From Invariant Descriptors to Deep Pose Estimation

K. Yi, E. Trulls, V. Lepetit, and P. Fua

SIFT

LIFT
Feature Points

Outstanding tool for matching points across images.

SIFT (Lowe, ICCV’99) started the trend: ~48k citations.
Local Feature Pipeline
Local Feature Pipeline

Local Feature Pipeline

- Harris, C., Stephens, M., "A Combined Corner and Edge Detector," AVC, 1988
- Förstner, W., Dickscheid, T., Schindler, F., "Detecting Interpretable and Accurate Scale-Invariant Keypoints," ICCV, 2009
- Zitnick, C., Ramnath, K., "Edge Foci Interest Points", ICCV, 2011
- Dominant Gradient Orientations (SIFT, SURF,...)
- Center of Mass (ORB,...)

Deep Learning Revolution

An opportunity to revisit and improve the pipeline:

• Reformulate its different components in terms of CNNs.
• Integrate them into a fully differentiable pipeline.
• Optimize them jointly.
1. Detecting Keypoints

TILDE: a Temporally Invariant Learned DEtector
(CVPR 2015)
Hand-Designed Features under Severe Illumination Changes

It’s not going to work well even using very good descriptors!

Matching with SURF

—> Poor repeatability.
Learning to find Keypoints that Are Robust to Illumination Changes.

Matching with TILDE
Learning from Aligned Image Stacks

- Pre-align images of a scene.
- Find locations that are often detected by a given feature detector.
- Train a CNN regressor to find these locations.
Examples

Images to match  SIFT  SURF  FAST  TILDE
Matching 5 days of the *Frankfurt* sequence with our keypoints
Quantitative Results
Webcam Dataset

Repeatability (%)
Keypoint Detection in Short

- Keypoint repeatability is crucial for many applications
- We can train a regressor to find repeatable keypoints.
2. Estimating Orientation

Learning to Assign Orientations to Feature Points
(CVPR 2016)
Local Feature Pipeline

- Harris, C., Stephens, M., "A Combined Corner and Edge Detector," AVC, 1988
- Förstner, W., Dickscheid, T., Schindler, F., "Detecting Interpretable and Accurate Scale-Invariant Keypoints," ICCV, 2009
- Zitnick, C., Ramnath, K., "Edge Foci Interest Points," ICCV, 2011

- Dominant Gradient Orientations (SIFT, SURF,...)
- Center of Mass (ORB,...)


Important but largely overlooked
Local Feature Pipeline

Image → Local Feature Detector → Orientation Estimation → Feature Descriptor

?
There is no such thing as a canonical orientation
Implicit Orientations

Learn to estimate **consistent** and **optimal** orientations for matching purposes.
Deep Siamese Network for Learning Orientation

minimize $\mathcal{L}(\text{Pair}) = \left\| \text{Desc}(\text{Patch}_1, \text{Orient}(\text{Patch}_1)) \right\|$

$- \text{Desc}(\text{Patch}_2, \text{Orient}(\text{Patch}_2))$

Orient($\text{patch}_1$) = arctan2(CNN($\text{patch}_1$) [1], CNN($\text{patch}_1$) [2])

Desc(·) is not learned. Any rotation sensitive descriptor can be used.
Matching Examples

Our Learned Orientations

Dominant Gradient Orientations
Matching Examples

Our Learned Orientations

Dominant Gradient Orientations
Quantitative Evaluation

86 sequences, 855 images

Descriptor matching performances (mAP) with nearest neighbor matching (Mikolajczyk & Schmid, IJCV’04).

Performance Gain with Learned Orientations

Average performance

<table>
<thead>
<tr>
<th></th>
<th>mAP (higher the better)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF-SIFT</td>
<td>0.30</td>
</tr>
<tr>
<td>EF-SIFT+</td>
<td>0.35</td>
</tr>
<tr>
<td>EF-SIFT+ (Ours-ReLU)</td>
<td>0.40</td>
</tr>
<tr>
<td>EF-SIFT+ (Ours-GHH)</td>
<td>0.45</td>
</tr>
</tbody>
</table>
Estimating Orientation in Short

• Orientations are a key component in the local feature pipeline that has been largely overlooked.

• We have proposed a Deep Learning based approach to learn good orientations for matching purposes.

• This delivers significant performance improvements in matching performance.
3. Computing Descriptors

Discriminative Learning of Deep Convolutional Feature Point Descriptors (ICCV 2015)
Siamese Network

- Minimize the distance for corresponding matches.
- Maximize it for non-corresponding patches.
Our Network

- 3 convolutional layers, no fully-connected layers.
- About 45k parameters.
- Hard mining is key to good performance.

—> After training, a drop-in replacement for SIFT.
Training and Testing Data

- MVS dataset (Brown et al, PAMI’11), 3 SfM sets of 64x64 grayscale patches. Each one contains ~150k 3D points and ~450k patches.
- Train on two and test on the third.

Statue of Liberty (LY)

Notre Dame (ND)

Yosemite (YO)
Quantitative Results

PR curve, training LY+YO, test ND

PR curve, training LY+ND, test YO

PR curve, training YO+ND, test LY

PR curve, training "all"

#2 on Yosemite

#1 training will all three sets
Descriptors in Short

• **Outperforms** both hand-crafted descriptors and state-of-the-art, learned descriptors.

• Good **generalization properties**: scaling, rotation, deformation, illumination changes.

• **Fast**: 0.76 ms on GPU, vs 0.14 ms for dense SIFT.

• No metric learning → **Drop-in replacement for SIFT**.
4. Putting it all Together

LIFT: Learned Invariant Feature Transform (ECCV 2016)
All three main components are now CNNs.
Tie everything together using **differentiable modules**:

- Soft Argmax (Chapelle et al., Information Retrieval’09)
- Crop and Rotate (Spatial Transformer Networks, Jaderberg et al., NIPS’15)

--> End-to-end differentiability.
Training the pipeline

Image → Local Feature Detector → Orientation Estimation → Feature Descriptor

Needs: keypoints (pos) & non-keypoints (neg)

Needs: pairs of corresponding (pos) & non-corresponding (neg) patches
Training with SfM Keypoints

• We need variability (illumination, perspective, etc). We build SfM reconstructions from photo-tourism sets.

• We keep only points with SfM correspondences as positive examples, that is, we learn to find repeatable points.
Quadruplet Siamese

- Use patches around SIFT locations.
- Perturb patch locations to avoid biases.

Still too hard to train from scratch!

- Components compete with each other, e.g. detectors aim for distinctiveness, descriptors for invariance.
- We propose a problem-specific strategy.

\( P_1, P_2: \) corresponding keypoints.
\( P_3: \) non-corresponding keypoint. \( P_4: \) non-keypoint.
Problem-Specific Training

1. Train the **Descriptor** using SfM (SIFT) patches.

2. Train the **Orientation Estimator** given the pre-trained descriptor.

3. Train the **Detector** with the pre-trained Orientation Estimator and Descriptor.

End-to-end differentiability is essential!
• The **Detector** runs in scale-space with traditional NMS.

• Keypoints are passed on to the **Orientation Estimator and Descriptor** modules.

• Our TensorFlow GPU-based implementation takes \(~3.0s\) on a 1600x1200 image, with an additional \(~2.6\) sec. of pure Python non-maximum suppression. On the same machine, SIFT takes \(~2\) sec (CPU, multi-threaded)
Matching features on DTU sequence #19. Correct matches depicted by green lines.

SIFT. Average: 34.1 matches

LIFT (Ours). Average: 98.5 matches
Matching features on **Webcam** sequence **Frankfurt**.
Correct matches depicted by **green** lines.

**SIFT.** Average: **23.1** matches

**LIFT (Ours).** Average: **60.6** matches
Quantitative Evaluation

Strecha (2 seq.)
- Outdoors.
- Wide-baseline stereo

DTU (60 seq.)
- Objects.
- Perspective changes.

Webcam (5 seq.)
- Outdoors.
- Fixed view, drastic illumination changes.

• **Metric:** Descriptor matching performances (mAP) with nearest neighbor matching (Mikolajczyk & Schmid, IJCV’04) as before.
Quantitative Evaluation

- **Best performance** on all datasets, with either ‘pic’ or ‘rf’.

- **SIFT remains #3** overall (#1: ours, #2: VGG).
<table>
<thead>
<tr>
<th># Images</th>
<th># Registered</th>
<th># Sparse Points</th>
<th># Observations</th>
<th>Track Length</th>
<th>Reproj. Error</th>
<th># Inlier Pairs</th>
<th># Inlier Matches</th>
<th># Dense Points</th>
<th>Pose Error</th>
<th>Dense Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>10,004</td>
<td>44K</td>
<td>4.47</td>
<td>0.39px</td>
<td>49</td>
<td>76K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.71 (0.01)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>14,468</td>
<td>70K</td>
<td>4.89</td>
<td>0.39px</td>
<td>57</td>
<td>124K</td>
<td>3.72K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>14,379</td>
<td>67K</td>
<td>4.75</td>
<td>0.39px</td>
<td>54</td>
<td>124K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>13,519</td>
<td>61K</td>
<td>4.55</td>
<td>0.39px</td>
<td>54</td>
<td>124K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>13,698</td>
<td>64K</td>
<td>4.66</td>
<td>0.39px</td>
<td>54</td>
<td>100K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10,172</td>
<td>46K</td>
<td>4.65</td>
<td>0.39px</td>
<td>54</td>
<td>80K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6,726</td>
<td>27K</td>
<td>4.09</td>
<td>0.39px</td>
<td>26</td>
<td>30K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7,760</td>
<td>32K</td>
<td>4.19</td>
<td>0.39px</td>
<td>26</td>
<td>30K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8,090</td>
<td>20K</td>
<td>5.21</td>
<td>0.39px</td>
<td>26</td>
<td>30K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6,418</td>
<td>25K</td>
<td>4.09</td>
<td>0.39px</td>
<td>26</td>
<td>30K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6,606</td>
<td>27K</td>
<td>4.09</td>
<td>0.39px</td>
<td>26</td>
<td>30K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7,634</td>
<td>30K</td>
<td>5.93</td>
<td>0.39px</td>
<td>26</td>
<td>40K</td>
<td>2.97K</td>
<td>0.082a (0.02a)</td>
<td>0.77 (0.01)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Results for our reconstruction benchmark. Pose error as mean (median) over all images. Dense error for 2cm (10cm) threshold [19].

First, second, third best results highlighted in bold. Number of images, sparse points, and dense points visualized in Figs. 1, 2, and 3.
Keypoints are only a means to an end!

• LIFT maximises the number of matches.
• Not all of them are useful.
  --> Need a good way to learn which ones are.
Three of the four main components are now CNNs. They have now been integrated into a single pipeline. 

→ Must now work on the fourth!
4. Correspondences
RANSAC + 5 point method is not enough
Deep Learning to the Rescue

Learn to reject outliers and estimate the Essential matrix simultaneously.

→ Incorporate global context into the matching process.

Hartley and Zisserman, 2003
Revisiting the 8-point Algorithm

N Correspondences

N x 9 Matrix

x1x2, y1y2, ...

\( X^T X \)

9x9 Matrix

Eigen/SVD

Essential Matrix
Simultaneous Classification and Regression

Deep Network
MLP + Global Context Normalization

\[ W \]

\[ W^T w X \]

\[ X^T w X \]

\[ X \]

N x 1 Vector

N x 9 Matrix

9x9 Matrix

Eigen/SVD

Essential Matrix

N Correspondences
Multi-Layer Perceptron with Global Context Normalization (GCN)

- Captures the global context at every layer output.
- Invariant to permutations of the input.
- Loss is the sum of a classification and a regression term.

Correspondences between image pairs

Batch

Global Context

N Correspondences

4 dim
Outlier rejection

RANSAC
Outlier rejection

Grid-Based Motion Statistics
Outlier rejection

Our results
Quantitative Results

Mean Accuracy = ratio of pairs below error threshold of X, while X goes from zero to 20 (degrees) —> AUC

1000 randomly chosen pairs from Yahoo Flickr Creative Commons 100 millions
Conclusion

• We implemented the **full keypoint extraction pipeline** using Deep Networks while preserving end-to-end differentiability.

• We showed how to train it *effectively* and *outperform* the state-of-the-art.

• We are now working on reformulating the extraction and matching problem as end-to-end trainable CNN.
Software

Source code and pre-trained models are available for every component of the pipeline:

✓ TILDE detector:
  • [github.com/cvlab-epfl/TILDE](https://github.com/cvlab-epfl/TILDE)

✓ Orientation estimator:
  • [github.com/cvlab-epfl/learn-orientation](https://github.com/cvlab-epfl/learn-orientation)

✓ Descriptors:
  • [github.com/cvlab-epfl/deepdesc-release](https://github.com/cvlab-epfl/deepdesc-release)

✓ One LIFT to rule them all:
  • [github.com/cvlab-epfl/tf-lift](https://github.com/cvlab-epfl/tf-lift)
Thank you. Questions?