Multiple-Kernel Local-Patch Descriptor

A. Mukundan    G. Tolias    O. Chum

Visual Recognition Group (VRG),
Czech Technical University in Prague

BMVC 2017
Local-patch descriptors

- Detect repeatable keypoints
- Normalize to square patches
- Extract local-patch descriptor

Tasks:
- Structure-from-Motion,
- Multi-View Stereo,
- Image-based localization,
- Image retrieval
Related work

Hand-crafted descriptors
• SIFT, SURF, Daisy, BRIEF, ORB, BRISK, ...
• Intuitive
• Tweakable
• No learning
• Low performance

Learned descriptors
• Dominated by CNN-based: DDesc, TFeat, DeepCompare, ...
• Lots of training data
• Long training time
• Behavior difficult to interpret
• High performance
Overview

• Hand-crafted design $\rightarrow$ kernel descriptor
• Inject supervision $\rightarrow$ data whitening from matching pairs of patches

• Intuitive design
• Understanding of learned patch similarity
• Performance boosted by supervision
A familiar descriptor: SIFT

Local patch divided into 4 x 4 spatial bins

[D. Lowe, IJCV '04]
SIFT (continued)

In each spatial bin:
- Orientation and magnitude of gradient per pixel
- Quantize orientation angles into $N_\theta$ bins
SIFT (continued)

In each spatial bin:
- Orientation and magnitude of gradient per pixel
- Quantize orientation angles into $N_\theta$ bins
SIFT (continued)

Descriptor: concatenation of histograms
SIFT (continued)

Descriptor: concatenation of histograms

\[ 4 \times 4 \times 8 = 128 \text{ dimensional descriptor} \]
SIFT – contribution of a single pixel
SIFT – contribution of a single pixel
SIFT – contribution of a single pixel
SIFT – contribution of a single pixel
Patch similarity using SIFT

\[
\begin{align*}
\times & \quad = \\
& \quad \left( \begin{array}{c}
+ \\
+ \\
+ \\
\vdots
\end{array} \right) \times \\
& \quad \left( \begin{array}{c}
+ \\
+ \\
+ \\
\vdots
\end{array} \right)
\end{align*}
\]
Patch similarity using SIFT

\[ X \times = ( + + + \ldots ) \times ( + + + \ldots ) \]
Patch similarity using SIFT
Patch similarity using SIFT

\[
\langle \psi(\mathcal{P}), \psi(\mathcal{Q}) \rangle = \langle \sum \psi(p), \sum \psi(q) \rangle = \sum_{p \in \mathcal{P}} \sum_{q \in \mathcal{Q}} \psi(p)^\top \psi(q) = \sum_{p \in \mathcal{P}} \sum_{q \in \mathcal{Q}} k(p, q)
\]

SIFT similarity between two patches \( \mathcal{P} \) and \( \mathcal{Q} \)
Patch similarity using SIFT

Compares all pixel pairs based on attributes $x, y, \theta$
Patch similarity using SIFT

Compares all pixel pairs based on attributes $x, y, \theta$

$$\kappa(\Delta_x) \cdot \kappa(\Delta_y) \cdot \kappa(\Delta_\theta)$$

$$\kappa(\Delta_x) = \begin{cases} 
1 & , \text{same bin} \\
0 & , \text{otherwise}
\end{cases}$$
Extension to continuous similarity

- Map 1D attribute to vector embedding
- Scalar product approximates von Mises kernel
- Dimensionality of embedding controls the approximation

[Bursuc et al. ICMR ’15]
Pixel embedding

$$\psi(x, y, \theta)$$

SIFT
Pixel embedding

\[ \psi(x, y, \theta) \]

Kernel descriptor
Visualizing patch similarity

- Pixel to pixel similarity
  \[ k(p, q) = \langle \psi(p), \psi(q) \rangle \approx k_x \cdot k_y \cdot k_\theta \]

- **Patchmap**: visualize contribution over the whole patch
Visualizing patch similarity

• Pixel to pixel similarity
  \[ k(p, q) = \langle \psi(p), \psi(q) \rangle \approx k_x \cdot k_y \cdot k_\theta \]

• **Patchmap**: visualize contribution over the whole patch
Visualizing patch similarity

- **Pixel to pixel similarity**
  \[ k(p, q) = \langle \psi(p), \psi(q) \rangle \approx k_x \cdot k_y \cdot k_\theta \]

- **Patchmap**: visualize contribution over the whole patch

10 iso-contours shown
- **Maximum similarity in red**
- **Minimum similarity in blue**
Visualizing patch similarity

- Pixel to pixel similarity
  \[ k(p, q) = \langle \psi(p), \psi(q) \rangle \]
  \[ \approx k_x \cdot k_y \cdot k_\theta \]

- **Patchmap**: visualize contribution over the whole patch

10 iso-contours shown
**Maximum similarity in red**
**Minimum similarity in blue**
Shift invariant similarity

Stationary kernel: depends on the difference, not the absolute value

Pixel \( p \)

Patch \( Q \)
Shift invariant similarity

Stationary kernel: depends on the difference, not the absolute value
Patchmap for SIFT

Pixel $p$

Patch $Q$
Patchmap for SIFT
Patchmap for SIFT

Pixel \( p \)

Patch \( Q \)
Patchmap for SIFT

Pixel $p$

Patch $Q$
Patchmap for SIFT

Pixel $p$

Patch $Q$
Patchmap for SIFT

Pixel $p$

Patch $Q$
Patch parametrizations

\[ \theta = \pi/4 \]

cartes

\[ \rho \]

polar

\[ \phi \]
Patch parametrizations

\[ \theta = \pi/4 \]

\[ \tilde{\theta} = \theta - \phi = -\pi/2 \]

cartes

polar
Patch parametrizations

*cartes*

*polar*
Patch parametrizations

cartes

dominant orientation misalignment

polar
Patch parametrizations

keypoint center misalignment

cartes

dominant orientation misalignment

polar
Multiple parametrizations

Polar
- Known to perform well [Bursuc ICMR ’15]
- Tolerance to orientation mistakes
- Discontinuity around the center

Cartes
- Stable around the center
- Tolerance to location mistakes

We preserve both behaviors
- Simple descriptor concatenation
Injecting supervision

Linear discriminant projections

• Supervised descriptor whitening
• Very fast to learn!
• Works with few training examples
• Learns optimal mixing coefficient in case of concatenated vectors

[Michalajczyk & Matas. ICCV ’07]
Shift invariant similarity

Stationary kernel: depends on difference of angles, not their absolute value
Shift invariant similarity

Stationary kernel: depends on difference of angles, not their absolute value
Shift invariant similarity

Stationary kernel: depends on difference of angles, not their absolute value
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel \( p \)

Patch \( Q \)
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel \( p \)

Patch \( Q \)
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel \( p \)

Patch \( Q \)
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!

Pixel $p$

Patch $Q$
Similarity learned by supervised whitening

Similarity function is not shift invariant anymore!
Similarity learned by supervised whitening

Similarity shape gets aligned with gradient orientation
→ Potentially handling over-counting along edges

Patch $\mathcal{P}$

Patch $\mathcal{Q}$

Both: overlaid
Datasets

Phototourism (PT) – ‘07

[Winder & Brown, CVPR ‘07]

HPatches (HP) – ‘17

[Balntas et al, CVPR ‘17]
Results – PhotoToursim

- Verification task: given 2 patches determine if they depict the same 3D point
- Metric: False positive rate at 95% recall (FPR95)
- Detectors: DoG
- **PCW**: our descriptor (polar + cartes + whitening)

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>Dimensions</th>
<th>Mean FPR95</th>
<th>CNN</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC-S2S</td>
<td>512</td>
<td>9.67</td>
<td>×</td>
<td>hours</td>
</tr>
<tr>
<td>DDESCC</td>
<td>128</td>
<td>9.85</td>
<td>×</td>
<td>hours</td>
</tr>
<tr>
<td>Matchnet</td>
<td>4096</td>
<td>7.75</td>
<td>×</td>
<td>hours</td>
</tr>
<tr>
<td>TF-M</td>
<td>128</td>
<td>6.47</td>
<td>×</td>
<td>hours</td>
</tr>
<tr>
<td><strong>PCW∗</strong></td>
<td>128</td>
<td><strong>5.98</strong></td>
<td></td>
<td>seconds</td>
</tr>
</tbody>
</table>
Results – HPatches

- Tasks: verification, matching, retrieval
- Metric: mAP
- Detectors: DoG, Hessian-Hessian, Harris-Laplace
- **PCW**: our descriptor (polar + cartes + whitening)

<table>
<thead>
<tr>
<th>Verification</th>
<th>Matching</th>
<th>Retrieval</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF–R</td>
<td>81.92</td>
<td>+TF–R 40.23</td>
</tr>
<tr>
<td>+TF–M</td>
<td>82.69</td>
<td>+TF–M 34.29 +SIFT 40.36</td>
</tr>
<tr>
<td><strong>PCW</strong></td>
<td>82.94</td>
<td>+TF–R 34.37 +RSIFT 43.84</td>
</tr>
<tr>
<td>+DC–S2S</td>
<td>83.03</td>
<td>+DDESC 35.44 +DDESC 44.55</td>
</tr>
<tr>
<td>+TF–R</td>
<td>83.24</td>
<td>+RSIFT 36.77 <strong>PCW</strong> 48.26</td>
</tr>
<tr>
<td><strong>PCW</strong></td>
<td>88.64</td>
<td><strong>PCW</strong> 43.81 <strong>PCW</strong> 61.21</td>
</tr>
</tbody>
</table>

Others: trained on part of Phototourism, whitening learned on part of HPatches

**PCW**: Whitening learned on part of Phototourism

**PCW**: Whitening learned on part of HPatches
Impact of the whitening

<table>
<thead>
<tr>
<th>Method</th>
<th>Verification</th>
<th>Matching</th>
<th>Retrieval</th>
</tr>
</thead>
<tbody>
<tr>
<td>polar [9]</td>
<td>80.77</td>
<td>32.51</td>
<td>48.04</td>
</tr>
<tr>
<td>cartes</td>
<td>70.67</td>
<td>15.79</td>
<td>30.73</td>
</tr>
<tr>
<td>polar + cartes</td>
<td>77.97</td>
<td>29.34</td>
<td>44.23</td>
</tr>
<tr>
<td>polar + cartes + LW</td>
<td>88.64</td>
<td>43.81</td>
<td>61.21</td>
</tr>
</tbody>
</table>

LW: Learned Whitening
Conclusions

- Hand-crafted descriptor
  - Intuitive design
  - Easy understanding

- Supervised whitening
  - Requires few training pairs
  - Extremely fast to learn
  - Significant performance boost
  - Understanding of the learned similarity

- Outperforms CNN-based descriptors