Supplementary Appendix for LIFT: Learned Invariant Feature Transform

Kwang Moo Yi*1, Eduard Trulls*1, Vincent Lepetit2, Pascal Fua1

1Computer Vision Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL)
2Institute for Computer Graphics and Vision, Graz University of Technology
{kwang.yi, eduard.trulls, pascal.fua}@epfl.ch, lepetit@icg.tugraz.at

1 Supplementary Appendix

In this appendix, we provide the implementation details necessary to make our experimental results reproducible. Second, we provide references to the algorithms used as baselines.

1.1 Implementation details

Table 1 details the parameters of the different components in our LIFT network.

<table>
<thead>
<tr>
<th>Conv. filter</th>
<th># of nodes</th>
<th>Pooling</th>
<th>Activation</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kp-1</td>
<td>25 × 25</td>
<td>1</td>
<td>—</td>
<td>GHH, N = 4, M = 4</td>
</tr>
<tr>
<td>Kp-2</td>
<td>softargmax</td>
<td>Mapping</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Kp-3</td>
<td>Spatial Transformer ‘Crop’</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Ori-1</td>
<td>5 × 5</td>
<td>10</td>
<td>Max pooling</td>
<td>ReLU</td>
</tr>
<tr>
<td>Ori-2</td>
<td>5 × 5</td>
<td>20</td>
<td>Max pooling</td>
<td>ReLU</td>
</tr>
<tr>
<td>Ori-3</td>
<td>3 × 3</td>
<td>50</td>
<td>Max pooling</td>
<td>ReLU</td>
</tr>
<tr>
<td>Ori-4</td>
<td>1 × 1</td>
<td>100</td>
<td>—</td>
<td>GHH, N = 4, M = 4</td>
</tr>
<tr>
<td>Ori-5</td>
<td>1 × 1</td>
<td>2</td>
<td>—</td>
<td>GHH, N = 4, M = 4</td>
</tr>
<tr>
<td>Ori-6</td>
<td>arctan</td>
<td>Mapping</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Ori-7</td>
<td>Spatial Transformer ‘Rot’</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Desc-1</td>
<td>7 × 7</td>
<td>32</td>
<td>l2 pooling, 2 × 2</td>
<td>Tanh 2</td>
</tr>
<tr>
<td>Desc-2</td>
<td>6 × 6</td>
<td>64</td>
<td>l2 pooling, 3 × 3</td>
<td>Tanh 3</td>
</tr>
<tr>
<td>Desc-3</td>
<td>5 × 5</td>
<td>128</td>
<td>l2 pooling, 4 × 4</td>
<td>Tanh 4</td>
</tr>
</tbody>
</table>

Table 1. The LIFT network architecture.

During the descriptor training, we use Stochastic Gradient Descent with momentum. With the trained descriptor, we train the Orientation Estimator with the ADAM [1] solver. Finally, the Detector is trained in conjunction with the two components using ADAM as well.

* First two authors contributed equally.
This work was supported in part by the EU FP7 project MAGELLAN under grant number ICT-FP7-611526.
1.2 Implementations of the Compared Methods

For fair comparison, we used the authors implementations where applicable, as well as the publicly available ones. All methods were used with default parameters, unless stated otherwise.

- SIFT [2]: OpenCV library.
  http://opencv.org/downloads.html
- SIFT-HesAff [3]: VLFeat library.
  http://www.vlfeat.org/download.html
- SURF [4]: OpenCV library.
  http://opencv.org/downloads.html
- KAZE [5]: Implementation by the authors.
  https://github.com/pablofdezalc/kaze
- ORB [6]: OpenCV library.
  http://opencv.org/downloads.html
  To obtain enough keypoints we used nFeatures=1000 and nLevels=3.
- Daisy [7]: Provided by the authors.
  https://github.com/etola/libdaisy
  Patches were extracted to be four times the scale, which was the value authors used in [8].
- sGLOH [9]: Implementation by the authors.
  http://www.math.unipa.it/fbellavia/htm/research.html
- MROGH [10]: Implementation by the authors.
  https://github.com/bfan/MROGH-feature-descriptor
- Edge Foci [8] and BiCE [7]: Implementation by the authors.
  http://research.microsoft.com/en-us/downloads/ae6ab93a-4e17-46b9-a1e4-907ae64d1e5/
- BRISK [11]: Implementation by the authors.
  http://www.asl.ethz.ch/people/lestefan/personal/BRISK
  To obtain enough keypoints we used threshold of 20.
- FREAK [12]: Implementation by the authors.
  https://github.com/kikohs/freak
- VGG [13]: Implementation by the authors.
  http://www.robots.ox.ac.uk/~vgg/software/learn_desc/
  http://www.vlfeat.org/download.html
  Patches were extracted with the VLFeat library, with a relativeExtent of 7.5, which is the same as what SIFT uses. We use the pre-learned model learned with the liberty dataset from [13].
- MatchNet [14]: Implementation by the authors.
  https://github.com/hanxf/matchnet
  We use the models learned with the liberty dataset.
- DeepDesc [15]: Implementation by the authors.
  https://github.com/etrulls/deepdesc-release
  We used the models trained on split #4.
- PN-Net [16]: Implementation by the authors.
  https://github.com/vbalnt/pnnet
  We used the networks trained on the liberty dataset.
References