoint total 640x480 time: ~ 33 ms
  - LIFT total 640x480 time: ~2 minutes
3D Generalizability of SuperPoint

- Trained+evaluated on planar, does it generalize to 3D?
- “Connect-the-dots” using nearest neighbor matches
- Works across many datasets / input modalities / resolutions!

Freiburg (Kinect)  NYU (Kinect)  MonoVO (fisheye)  ICL-NUIM (synth)

MS7 (Kinect)  KITTI (stereo)
New Announcement, Research @ MagicLeap

Public Release of Pre-trained Net:

github.com/MagicLeapResearch/SuperPointPretrainedNetwork

- Sparse Optical Flow Tracker Demo
- Implemented in Python + PyTorch
- Two files, minimal dependencies
- Easy to get up and running
Take-Aways

- “SuperPoint”: A Modern Deep SLAM Frontend
  - Non-patch based fully convolutional network
  - Real-time deployability
- Self-supervised recipe to train keypoints
  - Synthetic pre-training
  - Homography-inspired domain adaptation
- Public code available to run SuperPoint
Thank You
Questions?

SuperPoint: A Modern Deep SLAM Front-end

Image → ConvNet → Keypoint 2D Locations → Keypoint Descriptors
Extra Slides
Failure Mode: Extreme Rotation

- Extreme in-plane rotations
- Trained for ~30 deg rotations
- Optimized tracking scenarios
- LIFT also struggles, despite learned orientation estimation
Iterative Homographic Adaptation

MagicPoint

SuperPoint