Lecture 3: LIFT, SuperPoint
LIFT: Learned Invariant Feature Transform
Keypoint detection, orientation estimation and feature descriptor are all learned.
Descriptor learning

Use locations and orientations of SIFT keypoints for training descriptor:

\[ d = h_\rho(p_\theta) \]

Rotated patch

Use metric learning for training:

\[
L_{\text{desc}}(p^k_\theta, p^l_\theta) = \begin{cases} 
\| h_\rho(p^k_\theta) - h_\rho(p^l_\theta) \|_2 & \text{for positive pairs,} \\
\max(0, C - \| h_\rho(p^k_\theta) - h_\rho(p^l_\theta) \|_2) & \text{for negative pairs}
\end{cases}
\]

Hard negative mining:

- Forward pass a few pairs and evaluate losses
- Only use a fraction of highest losses for backpropagation
Orientation estimator learning

Orientation estimator produces an angle to rotate a patch:

$$\theta = g_\phi(p)$$

Let $G(P, x)$ be patch centered at $x$ after orientation normalization of patch $P$.

Loss for training orientation estimator:

$$L_{orientation}(P^1, x^1, P^2, x^2) = \| h_\rho(G(P^1, x^1)) - h_\rho(G(P^2, x^2)) \|_2$$

Want the descriptors to align after orientation correction.
Detector is not trained yet, so still use SFM points for $x$. 
Detector training

Obtain score map for input match (piecewise linear activations on convolutional output):

\[ S = f_\mu(P) = \sum_n^{N} \delta_n \max_m \left(W_{mn} * P + b_{mn}\right) \]

Use softmax for differentiable variant of non-maximal suppression:

\[ x = \text{softargmax} \left(S\right) = \frac{\sum_y \exp(\beta S(y)) y}{\sum_y \exp(\beta S(y))} \]

\( P^1, P^2 \): from same 3D keypoint in two views, \( P^3 \): from another keypoint, \( P^4 \): non-feature point. Detector trained on a joint loss:

\[ \mathcal{L}_{\text{detector}}(P^1, P^2, P^3, P^4) = \gamma \mathcal{L}_{\text{class}}(P^1, P^2, P^3, P^4) + \mathcal{L}_{\text{pair}}(P^1, P^2) \]

Pair loss: projections of same 3D point should have similar descriptors

\[ \mathcal{L}_{\text{pair}}(P^1, P^2) = \| h_\rho(G(P^1, \text{softargmax}(f_\mu(P^1)))) - h_\rho(G(P^2, \text{softargmax}(f_\mu(P^2)))) \|_2 \]

Classification loss: push score map to high values for positive classes

\[ \mathcal{L}_{\text{class}}(P^1, P^2, P^3, P^4) = \sum_{i=1}^{4} \alpha_i \max \left(0, \left(1 - \text{softmax} \left(f_\mu \left(P^i\right) \right) y_i\right)\right)^2 \]

\( y_i = +1 \) for \( i = 1, 2, 3 \) and \( y_i = -1 \) for \( i = 4 \)
Test-time pipeline

- Run detector independently
- Apply traditional non-maximum suppression to obtain patches
- Estimate orientations and descriptors on detected patches
Metrics

- **Repeatability**: fraction of keypoints common between images 1 and 2
- **Mean AP**: vary threshold, plot precision and recall, compute area under curve
- **Matching score**: ratio of correspondences and keypoints
SIFT performs very competitively for detection, also close for descriptor computation.
Qualitative results
Differences for LIFT, compared to UCN

- Learns interest point detector too, along with descriptor
- Patch-based matching approach
- Sparse keypoints instead of dense
- Trained on SIFT keypoints matched by SFM
- Shallower architecture compared to UCN
- Uses more complex activation functions
- At test-time, detector run separately
SuperPoint
**Motivation**

- UCN uses ground truth LIDAR points for training
- LIFT computes interest points, but effectively uses SIFT as ground truth
- **SuperPoint**: use synthetic data to generate pseudo ground truth

- Generalization to real images is challenging
- There is no ground truth for real images
- **SuperPoint**: self-training may be used to improve performance

- Frameworks like UCN do not compute interest points (dense correspondence)
- LIFT has two different steps for interest points and descriptors
- **SuperPoint**: share computation and representations across the two tasks
Architecture: encoder

- VGG-like encoder, 3 max-pooling layers
- Each “pixel” in encoded output maps to 8x8 region of input image
Architecture: interest point decoder

- Compute an $R^{H_c \times W_c \times 65}$ tensor
- Channels correspond existence of interest point in each pixel of 8x8 region
- One more channel corresponds to no interest point
- Channel-wise softmax to determine probability of interest point location
- Reshape to original $R^{H \times W}$ image size
Loss function: interest point decoder

- Let $x_{hw} \in \mathcal{X}$ be a cell of the interest point tensor.
- Ground truth interest point labels given by $y_{hw}$.
- Channel-wise softmax:

$$l_p(x_{hw}; y) = -\log \left( \frac{\exp(x_{hwy})}{\sum_{k=1}^{65} \exp(x_{hwk})} \right)$$

- Training loss:

$$\mathcal{L}_p(\mathcal{X}, Y) = \frac{1}{H_c W_c} \sum_{h=1}^{H_c} \sum_{w=1}^{W_c} l_p(x_{hw}; y_{hw})$$
Architecture: descriptor decoder

- Compute an $\mathbb{R}^{H_c \times W_c \times D}$ tensor
- Effectively, learn a $D$-dimensional descriptor for every 8x8 region
- Keep memory and run-time tractable
- Bicubic interpolation to image resolution and L2 normalization
Loss function: descriptor decoder

- Homography induced correspondence label:
  \[ s_{hwhw'} = \begin{cases} 
  1, & \text{if } ||\mathcal{H}p_h - p_{h'w'}|| \leq 8 \\
  0, & \text{otherwise} 
\end{cases} \]

- Descriptor loss:
  \[
  \mathcal{L}_d(D, D', S) = \frac{1}{(H_c W_c)^2} \sum_{h=1}^{H_c} \sum_{w=1}^{W_c} l_d(d_{hw}, d'_{h'w'}, s_{hwhw'}) 
  \]
  \[
  l_d(d, d'; s) = \lambda_d * s * \max(0, m_p - d^T d') + (1 - s) * \max(0, d^T d' - m_n) 
  \]

Centers of cells in left and right images
Descriptor cells in left and right images
Synthetic Pre-Training

- Hard to get a method-independent dataset with ground truth interest points
- Perhaps hard to even define notion of “ground truth” for interest points

- Render images with combinations of simple geometric shapes
- Various junctions, centers of ellipses, ends of line segments are “interest points”
- Apply homographic warps to train base keypoint detector

- Advantage: simple method with good performance
- Limitation: only a few specific types of interest points are covered.
Base Detector Evaluation

- Does well on synthetic dataset

<table>
<thead>
<tr>
<th>Metric</th>
<th>Noise</th>
<th>MagicPointL</th>
<th>MagicPointS</th>
<th>FAST</th>
<th>Harris</th>
<th>Shi</th>
</tr>
</thead>
<tbody>
<tr>
<td>mAP no noise</td>
<td>0.979</td>
<td>0.980</td>
<td>0.405</td>
<td>0.678</td>
<td>0.686</td>
<td></td>
</tr>
<tr>
<td>mAP noise</td>
<td>0.971</td>
<td>0.939</td>
<td>0.061</td>
<td>0.213</td>
<td>0.157</td>
<td></td>
</tr>
<tr>
<td>MLE no noise</td>
<td>0.860</td>
<td>0.922</td>
<td>1.656</td>
<td>1.245</td>
<td>1.188</td>
<td></td>
</tr>
<tr>
<td>MLE noise</td>
<td><strong>1.012</strong></td>
<td>1.078</td>
<td>1.766</td>
<td>1.409</td>
<td>1.383</td>
<td></td>
</tr>
</tbody>
</table>

- Especially robust performance with respect to noise
Base Detector Evaluation

- But does not do well on real data (repeatability metric)

<table>
<thead>
<tr>
<th></th>
<th>57 Illumination Scenes</th>
<th>59 Viewpoint Scenes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NMS=4</td>
<td>NMS=8</td>
</tr>
<tr>
<td>MagicPoint</td>
<td>.575</td>
<td>.507</td>
</tr>
<tr>
<td>FAST</td>
<td>.575</td>
<td>.472</td>
</tr>
<tr>
<td>Harris</td>
<td>.620</td>
<td>.533</td>
</tr>
<tr>
<td>Shi</td>
<td>.606</td>
<td>.511</td>
</tr>
<tr>
<td>Random</td>
<td>.101</td>
<td>.103</td>
</tr>
</tbody>
</table>

- Correct distance of 3 pixels
- Larger NMS threshold: discourages corners from clustering together
- Useful in applications like visual odometry
Homographic Adaptation

- Want to adapt better to real images (say, from MS-COCO dataset)
- But there is no ground truth
- Self-training strategy that uses detector itself to generate pseudo ground truth
- Warp images by random homography transformations
- Apply current version of detector
- Unwarp the images and aggregate responses

\[
\hat{F}(I; f_\theta) = \frac{1}{N_h} \sum_{i=1}^{N_h} \mathcal{H}_i^{-1} f_\theta(\mathcal{H}_i(I))
\]
Homographic Adaptation: Details

- Iterate multiple times to increase impact of self-training

- Choose warps to mimic camera transformations
**SuperPoint Evaluation**

- Self-training leads to improvement on real data (repeatability metric)

<table>
<thead>
<tr>
<th></th>
<th>57 Illumination Scenes NMS=4</th>
<th>57 Illumination Scenes NMS=8</th>
<th>59 Viewpoint Scenes NMS=4</th>
<th>59 Viewpoint Scenes NMS=8</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuperPoint</td>
<td>.652</td>
<td>.631</td>
<td>.503</td>
<td>.484</td>
</tr>
<tr>
<td>MagicPoint</td>
<td>.575</td>
<td>.507</td>
<td>.322</td>
<td>.260</td>
</tr>
<tr>
<td>FAST</td>
<td>.575</td>
<td>.472</td>
<td>.503</td>
<td>.404</td>
</tr>
<tr>
<td>Harris</td>
<td>.620</td>
<td>.533</td>
<td><strong>.556</strong></td>
<td>.461</td>
</tr>
<tr>
<td>Shi</td>
<td>.606</td>
<td>.511</td>
<td>.552</td>
<td>.453</td>
</tr>
<tr>
<td>Random</td>
<td>.101</td>
<td>.103</td>
<td>.100</td>
<td>.104</td>
</tr>
</tbody>
</table>

- SIFT better for localization and homography estimation at small thresholds

<table>
<thead>
<tr>
<th></th>
<th>Homography Estimation</th>
<th>Detector Metrics</th>
<th>Descriptor Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\epsilon = 1$</td>
<td>$\epsilon = 3$</td>
<td>$\epsilon = 5$</td>
</tr>
<tr>
<td>SuperPoint</td>
<td>.310</td>
<td><strong>.684</strong></td>
<td><strong>.829</strong></td>
</tr>
<tr>
<td>LIFT</td>
<td>.284</td>
<td>.598</td>
<td>.717</td>
</tr>
<tr>
<td>SIFT</td>
<td><strong>.424</strong></td>
<td>.676</td>
<td>.759</td>
</tr>
<tr>
<td>ORB</td>
<td>.150</td>
<td>.395</td>
<td>.538</td>
</tr>
</tbody>
</table>
SuperPoint Evaluation

- Outperforms LIFT, due to more diverse outdoor training data and self-training
- Denser (more repeatable) than SIFT, but poorer localization
- Does very well on mAP and matching score
- Cannot handle large rotations not seen in training data