SIFT - The Scale Invariant Feature Transform


Presented by Ofir Pele.

Based upon slides from:
- Sebastian Thrun and Jana Košecká
- Neeraj Kumar
Correspondence

- Fundamental to many of the core vision problems
  - Recognition
  - Motion tracking
  - Multiview geometry

- Local features are the key

Local Features: Detectors & Descriptors

Detected Interest Points/Regions

Descriptors

<0 12 31 0 0 23 ...

<5 0 0 11 37 15 ...

<14 21 10 0 3 22 ...>
Ideal Interest Points/Regions

- Lots of them
- Repeatable
- Representative orientation/scale
- Fast to extract and match
## SIFT Overview

**Detector**

1. Find Scale-Space Extrema
2. Keypoint Localization & Filtering
   - Improve keypoints and throw out bad ones

**Descriptor**

3. Orientation Assignment
   - Remove effects of rotation and scale
4. Create descriptor
   - Using histograms of orientations
SIFT Overview

Detector

1. **Find Scale-Space Extrema**

2. Keypoint Localization & Filtering
   – Improve keypoints and throw out bad ones

3. Orientation Assignment
   – Remove effects of rotation and scale

4. Create descriptor
   – Using histograms of orientations

Descriptor
Scale Space

- Need to find ‘characteristic scale’ for feature
- Scale-Space: Continuous function of scale $\sigma$
  - Only reasonable kernel is Gaussian:

$$L(x, y, \sigma_D) = G(x, y, \sigma_D) * I(x, y)$$

[Koenderink 1984, Lindeberg 1994]
Scale Selection

- Experimentally, Maxima of Laplacian-of-Gaussian gives best notion of scale:

- Thus use Laplacian-of-Gaussian (LoG) operator:

\[ \sigma^2 \nabla^2 G \]

Mikolajczyk 2002
Approximate LoG

- LoG is expensive, so we approximate it with Difference-of-Gaussians (DoG):

\[ D(\sigma) \equiv (G(k\sigma) - G(\sigma)) \ast I \]
DoG Efficiency

- The smoothed images need to be computed in any case for feature description.
- We need only to subtract two images.
DoB Filter (`Difference of Boxes')

Even faster approximation is using box filters (by integral image)

Fig. 1. Left to right: the (discretised and cropped) Gaussian second order partial derivatives in $y$-direction and $xy$-direction, and our approximations thereof using box filters. The grey regions are equal to zero.
Integral Image Computation-
code example
Integral Image Usage

Using the integral image representation one can compute the value of any rectangular sum in constant time.

Example: Rectangle D

\[ ii(4) + ii(1) - ii(2) - ii(3) \]
Scale-Space Construction

- First construct scale-space:

\[ G(\sigma) \ast I \]
\[ G(k\sigma) \ast I \]
\[ G(2\sigma) \ast I \]
\[ G(2k\sigma) \ast I \]
\[ G(2k^2\sigma) \ast I \]
Difference-of-Gaussians

Now take differences:
Scale-Space Extrema

- Choose all extrema within $3\times3\times3$ neighborhood.
- Low cost – only several usually checked

\[ D(k^2\sigma) \]
\[ D(k\sigma) \]
\[ D(\sigma) \]
SIFT Overview

1. Find Scale-Space Extrema

2. **Keypoint Localization & Filtering**
   - Improve keypoints and throw out bad ones

3. Orientation Assignment
   - Remove effects of rotation and scale

4. Create descriptor
   - Using histograms of orientations

**Descriptor**
Keypoint Localization & Filtering

- Now we have much less points than pixels.
- However, still lots of points (~1000s)...
  - With only pixel-accuracy at best
    • At higher scales, this corresponds to several pixels in base image
  - And this includes many bad points

Brown & Lowe 2002
Keypoint Localization

- The problem:

![Diagram showing keypoint localization with sampled points and true extrema.](image)
Keypoint Localization

■ The solution:
  – Take Taylor series expansion:

\[
D(\tilde{x}) = D + \frac{\partial D^T}{\partial \tilde{x}} \tilde{x} + \frac{1}{2} \tilde{x}^T \frac{\partial^2 D^T}{\partial \tilde{x}^2} \tilde{x}
\]

  – Minimize to get true location of extrema:

\[
\hat{x} = - \frac{\partial^2 D}{\partial \tilde{x}^2}^{-1} \frac{\partial D}{\partial \tilde{x}}
\]

Brown & Lowe 2002
Keypoints

(a) 233x189 image
(b) 832 DOG extrema
Keypoint Filtering - Low Contrast

- Reject points with bad contrast

\[ D(\hat{x}) \] is smaller than 0.03 (image values in \([0,1]\))
Keypoint Filtering - Edges

- Reject points with strong edge response in one direction only

Point detection

Point can move along edge

Point constrained

Point detection
Keypoint Filtering

(c) 729 left after peak value threshold (from 832)
(d) 536 left after testing ratio of principle curvatures
## SIFT Overview

<table>
<thead>
<tr>
<th>Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Find Scale-Space Extrema</td>
</tr>
<tr>
<td>2. Keypoint Localization &amp; Filtering</td>
</tr>
<tr>
<td>– Improve keypoints and throw out bad ones</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>3. Orientation Assignment</td>
</tr>
<tr>
<td>– Remove effects of rotation and scale</td>
</tr>
<tr>
<td>4. Create descriptor</td>
</tr>
<tr>
<td>– Using histograms of orientations</td>
</tr>
</tbody>
</table>
Ideal Descriptors

- Robust to:
  - Affine transformation
  - Lighting
  - Noise
- Distinctive
- Fast to match
  - Not too large
  - Usually L1 or L2 matching
# SIFT Overview

<table>
<thead>
<tr>
<th>Detector</th>
<th>1. Find Scale-Space Extrema</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Keypoint Localization &amp; Filtering</td>
</tr>
<tr>
<td></td>
<td>– Improve keypoints and throw out bad ones</td>
</tr>
</tbody>
</table>

### 3. Orientation Assignment

|          | Remove effects of rotation and scale |

<table>
<thead>
<tr>
<th>4. Create descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>– Using histograms of orientations</td>
</tr>
</tbody>
</table>

**Descriptor**
Orientation Assignment

- Now we have set of good points
- Choose a region around each point
  - Remove effects of scale and rotation
Orientation Assignment

- Use scale of point to choose correct image:

\[ L(x, y) = G(x, y, \sigma) * I(x, y) \]

- Compute gradient magnitude and orientation using finite differences:

\[
m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}
\]

\[
\theta(x, y) = \tan^{-1}\left( \frac{(L(x, y+1) - L(x, y-1))}{(L(x+1, y) - L(x-1, y))} \right)
\]
Orientation Assignment

- Create gradient histogram (36 bins)
  - Weighted by magnitude and Gaussian window (\(\sigma\) is 1.5 times that of the scale of a keypoint)
Orientation Assignment

- Any peak within 80% of the highest peak is used to create a keypoint with that orientation.
- ~15% assigned multiplied orientations, but contribute significantly to the stability.
- Finally, a parabola is fit to the 3 histogram values closest to each peak to interpolate the peak position for better accuracy.
<table>
<thead>
<tr>
<th>SIFT Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Detector</strong></td>
</tr>
<tr>
<td>1. Find Scale-Space Extrema</td>
</tr>
<tr>
<td>2. Keypoint Localization &amp; Filtering</td>
</tr>
<tr>
<td>– Improve keypoints and throw out bad ones</td>
</tr>
<tr>
<td>3. Orientation Assignment</td>
</tr>
<tr>
<td>– Remove effects of rotation and scale</td>
</tr>
<tr>
<td><strong>Create descriptor</strong></td>
</tr>
<tr>
<td>– Using histograms of orientations</td>
</tr>
</tbody>
</table>

**Descriptor**
SIFT Descriptor

- Each point so far has $x$, $y$, $\sigma$, $m$, $\theta$
- Now we need a descriptor for the region
  - Could sample intensities around point, but...
    - Sensitive to lighting changes
    - Sensitive to slight errors in $x$, $y$, $\theta$
- Look to biological vision
  - Neurons respond to gradients at certain frequency and orientation
    - But location of gradient can shift slightly!

Edelman et al. 1997
SIFT Descriptor

- 4x4 Gradient window
- Histogram of 4x4 samples per window in 8 directions
- Gaussian weighting around center (\( \sigma \) is 0.5 times that of the scale of a keypoint)
- 4x4x8 = 128 dimensional feature vector
SIFT Descriptor – Lighting changes

- Gains do not affect gradients
- Normalization to unit length removes contrast
- Saturation affects magnitudes much more than orientation
- Threshold gradient magnitudes to 0.2 and renormalize
Performance

- Very robust
  - 80% Repeatability at:
    - 10% image noise
    - 45° viewing angle
    - 1k-100k keypoints in database

- Best descriptor in [Mikolajczyk & Schmid 2005]’s extensive survey

- 28000+ citations on Google Scholar
Typical Usage

For set of database images:
1. Compute SIFT features
2. Save descriptors to database

For query image:
1. Compute SIFT features
2. For each descriptor:
   • Find a match
3. Verify matches
   • Geometry
   • Hough transform
Matching Descriptors

- Threshold on Distance – bad performance
- Nearest Neighbor – better
- Ratio Test – best performance
Matching Descriptors - Distance

- $L_2$ norm – used by Lowe
- $\text{SIFT}_{\text{DIST}}$: linear time EMD algorithm that adds robustness to orientation shifts
  Pele and Werman, ECCV 2008
Ratio Test

- Need to be careful with the definition of next closest:

Image 2

- **False** 2nd best match
- Best Match

Image 1

- **True** 2nd best match
Fast Nearest-Neighbor Matching to Feature Database

- Hypotheses are generated by **approximate nearest neighbor** matching of each feature to vectors in the database
  - SIFT use best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point

- **Result:** Can give speedup by factor of 1000 while finding nearest neighbor (of interest) 95% of the time
3D Object Recognition

- Only 3 keys are needed for recognition, so extra keys provide robustness
Recognition under occlusion
Test of illumination Robustness

- Same image under differing illumination

273 keys verified in final match
Location recognition
Image Registration Results

[Brown & Lowe 2003]
Cases where SIFT didn’t work
Cases where SIFT didn’t work

- Same **object** under differing illumination
Large illumination change

- Same object under differing illumination
- 43 keypoints in left image and the corresponding closest keypoints on the right (1 for each)
Large illumination change

- Same **object** under differing illumination
- 43 keypoints in left image and the corresponding closest keypoints on the right (5 for each)
Non rigid deformations

- 11 keypoints in left image and the corresponding closest keypoints on the right (1 for each)
Non rigid deformations

11 keypoints in left image and the corresponding closest keypoints on the right (5 for each)
Conclusion: SIFT

- Built on strong foundations
  - First principles (LoG and DoG)
  - Biological vision (Descriptor)
  - Empirical results
- Many heuristic optimizations
  - Rejection of bad points
  - Sub-pixel level fitting
  - Thresholds carefully chosen
Conclusion: SIFT

- In wide use both in academia and industry
- Many available implementations:
  - Binaries available at Lowe’s website
  - C/C++ open source by A. Vedaldi (UCLA)
  - C# library by S. Nowozin (Tu-Berlin ➔ Microsoft)
- Protected by a patent
Conclusion: SIFT

- Empirically found (Mikolajczyk & Schmid 2005) to show very good performance, robust to *image rotation, scale, intensity change*, and to moderate *affine* transformations

Scale = 2.5
Rotation = 45°
A note regarding invariance/robustness

- There is a tradeoff between invariance and distinctiveness.
- For some tasks it is better not to be invariant
- 11 color names - J. van de Weijer, C. Schmid, Applying Color Names to Image Description. ICIP 2007
SIFT extensions
Color


- Hue and Opponent histograms - J. van de Weijer, C. Schmid. Coloring Local Feature Extraction. ECCV 2006

- 11 color names - J. van de Weijer, C. Schmid, Applying Color Names to Image Description. ICIP 2007
PCA-SIFT

- Only change step 4 (creation of descriptor)
- Pre-compute an eigen-space for local gradient patches of size 41x41
- $2 \times 39 \times 39 = 3042$ elements
- Only keep 20 components
- A more compact descriptor
- In K. Mikolajczyk, C. Schmid 2005 PCA-SIFT tested inferior to original SIFT
Speed Improvements

- SURF - Bay et al. 2006
- Approx SIFT - Grabner et al. 2006
- GPU implementation - Sudipta N. Sinha et al. 2006

![Graph showing speed improvements for different feature counts and image sizes.](image-url)
GLOH (Gradient location-orientation histogram)

17 location bins
16 orientation bins
Analyze the $17 \times 16 = 272$-d eigen-space, keep 128 components