A Script...

- We’ve created a script... for the part of the lecture on object recognition & categorization
  - K. Grauman, B. Leibe
  - Visual Object Recognition

- Chapter 3: Local Feature Extraction (Last+this lecture)
- Chapter 4: Matching (Tuesday’s topic)
- Chapter 5: Geometric Verification (Thursday’s topic)

- Available on the L2P -

Recap: Local Feature Matching Outline

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

Recap: Requirements for Local Features

- Problem 1:
  - Detect the same point independently in both images
- Problem 2:
  - For each point correctly recognize the corresponding one

We need a repeatable detector!

We need a reliable and distinctive descriptor!

Recap: Harris Detector [Harris88]

1. Compute second moment matrix (autocorrelation matrix)
   \[ M(\sigma, \sigma_y) = g(\sigma, \sigma_y) I(\sigma_y, \sigma_x) \]
   \[ I(\sigma_y, \sigma_x) = I(\sigma_y) I(\sigma_x) \]

2. Square of derivatives
3. Gaussian filter \( g(x) \)

4. Cornerness function - two strong eigenvalues
   \[ R = \text{det}(M(\sigma, \sigma_y)) - \alpha \text{trace}(M(\sigma, \sigma_y))^2 \]
   \[ = g(I(\sigma)^2) g(I(\sigma)^2) - [g(I(\sigma))]^2 - \alpha [g(I(\sigma))]^2 + g[I(\sigma)]^2 \]

5. Perform non-maximum suppression
Recap: Harris Detector Responses [Harris88]

Effect: A very precise corner detector.

Slide credit: Krystian Mikolajczyk

Recap: Hessian Detector Responses [Beaudet78]

Effect: Responses mainly on corners and strongly textured areas.

Slide credit: Krystian Mikolajczyk

Recap: Hessian Detector Responses [Beaudet78]

Effect: Responses mainly on corners and strongly textured areas.

Slide credit: Krystian Mikolajczyk

Recap: Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Slide credit: Krystian Mikolajczyk

Recap: Hessian Detector [Beaudet78]

- Hessian determinant

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

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

In Matlab:

\[
I_{xx} \ast I_{yy} - (I_{xy})^2
\]

Slide credit: Krystian Mikolajczyk

Recap: Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Slide credit: Krystian Mikolajczyk

Recap: Laplacian-of-Gaussian (LoG)

- Interest points:
  - Local maxima in scale space of Laplacian-of-Gaussian

Slide adapted from Krystian Mikolajczyk

Topics of This Lecture

- Local Feature Extraction (cont’d)
  - Scale Invariant Region Selection
  - Orientation normalization
  - Affine Invariant Feature Extraction

- Local Descriptors
  - SIFT
  - Applications

- Recognition with Local Features
  - Matching local features
  - Finding consistent configurations
  - Alignment: linear transformations
  - Affine estimation
  - Homography estimation

Slide credit: B. Leibe
Recap: LoG Detector Responses

Key point localization with DoG
- Detect maxima of difference-of-Gaussian (DoG) in scale space
- Then reject points with low contrast (threshold)
- Eliminate edge responses

Candidate keypoints: list of (x,y,σ)

Difference-of-Gaussian (DoG)
- We can efficiently approximate the Laplacian with a difference of Gaussians:
  \[ L = σ^2 \left( G_0(x,y,σ) + G_0(x,y,σ) \right) \] (Laplacian)
  \[ DoG = G(x,y,σ) - G(x,y,σ) \] (Difference of Gaussians)
- Advantages?
  - No need to compute 2nd derivatives.
  - Gaussians are computed anyway, e.g. in a Gaussian pyramid.

DoG - Efficient Computation
- Computation in Gaussian scale pyramid

Results: Lowe's DoG

Harris-Laplace [Mikolajczyk '01]
1. Initialization: Multiscale Harris corner detection
Harris-Laplace [Mikolajczyk '01]

1. Initialization: Multiscale Harris corner detection
2. Scale selection based on Laplacian (same procedure with Hessian \( \Rightarrow \) Hessian-Laplace)

Topics of This Lecture

- Local Feature Extraction (cont’d)
  - Scale Invariant Region Selection
  - Orientation normalization
  - Affine Invariant Feature Extraction
- Local Descriptors
  - SIFT
  - Applications
- Recognition with Local Features
  - Matching local features
  - Finding consistent configurations
  - Alignment: linear transformations
  - Affine estimation
  - Homography estimation

Rotation Invariant Descriptors

- Find local orientation
  - Dominant direction of gradient for the image patch
- Rotate patch according to this angle
  - This puts the patches into a canonical orientation.

Orientation Normalization: Computation

- Compute orientation histogram [Lowe, SIFT, 1999]
- Select dominant orientation
- Normalize: rotate to fixed orientation

Summary: Scale Invariant Detection

- Given: Two images of the same scene with a large scale difference between them.
- Goal: Find the same interest points independently in each image.
- Solution: Search for maxima of suitable functions in scale and in space (over the image).

- Two strategies
  - Laplacian-of-Gaussian (LoG)
  - Difference-of-Gaussian (DoG) as a fast approximation
  - These can be used either on their own, or in combinations with single-scale keypoint detectors (Harris, Hessian).
The Need for Invariance

- Up to now, we had invariance to
  - Translation
  - Scale
  - Rotation
- Not sufficient to match regions under viewpoint changes
  - For this, we need also affine adaptation

Affine Adaptation

- Problem:
  - Determine the characteristic shape of the region.
  - Assumption: shape can be described by "local affine frame".
- Solution: iterative approach
  - Use a circular window to compute second moment matrix.
  - Compute eigenvectors to adapt the circle to an ellipse.
  - Recompute second moment matrix using new window and iterate...

Iterative Affine Adaptation

1. Detect keypoints, e.g. multi-scale Harris
2. Automatically select the scales
3. Adapt affine shape based on second order moment matrix
4. Refine point location

Affine Normalization/Deskewing

- Steps
  - Rotate the ellipse’s main axis to horizontal
  - Scale the x axis, such that it forms a circle

Affine Adaptation Example

Scale-invariant regions (blobs)

Affine Adaptation Example

Affine-adapted blobs
Summary: Affine-Inv. Feature Extraction

Extract affine regions → Normalize regions → Eliminate rotational ambiguity → Compare descriptors

Invariance vs. Covariance

- Invariance:
  \[ \text{features(transform(image))} = \text{features(image)} \]
- Covariance:
  \[ \text{features(transform(image))} = \text{transform(features(image))} \]

Covariant detection \(\Rightarrow\) invariant description

Invariance vs. Covariance

- Invariance:
  \[ \text{features(transform(image))} = \text{features(image)} \]
- Covariance:
  \[ \text{features(transform(image))} = \text{transform(features(image))} \]

Covariant detection \(\Rightarrow\) invariant description

Topics of This Lecture

- Local Feature Extraction (cont’d)
  - Orientation normalization
  - Affine Invariant Feature Extraction
- Local Descriptors
  - SIFT
  - Applications
- Recognition with Local Features
  - Matching local features
  - Finding consistent configurations
  - Alignment: linear transformations
  - Affine estimation
  - Homography estimation

Local Descriptors

- We know how to detect points
- Next question:
  How to describe them for matching?

Point descriptor should be:
1. Invariant
2. Distinctive

Local Descriptors

- Simplest descriptor: list of intensities within a patch.
- What is this going to be invariant to?

Feature Descriptors

- Disadvantage of patches as descriptors:
  - Small shifts can affect matching score a lot

- Solution: histograms
Feature Descriptors: SIFT

- **Scale Invariant Feature Transform**
  - **Descriptor computation:**
    - Divide patch into 4x4 sub-patches: 16 cells
    - Compute histogram of gradient orientations (8 reference angles) for all pixels inside each sub-patch
    - Resulting descriptor: 4x4x8 = 128 dimensions


Overview: SIFT

- Extraordinarily robust matching technique
  - Can handle changes in viewpoint up to ~60 deg. out-of-plane rotation
  - Can handle significant changes in illumination
    - Sometimes even day vs. night (below)
  - Fast and efficient—can run in real time
  - Lots of code available

Working with SIFT Descriptors

- One image yields:
  - n 2D points giving positions of the patches
    - [n x 2 matrix]
  - n scale parameters specifying the size of each patch
    - [n x 1 vector]
  - n orientation parameters specifying the angle of the patch
    - [n x 1 vector]
  - n 128-dimensional descriptors: each one is a histogram of the gradient orientations within a patch
    - [n x 128 matrix]

B. Leibe

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]
  - GPU implementation available
    - Feature extraction @ 100Hz (detector + descriptor, 640x480 img)

You Can Try It At Home...

- For most local feature detectors, executables are available online:
  - [http://robots.ox.ac.uk/~vgg/research/affine/detectors.html#binaries](http://robots.ox.ac.uk/~vgg/research/affine/detectors.html#binaries)
  - [http://www.vision.ee.ethz.ch/~surf](http://www.vision.ee.ethz.ch/~surf)
  - [http://www.robots.ox.ac.uk/~vgg/research/affine](http://www.robots.ox.ac.uk/~vgg/research/affine)
Topics of This Lecture

- Local Feature Extraction (cont’d)
  - Orientation normalization
  - Affine Invariant Feature Extraction
- Local Descriptors
  - SIFT
  - Applications
- Recognition with Local Features
  - Matching local features
  - Finding consistent configurations
  - Alignment: linear transformations
  - Affine estimation
  - Homography estimation

Applications of Local Invariant Features

- Wide baseline stereo
- Motion tracking
- Panoramas
- Mobile robot navigation
- 3D reconstruction
- Recognition
  - Specific objects
  - Textures
  - Categories
- ...

Wide-Baseline Stereo

Automatic Mosaicing

Panorama Stitching

Recognition of Specific Objects, Scenes
Recognition of Categories

- Constellation model
- Bags of words

<table>
<thead>
<tr>
<th>Category</th>
<th>Sample Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Animals</td>
<td>[Images]</td>
</tr>
<tr>
<td>Vehicles</td>
<td>[Images]</td>
</tr>
<tr>
<td>Objects</td>
<td>[Images]</td>
</tr>
<tr>
<td>Landscapes</td>
<td>[Images]</td>
</tr>
</tbody>
</table>

Csurka et al. (2004), Derko & Schmid (2005), Fergus et al. (2005), Lazebnik et al. (2006), ...

Value of Local Features

- Advantages
  - Critical to find distinctive and repeatable local regions for multi-view matching.
  - Complexity reduction via selection of distinctive points.
  - Describe images, objects, parts without requiring segmentation; robustness to clutter & occlusion.
  - Robustness: similar descriptors in spite of moderate view changes, noise, blur, etc.

- How can we use local features for such applications?
  - Next: matching and recognition

Topics of This Lecture

- Local Feature Extraction (cont’d)
  - Orientation normalization
  - Affine Invariant Feature Extraction
- Local Descriptors
  - SIFT
  - Applications
- Recognition with Local Features
  - Matching local features
  - Finding consistent configurations
  - Alignment: linear transformations
  - Affine estimation
  - Homography estimation

Recognition with Local Features

- Image content is transformed into local features that are invariant to translation, rotation, and scale
- Goal: Verify if they belong to a consistent configuration

Warping vs. Alignment

Warping: Given a source image and a transformation, what does the transformed output look like?

Alignment: Given two images with corresponding features, what is the transformation between them?

Parametric (Global) Warping

Transformation $T$ is a coordinate-changing machine:

$$p' = T(p)$$

- What does it mean that $T$ is global?
  - It’s the same for any point $p$
  - It can be described by just a few numbers (parameters)
- Let’s represent $T$ as a matrix:

$$p' = Mp, \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} M' \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
2D Linear Transforms

\[
\begin{bmatrix}
  x' \\
  y'
\end{bmatrix} =
\begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix}
\begin{bmatrix}
  x \\
  y
\end{bmatrix}
\]

- Only linear 2D transformations can be represented with a 2x2 matrix.
- Linear transformations are combinations of ...
  - Scale,
  - Rotation,
  - Shear, and
  - Mirror

Homogeneous Coordinates

- Q: How can we represent translation as a 3x3 matrix using homogeneous coordinates?

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

- A: Using the rightmost column:

\[
\text{Translation} =
\begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\]

Basic 2D Transformations

- Basic 2D transformations as 3x3 matrices

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

- Scaling

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

- Rotation

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  \cos \theta & -\sin \theta & 0 \\
  \sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

- Shearing

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  1 & \Delta_x & 0 \\
  \Delta_y & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

2D Affine Transformations

- Affine transformations are combinations of ...
  - Linear transformations, and
  - Translations

- Parallel lines remain parallel
Projective Transformations

\[
\begin{bmatrix}
  x' \\
  y' \\
  w'
\end{bmatrix} = \begin{bmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{bmatrix} \begin{bmatrix}
  x \\
  y \\
  w
\end{bmatrix}
\]

- Projective transformations:
  - Affine transformations, and
  - Projective warps
- Parallel lines do not necessarily remain parallel

Alignment Problem

- We have previously considered how to fit a model to image evidence
  - e.g., a line to edge points
- In alignment, we will fit the parameters of some transformation according to a set of matching feature pairs ("correspondences").

Let’s Start with Affine Transformations

- Simple fitting procedure (linear least squares)
- Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras
- Can be used to initialize fitting for more complex models

Fitting an Affine Transformation

- Assuming we know the correspondences, how do we get the transformation?

Fitting an Affine Transformation

- Goal: a linear function to predict \( X' \)'s from \( X \)s:
  \( Xa + b = X' \)
- We want to find \( a \) and \( b \).
- How many \((X,X')\) pairs do we need?
  \[
  \begin{bmatrix}
    X_1 \\
    X_2 \\
    \vdots
  \end{bmatrix} + \begin{bmatrix}
    a \\
    b
  \end{bmatrix} = \begin{bmatrix}
    X_1' \\
    X_2' \\
    \vdots
  \end{bmatrix}
  \]
- What if the data is noisy?

Recall: Least Squares Estimation

- Set of data points: \((X_1,X_1'),(X_2,X_2'),(X_3,X_3')\)
- Goal: a linear function to predict \( X' \)'s from \( X \)s:
  \( Xa + b = X' \)
- We want to find \( a \) and \( b \).
- How many \((X,X')\) pairs do we need?
  \[
  \begin{bmatrix}
    X_1 \\
    X_2 \\
    \vdots
  \end{bmatrix} + \begin{bmatrix}
    a \\
    b
  \end{bmatrix} = \begin{bmatrix}
    X_1' \\
    X_2' \\
    \vdots
  \end{bmatrix}
  \]
- What if the data is noisy?

\[
\text{Overconstrained problem:} \quad \min \| Xa - B \|^2
\]

\[ x = A\backslash B \]

Matlab:

\[
\text{Least-squares minimization}
\]
Assuming we know the correspondences, how do we get the transformation?

This is called a projective transform. A projective transform is a mapping between any two perspectives with the same center of projection.

- Properties:
  - Rectangle should map to arbitrary quadrilateral
  - Parallel lines aren’t
  - but must preserve straight lines

This is called a homography.

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = 
\begin{bmatrix}
    h_{11} & h_{12} & h_{13} \\
    h_{21} & h_{22} & h_{23} \\
    h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

\[
\begin{bmatrix}
    w \\
    x' \\
    y' \\
    1
\end{bmatrix} = 
\begin{bmatrix}
    w \\
    x \\
    y \\
    1
\end{bmatrix}
\]

\[
\begin{bmatrix}
    p \\
    x' \\
    y' \\
    1
\end{bmatrix} = 
\begin{bmatrix}
    p \\
    x \\
    y \\
    1
\end{bmatrix}
\]

Once we have solved for the parameters, how do we compute the coordinates of the corresponding point for \((x'_{new}), (y'_{new})\)?

Homography

- A projective transform is a mapping between any two perspectives with the same center of projection.
  - i.e. two planes in 3D along the same sight ray
- Properties:
  - Rectangle should map to arbitrary quadrilateral
  - Parallel lines aren’t
  - but must preserve straight lines
- This is called a homography.

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = 
\begin{bmatrix}
    h_{11} & h_{12} & h_{13} \\
    h_{21} & h_{22} & h_{23} \\
    h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

\[
\begin{bmatrix}
    w \\
    x' \\
    y' \\
    1
\end{bmatrix} = 
\begin{bmatrix}
    w \\
    x \\
    y \\
    1
\end{bmatrix}
\]

\[
\begin{bmatrix}
    p \\
    x' \\
    y' \\
    1
\end{bmatrix} = 
\begin{bmatrix}
    p \\
    x \\
    y \\
    1
\end{bmatrix}
\]

To solve for the homography parameters, we need 8 matches (correspondence pairs).

Fitting a Homography

- Estimating the transformation

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix} = 
\begin{bmatrix}
    h_{11} & h_{12} & h_{13} \\
    h_{21} & h_{22} & h_{23} \\
    h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

Set scale factor to 1 ⇒ 8 parameters left.
Fitting a Homography

- Estimating the transformation

\[ x' = Hx \]

\[ x'' = \frac{1}{z'} x' \]

\[ \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \]

\[ A = UDV^T \]

\[ Ah = 0 \]

Minimizes least square error

Slide credit: Krystian Mikolajczyk
Image Warping with Homographies

Uses: Analyzing Patterns and Shapes

- What is the shape of the b/w floor pattern?

Analyzing Patterns and Shapes

- Basic matching algorithm
  1. Detect interest points in two images.
  2. Extract patches and compute a descriptor for each one.
  3. Compare one feature from image 1 to every feature in image 2 and select the nearest-neighbor pair.
  4. Repeat the above for each feature from image 1.
  5. Use the list of best pairs to estimate the transformation between images.

Summary: Recognition by Alignment

- Transformation estimation
  - Affine
  - Homography

References and Further Reading

- More details on homography estimation can be found in Chapter 4.7 of
  - R. Hartley, A. Zisserman
  - Multiple View Geometry in Computer Vision
  - 2nd Ed., Cambridge Univ. Press, 2004

- Details about the DoG detector and the SIFT descriptor can be found in
  - D. Lowe, Distinctive image features from scale-invariant keypoints,
  - IJCV 60(2), pp. 91-110, 2004

- Try the available local feature detectors and descriptors
  - http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html#binaries