Read Szeliski 7.1.2 and 7.1.3

Local Image Features

Computer Vision

James Hays

Acknowledgment: Many slides from Derek Hoiem and Grauman&Leibe 2008 AAAI Tutorial
“Flashed Face Distortion”
2nd Place in the 8th Annual
Best Illusion of the Year
Contest, VSS 2012
Keep your eyes on the cross
Project 2

The top 100 most confident local feature matches from a baseline implementation of project 2. In this case, 93 were correct (highlighted in green) and 7 were incorrect (highlighted in red).

Project 2: Local Feature Matching
This section: correspondence and alignment

- Correspondence: matching points, patches, edges, or regions across images
Overview of Keypoint Matching

1. Find a set of distinctive keypoints

2. Compute a local descriptor from the region around each keypoint

3. Match local descriptors

\[ d(f_A, f_B) < T \]
Review: Harris corner detector

• Define distinctiveness by local auto-correlation.

• Approximate local auto-correlation by second moment matrix

• Quantify distinctiveness (or cornerness) as function of the eigenvalues of the second moment matrix.

• But we don’t actually need to compute the eigenvalues. Instead, we use the determinant and trace of the second moment matrix.

\[
E(u, v) = \lambda_{max}^{-1/2} - \lambda_{min}^{-1/2}
\]
If you’re not comfortable with Eigenvalues and Eigenvectors, Gilbert Strang’s linear algebra lectures are linked from the course homepage.

Lecture 21: Eigenvalues and eigenvectors

Linear Algebra Lecture 21
Eigenvalues - Eigenvectors
\[ \det [A - \lambda I] = 0 \]
\[ \text{Trace} = \lambda_1 + \lambda_2 + \ldots + \lambda_n \]
Harris Detector [Harris88]

- Second moment matrix

\[
\mu(\sigma_I, \sigma_D) = g(\sigma_I) \ast \begin{bmatrix}
I_x^2(\sigma_D) & I_xI_y(\sigma_D) \\
I_xI_y(\sigma_D) & I_y^2(\sigma_D)
\end{bmatrix}
\]

1. Image derivatives (optionally, blur first)

\[
\det M = \lambda_1\lambda_2
\]

trace \( M = \lambda_1 + \lambda_2 \)

2. Square of derivatives

3. Gaussian filter \( g(\sigma_I) \)

4. Cornerness function – both eigenvalues are strong

\[
har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))^2] = g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2
\]

5. Non-maxima suppression
Harris Detector: Steps
Harris Detector: Steps

Compute corner response $R$
Harris Detector: Steps

Find points with large corner response: $R > \text{threshold}$
Harris Detector: Steps

Take only the points of local maxima of $R$
Harris Detector: Steps
Invariance and covariance

- We want corner locations to be *invariant* to photometric transformations and *covariant* to geometric transformations
  - **Invariance**: image is transformed and corner locations do not change
  - **Covariance**: if we have two transformed versions of the same image, features should be detected in corresponding locations
Affine intensity change

- Only derivatives are used $\Rightarrow$ invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a I$

*Partially invariant* to affine intensity change
Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation
Corner location is not covariant to scaling!

All points will be classified as edges.
Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation
So far: can localize in x-y, but not scale
Automatic Scale Selection

How to find corresponding patch sizes?

$$f(I_{i_1...i_m}(x, \sigma)) = f(I'_{i_1...i_m}(x', \sigma'))$$
Automatic Scale Selection

- Function responses for increasing scale (scale signature)
Automatic Scale Selection

• Function responses for increasing scale (scale signature)

\[ f(I_{i,j} (x, \sigma)) \]

\[ f(I_{i,j} (x', \sigma)) \]

K. Grauman, B. Leibe
Automatic Scale Selection

- Function responses for increasing scale (scale signature)
Automatic Scale Selection

- Function responses for increasing scale (scale signature)
Automatic Scale Selection

- Function responses for increasing scale (scale signature)
Automatic Scale Selection

• Function responses for increasing scale (scale signature)
Orientation Normalization

- Compute orientation histogram
- Select dominant orientation
- Normalize: rotate to fixed orientation

[Lowe, SIFT, 1999]
Maximally Stable Extremal Regions [Matas ‘02]

- Based on Watershed segmentation algorithm
- Select regions that stay stable over a large parameter range
Example Results: MSER
Comparison

Harris

Hessian

MSER

LoG
Local features: main components

1) Detection: Identify the interest points

2) Description: Extract vector feature descriptor surrounding $\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$ each interest point.

3) Matching: Determine correspondence between descriptors in two views

$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$
Image representations

• Templates
  – Intensity, color, gradients, etc.
  – Keeps spatial layout

• Histograms
  – Distribution of intensity, color, texture, SIFT descriptors, etc.
  – Discards spatial layout
Image Representations: Histograms

Histogram: Probability or count of data in each bin

• Joint histogram
  – Requires lots of data
  – Loss of resolution to avoid empty bins

Marginal histogram
  • Requires independent features
  • More data/bin than joint histogram

Images from Dave Kauchak
Image Representations: Histograms

Clustering

Use the same cluster centers for all images

Images from Dave Kauchak
Computing histogram distance

\[ \text{histint}(h_i, h_j) = 1 - \sum_{m=1}^{K} \min(h_i(m), h_j(m)) \]

Histogram intersection (assuming normalized histograms)

\[ \chi^2(h_i, h_j) = \frac{1}{2} \sum_{m=1}^{K} \frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)} \]

Chi-squared Histogram matching distance

Cars found by color histogram matching using chi-squared
Histograms: Implementation issues

- **Quantization**
  - Grids: fast but applicable only with few dimensions
  - Clustering: slower but can quantize data in higher dimensions

- **Matching**
  - Histogram intersection or Euclidean may be faster
  - Chi-squared often works better
  - Earth mover’s distance is good for when nearby bins represent similar values
What kind of things do we compute histograms of?

- **Color**
  
  ![L*a*b* color space](image1.png)
  ![HSV color space](image2.png)

- **Texture (filter banks or HOG over regions)**
What kind of things do we compute histograms of?

- Histograms of oriented gradients

SIFT – Lowe IJCV 2004
SIFT vector formation

• 4x4 array of gradient orientation histogram weighted by magnitude
• 8 orientations x 4x4 array = 128 dimensions
• Motivation: some sensitivity to spatial layout, but not too much.
Ensure smoothness

• Gaussian weight
• Interpolation
  – a given gradient contributes to 8 bins:
    4 in space times 2 in orientation
Reduce effect of illumination

• 128-dim vector normalized to 1
• Optionally, threshold gradient magnitudes to avoid excessive influence of high gradients
  – after normalization, clamp gradients >0.2
  – renormalize
Local Descriptors: Shape Context

Count the number of points inside each bin, e.g.:

- Count = 4
- Count = 10

Log-polar binning: more precision for nearby points, more flexibility for farther points.

Belongie & Malik, ICCV 2001
Shape Context Descriptor
Self-similarity Descriptor

Figure 1. These images of the same object (a heart) do NOT share common image properties (colors, textures, edges), but DO share a similar geometric layout of local internal self-similarities.

Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007
Self-similarity Descriptor

Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007
Self-similarity Descriptor

Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007
Learning Local Image Descriptors, Winder and Brown, CVPR 2007

Image Patch → Smooth $G(x, \sigma)$ → T-Block Filter → S-Block Pooling → N-Block Normalize → Descriptor

- Image Patch: 64x64 Pixels
- Smooth $G(x, \sigma)$: ~64x64 vectors of dimension k
- T-Block Filter: N histograms of dimension k

S1: SIFT grid with bilinear weights
S2: GLOH polar grid with bilinear radial and angular weights
S3: 3x3 grid with Gaussian weights
S4: 17 polar samples with Gaussian weights
We obtained a mixed training set consisting of tourist photographs of the Trevi Fountain and of Yosemite Valley (920 images), and a test set consisting of images of Notre Dame (500 images). We extracted interest points and matched them between all of the images within a set using the SIFT detector and descriptor [9]. We culled candidate matches using a symmetry criterion and used RANSAC [5] to estimate initial fundamental matrices between image pairs. This stage was followed by bundle adjustment to reconstruct 3D points and to obtain accurate camera matrices for each source image. A similar technique has been described by [17].

Figure 5. Selected ROC curves for the trained descriptors with four dimensional T-blocks ($k = 4$). Those that perform better than SIFT all make use of the S2 log-polar summation stage. See Table 4 for details.
Local Descriptors

• Most features can be thought of as templates, histograms (counts), or combinations
• The ideal descriptor should be
  – Robust
  – Distinctive
  – Compact
  – Efficient
• Most available descriptors focus on edge/gradient information
  – Capture texture information
  – Color rarely used

K. Grauman, B. Leibe
Local features: main components

1) Detection: Identify the interest points

2) Description: Extract vector feature descriptor surrounding each interest point.

3) Matching: Determine correspondence between descriptors in two views

Kristen Grauman
Matching

• Simplest approach: Pick the nearest neighbor. Threshold on absolute distance
• Problem: Lots of self similarity in many photos
Distance: 0.34, 0.30, 0.40
Distance: 0.61
Distance: 1.22
Nearest Neighbor Distance Ratio

\[ \frac{NN_1}{NN_2} \]

where \( NN_1 \) is the distance to the first nearest neighbor and \( NN_2 \) is the distance to the second nearest neighbor.

• Sorting by this ratio (into ascending order) puts matches in order of confidence (in descending order of confidence).
Matching Local Features

- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2\textsuperscript{nd} nearest descriptor

\[ \text{PDF for correct matches} \quad \text{PDF for incorrect matches} \]

Lowe IJCV 2004
6.4 Matching to large databases

An important remaining issue for measuring the distinctiveness of features is how the reliability of matching varies as a function of the number of features in the database being matched. Most of the examples in this paper are generated using a database of 32 images with about 40,000 keypoints. Figure 10 shows how the matching reliability varies as a func-

Lowe IJCV 2004
SIFT Repeatability

Lowe IJCV 2004
SIFT Repeatability

Lowe IJCV 2004
Choosing a detector

• What do you want it for?
  – Precise localization in x-y: Harris
  – Good localization in scale: Difference of Gaussian
  – Flexible region shape: MSER

• Best choice often application dependent
  – Harris-/Hessian-Laplace/DoG work well for many natural categories
  – MSER works well for buildings and printed things

• Why choose?
  – Get more points with more detectors

• There have been extensive evaluations/comparisons
  – [Mikolajczyk et al., IJCV’05, PAMI’05]
  – All detectors/descriptors shown here work well
Comparison of Keypoint Detectors

Table 7.1. Overview of feature detectors.

<table>
<thead>
<tr>
<th>Feature Detector</th>
<th>Corner</th>
<th>Blob</th>
<th>Region</th>
<th>Rotation invariant</th>
<th>Scale invariant</th>
<th>Affine invariant</th>
<th>Repeatability</th>
<th>Localization accuracy</th>
<th>Robustness</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harris</td>
<td>√</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td></td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
</tr>
<tr>
<td>Hessian</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>SUSAN</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+++</td>
</tr>
<tr>
<td>Harris-Laplace</td>
<td>(√)</td>
<td>(√)</td>
<td></td>
<td>√</td>
<td>√</td>
<td></td>
<td>+++</td>
<td>+++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Hessian-Laplace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>++</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>DoG</td>
<td>(√)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>SURF</td>
<td>(√)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+++</td>
</tr>
<tr>
<td>Harris-Affine</td>
<td>(√)</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Hessian-Affine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
</tr>
<tr>
<td>Salient Regions</td>
<td>(√)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(√)</td>
<td>++</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Edge-based</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MSER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
<td>+++</td>
</tr>
<tr>
<td>Intensity-based</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>Superpixels</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(√)</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>
Choosing a descriptor

- Again, need not stick to one

- For object instance recognition or stitching, SIFT or variant is a good choice

- Learning-based methods are taking over this space, although not as quickly as one might expect.
Things to remember

• Keypoint detection: repeatable and distinctive
  – Corners, blobs, stable regions
  – Harris, DoG

• Descriptors: robust and selective
  – spatial histograms of orientation
  – SIFT