Overview

• Keypoint matching
• Hessian detector
• Blob detection
• Feature descriptors
• Fitting
• RANSAC
  – Based on slides by S. Lazebnik and D. Hoiem
Overview of Keypoint Matching

1. Find a set of distinctive keypoints
2. Define a region around each keypoint
3. Extract and normalize the region content
4. Compute a local descriptor from the normalized region
5. Match local descriptors

\[ d(f_A, f_B) < T \]

K. Grauman, B. Leibe
Goals for Keypoints

Detect points that are *repeatable* and *distinctive*
Key trade-offs

Detection
- More Repeatable
  - Robust detection
  - Precise localization
- More Points
  - Robust to occlusion
  - Works with less texture

Description
- More Distinctive
  - Minimize wrong matches
- More Flexible
  - Robust to expected variations
  - Maximize correct matches
Hessian Detector [Beaudet78]

• Hessian determinant

\[ Hessian (I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix} \]

Intuition: Search for strong curvature in two orthogonal directions

K. Grauman, B. Leibe
Hessian Detector [Beaudet78]

- Hessian determinant

\[
Hessian (x, \sigma) = \begin{bmatrix}
I_{xx}(x, \sigma) & I_{xy}(x, \sigma) \\
I_{xy}(x, \sigma) & I_{yy}(x, \sigma)
\end{bmatrix}
\]

\[
det M = \lambda_1 \lambda_2
\]

\[
\text{trace } M = \lambda_1 + \lambda_2
\]

Find maxima of determinant

\[
det(Hessian(x)) = I_{xx}(x)I_{yy}(x) - I_{xy}^2(x)
\]

In Matlab:

\[
I_{xx} \ast I_{yy} - (I_{xy})^2
\]
Effect: Responses mainly on corners and strongly textured areas.
Hessian Detector – Responses [Beaudet78]
So far: can localize in x-y, but not scale
Automatic Scale Selection

\[ f(I_{i_1...i_m}(x, \sigma)) = f(I_{i_1...i_m}(x', \sigma')) \]

How to find corresponding patch sizes?
Automatic Scale Selection

- Function responses for increasing scale (scale signature)

\[
f(I_{i_1...i_m}(x, \sigma))
\]

K. Grauman, B. Leibe
Automatic Scale Selection

- Function responses for increasing scale (scale signature)

\[ f(I_{i_1...i_m}(x, \sigma)) \]

K. Grauman, B. Leibe
Automatic Scale Selection

- Function responses for increasing scale (scale signature)
Blob detection
Feature detection with scale selection

- We want to extract features with characteristic scale that is *covariant* with the image transformation.
Blob detection: Basic idea

- To detect blobs, convolve the image with a “blob filter” at multiple scales and look for extrema of filter response in the resulting scale space.
Blob detection: Basic idea

- Find maxima and minima of blob filter response in space and scale

Source: N. Snavely
Blob filter

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

\[ \nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \]
Recall: Edge detection

\[ f \]

\[ \frac{d}{dx} \]

\[ g \]

\[ f \ast \frac{d}{dx} g \]

Edge = maximum of derivative

Source: S. Seitz
Edge detection, Take 2

\[ f \]

\[ \frac{d^2}{dx^2} g \]

\[ f \ast \frac{d^2}{dx^2} g \]

Edge = zero crossing of second derivative

Source: S. Seitz
From edges to blobs

- **Edge** = ripple
- **Blob** = superposition of two ripples

**Spatial selection**: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is “matched” to the scale of the blob.
Scale selection

- We want to find the characteristic scale of the blob by convolving it with Laplacians at several scales and looking for the maximum response.
- However, Laplacian response decays as scale increases:

![Graphs showing unnormalized Laplacian response with increasing scale parameter σ.]
Scale normalization

• The response of a derivative of Gaussian filter to a perfect step edge decreases as $\sigma$ increases

\[ \frac{1}{\sigma \sqrt{2\pi}} \]
Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as $\sigma$ increases.
- To keep response the same (scale-invariant), must multiply Gaussian derivative by $\sigma$.
- Laplacian is the second Gaussian derivative, so it must be multiplied by $\sigma^2$. 
Effect of scale normalization

Original signal

Unnormalized Laplacian response

Scale-normalized Laplacian response

maximum
Blob detection in 2D

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

\[
\nabla^2 \text{norm } g = \sigma^2 \left( \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)
\]

Scale-normalized:
Scale selection

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius \( r \)?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle.
- The Laplacian is given by (up to scale):
  \[
  (x^2 + y^2 - 2\sigma^2) \, e^{-(x^2 + y^2)/2\sigma^2}
  \]
- Therefore, the maximum response occurs at \( \sigma = \frac{r}{\sqrt{2}} \).
Characteristic scale

- We define the characteristic scale of a blob as the scale that produces peak of Laplacian response in the blob center.

Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales
Scale-space blob detector: Example
Scale-space blob detector: Example

\[ \text{sigma} = 11.9912 \]
Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space
Scale-space blob detector: Example
Efficient implementation

- Approximating the Laplacian with a difference of Gaussians:

\[ L = \sigma^2 \left( G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right) \]

(Dlaplacian)

\[ DoG = G(x, y, k\sigma) - G(x, y, \sigma) \]

(Difference of Gaussians)
Efficient implementation

Maximally Stable Extremal Regions [Matas ‘02]

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

• 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
From feature detection to feature description

- Scaled and rotated versions of the same neighborhood will give rise to blobs that are related by the same transformation.
- What to do if we want to compare the appearance of these image regions?
  - **Normalization**: transform these regions into same-size circles.
  - Problem: rotational ambiguity.
Eliminating rotation ambiguity

- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram
SIFT features

• Detected features with characteristic scales and orientations:

From feature detection to feature description

- Detection is *covariant*:
  \[ \text{features(\text{transform(image)})} = \text{transform(\text{features(image)})} \]
- Description is *invariant*:
  \[ \text{features(\text{transform(image)})} = \text{features(image)} \]
Properties of SIFT

Extraordinarily robust detection and description technique

- Can handle changes in viewpoint
  - Up to about 60 degree out-of-plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night
- Fast and efficient—can run in real time
- Lots of code available
Local Descriptors: SIFT Descriptor

- Histogram of oriented gradients
  - Captures important texture information
  - Robust to small translations / affine deformations

[Lowe, ICCV 1999]

K. Grauman, B. Leibe
Details of Lowe’s SIFT algorithm

- Run DoG detector
  - Find maxima in location/scale space
  - Remove edge points (bad localization)
- Find all major orientations
  - Bin orientations into 36 bin histogram
    - Weight by gradient magnitude
    - Weight by distance to center (Gaussian-weighted mean)
  - Return orientations within 0.8 of peak of histogram
    - Use parabola for better orientation fit
- For each (x,y,scale,orientation), create descriptor:
  - Sample 16x16 gradient mag. and rel. orientation
  - Bin 4x4 samples into 4x4 histograms
  - Threshold values to max of 0.2, divide by L2 norm
  - Final descriptor: 4x4x8 normalized histograms

\[
H = \begin{bmatrix}
D_{xx} & D_{xy} \\
D_{xy} & D_{yy}
\end{bmatrix}
\]

\[
\frac{\text{Tr}(H)^2}{\text{Det}(H)} < \frac{(r + 1)^2}{r}
\]

r: eigenvalue ratio

Lowe IJCV 2004
Matching SIFT Descriptors

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

![PDF graph showing distribution of correct and incorrect matches](Lowe IJCV 2004)
SIFT Repeatability

Graph showing the repeatability of SIFT features across different viewpoint angles.
SIFT Repeatability

![Graph showing the repeatability of SIFT descriptors with different numbers of orientations.](Lowe IJCV 2004)
Local Descriptors: SURF

Fast approximation of SIFT idea
Efficient computation by 2D box filters & integral images
⇒ 6 times faster than SIFT
Equivalent quality for object identification

[Bay, ECCV’06], [Cornelis, CVGPU’08]

K. Grauman, B. Leibe
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

K. Grauman, B. Leibe
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
  – All detectors/descriptors shown here work well
Choosing a descriptor

- Again, need not stick to one

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

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

- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT
Fitting

• We’ve learned how to detect edges, corners, blobs. Now what?
• We would like to form a higher-level, more compact representation of the features in the image by grouping multiple features according to a simple model
Fitting

- Choose a parametric model to represent a set of features

- Simple model: lines
- Simple model: circles
- Complicated model: car

Source: K. Grauman
Fitting: Overview

- If we know which points belong to the line, how do we find the “optimal” line parameters?
  - Least squares

- What if there are outliers?
  - Robust fitting, RANSAC

- What if there are many lines?
  - Voting methods: RANSAC, Hough transform

- What if we’re not even sure it’s a line?
  - Model selection (not covered)
Least squares line fitting

• Data: \((x_1, y_1), \ldots, (x_n, y_n)\)
• Line equation: \(y_i = mx_i + b\)
• Find \((m, b)\) to minimize

\[
E = \sum_{i=1}^{n} (y_i - mx_i - b)^2
\]

\[
E = \|Y - XB\|^2 \quad \text{where} \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \quad B = \begin{bmatrix} m \\ b \end{bmatrix}
\]

\[
E = \|Y - XB\|^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)
\]

\[
\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0
\]

\[
X^T XB = X^T Y
\]

\[
Pseudo-inverse \ solution \quad B = (X^T X)^{-1} X^T Y
\]
Problem with “vertical” least squares

• Not rotation-invariant
• Fails completely for vertical lines
Total least squares

• Distance between point $(x_i, y_i)$ and line $ax + by = d$ ($a^2 + b^2 = 1$): $|ax_i + by_i - d|

Unit normal: $N = (a, b)$

$ax + by = d$
Total least squares

- Distance between point \((x_i, y_i)\) and line \(ax + by = d\) \((a^2 + b^2 = 1)\): \(|ax_i + by_i - d|\)
- Find \((a, b, d)\) to minimize the sum of squared perpendicular distances

\[
E = \sum_{i=1}^{n} (ax_i + by_i - d)^2
\]
Total least squares

• Distance between point \((x_i, y_i)\) and line \(ax + by = d\) \((a^2 + b^2 = 1)\): 
  \[|ax_i + by_i - d|\]

• Find \((a, b, d)\) to minimize the sum of squared perpendicular distances

\[
E = \sum_{i=1}^{n} (ax_i + by_i - d)^2
\]

\[
\frac{\partial E}{\partial d} = \sum_{i=1}^{n} -2(ax_i + by_i - d) = 0
\]

\[
E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = (UN)^T (UN)
\]

\[
\frac{dE}{dN} = 2(U^TU)N = 0
\]

Solution to \((U^TU)N = 0\), subject to \(||N||^2 = 1\): eigenvector of \(U^TU\) associated with the smallest eigenvalue (least squares solution to homogeneous linear system \(UN = 0\))
Total least squares

\[ U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \]

\[ U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2 \end{bmatrix} \]

second moment matrix

F&P (2nd ed.) sec. 22.1
Least squares: Robustness to noise
Least squares: Robustness to noise

Problem: squared error heavily penalizes outliers
RANSAC

• Random sample consensus (RANSAC): Very general framework for model fitting in the presence of outliers

• Outline
  – Choose a small subset of points uniformly at random
  – Fit a model to that subset
  – Find all remaining points that are “close” to the model and reject the rest as outliers
  – Do this many times and choose the best model

RANSAC for line fitting example

Source: R. Raguram
RANSAC for line fitting example

Source: R. Raguram
1. Randomly select minimal subset of points

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram
RANSAC for line fitting

- Repeat $N$ times:
  - Draw $s$ points uniformly at random
  - Fit line to these $s$ points
  - Find *inliers* to this line among the remaining points (i.e., points whose distance from the line is less than $t$)
  - If there are $d$ or more inliers, accept the line and refit using all inliers
Choosing the parameters

• Initial number of points \( s \)
  – Typically minimum number needed to fit the model

• Distance threshold \( t \)
  – Choose \( t \) so probability for inlier is \( p \) (e.g. 0.95)
  – Zero-mean Gaussian noise with std. dev. \( \sigma \):
    \( t^2 = 3.84 \sigma^2 \)

• Number of samples \( N \)
  – Choose \( N \) so that, with probability \( p \), at least one
    random sample is free from outliers (e.g. \( p = 0.99 \))
    (outlier ratio: \( e \))

Source: M. Pollefeys
Choosing the parameters

- **Initial number of points** \( s \)
  - Typically minimum number needed to fit the model

- **Number of samples** \( N \)
  - Choose \( N \) so that, with probability \( p \), at least one random sample is free from outliers (e.g. \( p=0.99 \)) (outlier ratio: \( e \))

\[
\left(1-(1-e)^s\right)^N = 1-p
\]

\[N = \log(1-p)/\log(1-(1-e)^s)\]

<table>
<thead>
<tr>
<th>( s )</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>25%</th>
<th>30%</th>
<th>40%</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>19</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>13</td>
<td>17</td>
<td>34</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>6</td>
<td>12</td>
<td>17</td>
<td>26</td>
<td>57</td>
<td>146</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>7</td>
<td>16</td>
<td>24</td>
<td>37</td>
<td>97</td>
<td>293</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>8</td>
<td>20</td>
<td>33</td>
<td>54</td>
<td>163</td>
<td>588</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>9</td>
<td>26</td>
<td>44</td>
<td>78</td>
<td>272</td>
<td>1177</td>
</tr>
</tbody>
</table>

Source: M. Pollefeys
Choosing the parameters

• Initial number of points \( s \)
  – Typically minimum number needed to fit the model

• Distance threshold \( t \)
  – Choose \( t \) so probability for inlier is \( p \) (e.g. 0.95)
  – Zero-mean Gaussian noise with std. dev. \( \sigma \):
    \[ t^2 = 3.84 \sigma^2 \]

• Number of samples \( N \)
  – Choose \( N \) so that, with probability \( p \), at least one
    random sample is free from outliers (e.g. \( p = 0.99 \))
    (outlier ratio: \( e \))

• Consensus set size \( d \)
  – Should match expected inlier ratio

Source: M. Pollefeys
Adaptively determining the number of samples

- Outlier ratio $e$ is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield $e=0.2$

- Adaptive procedure:
  - $N=\infty$, $\text{sample\_count}=0$
  - While $N > \text{sample\_count}$
    - Choose a sample and count the number of inliers
    - If inlier ratio is highest of any found so far, set $e = 1 - (\text{number of inliers})/(\text{total number of points})$
    - Recompute $N$ from $e$:
      $$N = \frac{\log(1-p)}{\log(1-(1-e)^s)}$$
    - Increment the $\text{sample\_count}$ by 1

Source: M. Pollefeys
RANSAC pros and cons

- **Pros**
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice

- **Cons**
  - Computational time grows quickly with fraction of outliers and number of parameters
  - Not as good for getting multiple fits (though one solution is to remove inliers after each fit and repeat)
  - Sensitivity to threshold $t$
Slide Credits

• This set of slides contains contributions kindly made available by the following authors
  – Derek Hoiem
  – Svetlana Lazebnik
  – Kristen Grauman
  – Bastian Leibe
  – David Lowe
  – Marc Pollefeys
  – Rahul Raguram
  – Steve Seitz
  – Noah Snavely