Image Processing

Cosimo Distante

Lecture: Local Features
Last Time: Image Pyramids

• Image Pyramids
• Blending
Today: Feature Detection and Matching

- Local features
- Pyramids for invariant feature detection
- Invariant descriptors
- Matching
Image matching

by Diva Sian

by swashford
Harder case

by Diva Sian

by scgbt
Harder still?

NASA Mars Rover images
Answer below (look for tiny colored squares...)

NASA Mars Rover images with SIFT feature matches
Figure by Noah Snavely
Local features and alignment

- We need to match (align) images
- Global methods sensitive to occlusion, lighting, parallax effects. So look for local features that match well.
- How would you do it by eye?
Local features and alignment

• Detect feature points in both images
Local features and alignment

• Detect feature points in both images
• Find corresponding pairs
Local features and alignment

• Detect feature points in both images
• Find corresponding pairs
• Use these pairs to align images
Local features and alignment

- Problem 1:
  - Detect the \textit{same} point \textit{independently} in both images

no chance to match!

We need a repeatable \textbf{detector}
Local features and alignment

• Problem 2:
  – For each point correctly recognize the corresponding one

We need a reliable and distinctive descriptor

[Darya Frolova and Denis Simakov]
Geometric transformations
Photometric transformations
And other nuisances…

• Noise
• Blur
• Compression artifacts
• …
Invariant local features

Subset of local feature types designed to be invariant to common geometric and photometric transformations.

Basic steps:
1) Detect distinctive interest points
2) Extract invariant descriptors

Figure: David Lowe
Main questions

• Where will the interest points come from?
  – What are salient features that we’ll detect in multiple views?

• How to describe a local region?

• How to establish correspondences, i.e., compute matches?
Figure 4.3: *Image pairs with extracted patches below. Notice how some patches can be localized or matched with higher accuracy than others.*
Finding Corners

Key property: in the region around a corner, image gradient has two or more dominant directions.

Corners are repeatable and distinctive.


Source: Lana Lazebnik
Corners as distinctive interest points

We should easily recognize the point by looking through a small window.
Shifting a window in *any direction* should give a *large change* in intensity.

- **“flat”** region: no change in all directions
- **“edge”:** no change along the edge direction
- **“corner”:** significant change in all directions

Source: A. Efros
Harris Detector formulation

Change of intensity for the shift \([u, v]\):

\[
E(u, v) = \sum_{x, y} w(x, y) \left[ I(x + u, y + v) - I(x, y) \right]^2
\]

Window function

Shifted intensity

Intensity

Window function \(w(x, y) = \)

1 in window, 0 outside

or

Gaussian

Source: R. Szeliski
Feature detection: the math

Consider shifting the window $W$ by $(u,v)$

- how do the pixels in $W$ change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD “error” of $E(u,v)$:

\[
E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2
\]
Small motion assumption

Taylor Series expansion of $I$:

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \text{higher order terms}$$

If the motion $(u,v)$ is small, then first order approx is good

$$I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v$$

$$\approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide…
Consider shifting the window $W$ by $(u,v)$

- how do the pixels in $W$ change?
- compare each pixel before and after by summing up the squared differences
- this defines an “error” of $E(u,v)$:

$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

$$\approx \sum_{(x,y) \in W} [I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}]^2$$

$$\approx \sum_{(x,y) \in W} \left[[I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}\right]^2$$
Feature detection: the math

This can be rewritten:

\[ E(u, v) = \sum_{(x, y) \in W} [u \ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} [u \ v] \]

For the example above

- You can move the center of the green window to anywhere on the blue unit circle
- Which directions will result in the largest and smallest E values?
- We can find these directions by looking at the eigenvectors of \( M \)
Harris Detector formulation

This measure of change can be approximated by:

\[
E(u,v) \approx [u \ v] M [u \ v]^T
\]

where \( M \) is a 2×2 matrix computed from image derivatives:

\[
M = \sum_{x,y} w(x,y) \begin{bmatrix}
I_x^2 & I_x I_y \\
I_x I_y & I_y^2
\end{bmatrix}
\]

Gradient with respect to \( x \), times gradient with respect to \( y \)

Sum over image region – area we are checking for corner

\[
M = \begin{bmatrix}
\sum I_x I_x & \sum I_x I_y \\
\sum I_x I_y & \sum I_y I_y
\end{bmatrix} = \sum \begin{bmatrix}
I_x \\
I_y
\end{bmatrix} [I_x \ I_y]
\]
Harris Detector formulation

where $M$ is a $2 \times 2$ matrix computed from image derivatives:

$$M = \sum_{x, y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Sum over image region – area we are checking for corner

Gradient with respect to $x$, times gradient with respect to $y$
Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix $A$ are the vectors $x$ that satisfy:

$$Ax = \lambda x$$

The scalar $\lambda$ is the **eigenvalue** corresponding to $x$

- The eigenvalues are found by solving:

$$\det(A - \lambda I) = 0$$

- In our case, $A = M$ is a 2x2 matrix, so we have

$$\det\begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix} = 0$$

- The solution:

$$\lambda_{\pm} = \frac{1}{2} \left[ (m_{11} + m_{22}) \pm \sqrt{4m_{12}m_{21} + (m_{11} - m_{22})^2} \right]$$

Once you know $\lambda$, you find $x$ by solving

$$\begin{bmatrix} m_{11} - \lambda & m_{12} \\ m_{21} & m_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$
Feature detection: the math

$$E(u, v) = \sum_{(x, y) \in W} \begin{bmatrix} u \\ v \end{bmatrix} \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

**Eigenvalues and eigenvectors of H**

- Define shifts with the smallest and largest change (E value)
- $x_+ = \text{direction of largest increase in } E$.
- $\lambda_+ = \text{amount of increase in direction } x_+$
- $x_- = \text{direction of smallest increase in } E$.
- $\lambda_- = \text{amount of increase in direction } x_-$

$$Mx_+ = \lambda_+ x_+$$
$$Mx_- = \lambda_- x_-$$
What does this matrix reveal?

First, consider an axis-aligned corner:
What does this matrix reveal?

First, consider an axis-aligned corner:

\[
M = \begin{bmatrix}
\sum I_x^2 & \sum I_x I_y \\
\sum I_x I_y & \sum I_y^2 \\
\end{bmatrix} = \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2 \\
\end{bmatrix}
\]

This means dominant gradient directions align with \( x \) or \( y \) axis.

If either \( \lambda \) is close to 0, then this is **not** a corner, so look for locations where both are large.

What if we have a corner that is not aligned with the image axes?
General Case

Since $M$ is symmetric, we have

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

We can visualize $M$ as an ellipse with axis lengths determined by the eigenvalues and orientation determined by $R$.
Interpreting the eigenvalues

Classification of image points using eigenvalues of $M$:

- **“Corner”**: $\lambda_1$ and $\lambda_2$ are large, $\lambda_1 \sim \lambda_2$; $E$ increases in all directions.
- **“Edge”**: $\lambda_1 >> \lambda_2$.
- **“Flat” region**: $\lambda_1$ and $\lambda_2$ are small; $E$ is almost constant in all directions.

$\lambda_1$ and $\lambda_2$ are the eigenvalues of the matrix $M$. The classification of image points is based on the relative sizes of these eigenvalues.
Corner response function

\[ R = \det(M) - \alpha \text{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2 \]

\( \alpha: \) constant (0.04 to 0.06)
Harris Corner Detector

- Algorithm steps:
  - Compute $M$ matrix within all image windows to get their $R$ scores
  - Find points with large corner response ($R > \text{threshold}$)
  - Take the points of local maxima of $R$
Harris Detector: Workflow

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.
Harris Detector: Workflow

Compute corner response $R$
Harris Detector: Workflow

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

Take only the points of local maxima of $R$
Harris Detector: Workflow
Harris Detector: Properties

- Rotation invariance

Ellipse rotates but its shape (i.e. eigenvalues) remains the same.

Corner response $R$ is invariant to image rotation.
Harris Detector: Properties

• Not invariant to image scale

All points will be classified as edges  Corner!
• How can we detect **scale invariant** interest points?
How to cope with transformations?

• Exhaustive search
• Invariance
• Robustness
Exhaustive search

• Multi-scale approach
Exhaustive search

- Multi-scale approach
Exhaustive search

- Multi-scale approach
Exhaustive search

- Multi-scale approach
Invariance

• Extract patch from each image individually
Automatic scale selection

- Solution:
  - Design a function on the region, which is “scale invariant” (the same for corresponding regions, even if they are at different scales)
    
    Example: average intensity. For corresponding regions (even of different sizes) it will be the same.
    
  - For a point in one image, we can consider it as a function of region size (patch width)
Automatic scale selection

• Common approach:

Take a local maximum of this function

Observation: region size, for which the maximum is achieved, should be *invariant* to image scale.

**Important:** this scale invariant region size is found in each image *independently*!
Automatic Scale Selection

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

Same operator responses if the patch contains the same image up to scale factor.
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)

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

\[ f(I_{i_1 \ldots i_m}(x', \sigma)) \]
Automatic Scale Selection

- Function responses for increasing scale (scale signature)

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

\[ f(I_{i_1 \ldots i_m}(x', \sigma)) \]
Automatic Scale Selection

- Function responses for increasing scale (scale signature)

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

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

- Function responses for increasing scale (scale signature)
Scale selection

- Use the scale determined by detector to compute descriptor in a normalized frame

[Images from T. Tuytelaars]
What Is A Useful Signature Function?

- Laplacian-of-Gaussian = “blob” detector
Characteristic scale

We define the *characteristic scale* as the scale that produces peak of Laplacian response.

Scale-space blob detection

$L_{xx}(\sigma) + L_{yy}(\sigma) \rightarrow \sigma^3$

Squared filter response maps

$\Rightarrow$ List of $(x, y, \sigma)$
Laplacian-of-Gaussian (LoG)

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

\[ L_{xx}(\sigma) + L_{yy}(\sigma) \]

\[ \Rightarrow \text{List of } (x, y, \sigma) \]
Scale-space blob detector: Example

Source: Lana Lazebnik
Scale-space blob detector: Example

\[ \text{sigma} = 11.9912 \]

Source: Lana Lazebnik
Scale-space blob detector: Example

Source: Lana Lazebnik
Image Processing

Cosimo Distante

Lecture 13: Local Features (SIFT)
SIFT Background

Scale-invariant feature transform

- **SIFT**: to detect and describe local features in an images.
- Cited more than 41180 times till now.
- Wildly used in image search, object recognition, video tracking, gesture recognition, *etc*.

*Distinctive image features from scale-invariant keypoints*
David G Lowe
International journal of computer vision 60 (2), 91-110
Scale space

- Keypoints are detected using scale-space extrema in difference-of-Gaussian function $D$

- $D$ definition:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

$$= L(x, y, k\sigma) - L(x, y, \sigma)$$

- Efficient to compute
Relationship of $D$ to $\sigma^2 \nabla^2 G$

- Close approximation to scale-normalized Laplacian of Gaussian, $\sigma^2 \nabla^2 G$

Diffusion equation: $\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$

Approximate $\partial G/\partial \sigma$:

\[ \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \]

- giving,

\[ \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma} \approx \sigma \nabla^2 G \]

$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G$

When $D$ has scales differing by a constant factor it already incorporates the $\sigma^2$ scale normalization required for scale-invariance
Scale space construction

\[
G(x, y, k\sigma) = \frac{1}{2\pi (k\sigma)^2} e^{-(x^2 + y^2)/(2k^2\sigma^2)}
\]

\[
k = \sqrt{2}
\]
**Octave**

Similar to music:

it is the doubling (or halving) of the scale.

When you are talking about frequencies,
- an octave **higher** means double the frequency,
  and
- an octave **lower** is half the frequency.

In vision and images every octave (usually in an image pyramid where the source image is scaled down at a fixed rate e.g. 1.2x each time) is a doubling or halving of the size/scale.
Scale Space Construction

<table>
<thead>
<tr>
<th>Scale</th>
<th>0.7072</th>
<th>1.0001</th>
<th>1.4144</th>
<th>2.0002</th>
<th>2.8287</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octave</td>
<td>1.4144</td>
<td>2.0002</td>
<td>2.8287</td>
<td>4.0005</td>
<td>5.6575</td>
</tr>
<tr>
<td></td>
<td>2.8287</td>
<td>4.0005</td>
<td>5.6575</td>
<td>8.0009</td>
<td>11.3150</td>
</tr>
<tr>
<td></td>
<td>5.6575</td>
<td>8.0009</td>
<td>11.3150</td>
<td>16.0018</td>
<td>22.6300</td>
</tr>
</tbody>
</table>

\[ \sigma = 0.7072 \]

\[ k = \sqrt{2} \]
How many scales per octave?
Initial value of sigma

Figure 4: The top line in the graph shows the percent of keypoint locations that are repeatably detected in a transformed image as a function of the prior image smoothing for the first level of each octave. The lower line shows the percent of descriptors correctly matched against a large database.

• Sigma=1.6
Key point localization with DoG

- Detect local extrema of difference-of-Gaussian (DoG) in scale space. Compare X with 26 pixels (adjacent pixels in scales and those in adjacent scales).
- Then reject points (outliers) with low contrast (threshold).
- Eliminate edge responses.
Initial Outlier Rejection

• Low contrast candidates
• Poorly localized candidates along an edge
• Taylor series expansion of DOG, \( D \)

\[
D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \ast I(x, y)
\]

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

where

\[
X = (x, y, \sigma)^T
\]
Initial Outlier Rejection

- Take the derivative with respect to $X$, and set it to 0, giving

$$0 = \frac{\partial D}{\partial X} + \frac{\partial^2 D}{\partial X^2} \hat{X}$$

$$\hat{X} = - \frac{\partial^2 D^{-1}}{\partial X^2} \frac{\partial D}{\partial X}$$

is the location of the keypoint

This is a 3x3 linear system
Localization

Derivatives approximated by finite differences,
– example:

If \( X > 0.5 \) in any dimension, process repeated
Filtering

Contrast (use prev. equation):
- If $|D(X)| < 0.03$, throw it out

Edge-iness:
- Use ratio of principal curvatures to throw out poorly defined peaks
- Curvatures come from Hessian:
- Ratio of $\text{Trace}(H)^2$ and $\text{Determinant}(H)$

$$D(\hat{X}) = D + \frac{1}{2} \frac{\partial D^T}{\partial X} \hat{X}$$

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

$$\text{Tr}(H) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2$$

$$\text{Det}(H) = D_{xx} D_{yy} - (D_{xy})^2 = \lambda_1 \lambda_2$$
Filtering

\[ Tr(H) = D_{xx} + D_{yy} = \lambda_1 + \lambda_2 \]

\[ Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \lambda_1 \lambda_2 \]

If ratio

\[ \frac{Tr(H)^2}{Det(H)} = \frac{(D_{xx} + D_{yy})^2}{D_{xx}D_{yy} - (D_{xy})^2} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} = \frac{(r+1)^2}{r} \]

Throw it out

\[ r = \frac{\lambda_1}{\lambda_2} \]

Eliminates keypoint if \( r > 10 \)
Example of keypoint detection

(a) 233x189 image
(b) 832 DOG extrema
(c) 729 left after peak value threshold
(d) 536 left after testing ratio of principle curvatures (removing edge responses)
Orientation assignment

- Descriptor computed relative to keypoint’s orientation achieves rotation invariance
- Precomputed along with mag. for all levels (useful in descriptor computation)

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

\[ \theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y))) \]

- Multiple orientations assigned to keypoints from an orientation histogram
  - Significantly improve stability of matching
Orientation assignment

Descriptor computed relative to keypoint’s orientation achieves rotation invariance

Precomputed along with mag. for all levels (useful in descriptor computation)

Multiple orientations assigned to keypoints from an orientation histogram
  – Significantly improve stability of matching

\[ D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \ast I(x, y) \]
\[ = L(x, y, k\sigma) - L(x, y, \sigma) \]

\[ m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \]
\[ \theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y))) \]
Main questions

• Where will the interest points come from?
  – What are salient features that we’ll detect in multiple views?
• How to describe a local region?
• How to establish correspondences, i.e., compute matches?
Local Image descriptor at Key Points

• Possible descriptor
  – Store intensity samples in the neighborhood
  – Sensitive to lighting changes, 3D object transformation

• Use of Gradient Orientation histograms
  – Robust representation
Similarity to IT Cortex

- Complex neurons respond to a gradient at a particular orientation
- Location of the feature can shift over a small receptive field
  - The function of the cells allow for matching and recognition of 3D objects from a range of view points
- Experiments show better recognition accuracy for 3D objects rotated in depth by up to 20 degrees
Orientation assignment

• Create a weighted direction histogram in a neighborhood of a key point (for example 4/8/16 or more bins)

Weights are
  o Gradient magnitude
  o Spatial Gaussian filter with $\sigma=1.5 \times <\text{scale of key point}>$
Rotation Invariant Descriptors

- **Harris corner response measure:** depends only on the eigenvalues of the matrix $M$
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.
Rotation Invariant Descriptors

Image from Matthew Brown
Orientation Assignment:

- Calculate weighted orientation histogram

36 buckets
10 degree range of angles in each bucket, i.e.

- $0 \leq \text{ang} < 10$ : bucket 1
- $10 \leq \text{ang} < 20$ : bucket 2
- $20 \leq \text{ang} < 30$ : bucket 3 …
Orientation Assignment:
- Calculate weighted orientation histogram

Orientation of keypoint is approximately 25 degrees
SIFT (Scale Invariant Feature Transform)

Keypoints Descriptor:

- Empirical result:
  - Cell size: 4×4 pixels
  - Block size: 4×4 cells
  - Dimension: 4×4 (cells) × 8 (bins) = 128

Weighted magnitude
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

Feature descriptors: SIFT

Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affine viewpoint change of 50 degrees and addition of 4% noise.
Feature descriptors: SIFT

- Store numbers in a vector
- Normalize to unit vector (UN)
  - Illumination invariance (affine changes)
- For non-linear intensity transforms
  - Bound Unit Vector items to maximum 0.2 (remove large gradients)
  - Renormalize to unit vector
Scale Invariant Detection: Summary

- **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)
Local descriptors

• We know how to detect points
• Next question:

How to describe them for matching?

Point descriptor should be:
1. Invariant
2. Distinctive
Feature descriptors: SIFT

Extraordinarily robust matching 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 (below)
- Fast and efficient—can run in real time
- Lots of code available
Working with SIFT descriptors

• One image yields:
  – \(n\) 128-dimensional descriptors: each one is a histogram of the gradient orientations within a patch
    • \([n \times 128\) matrix]\]
  – \(n\) scale parameters specifying the size of each patch
    • \([n \times 1\) vector]\]
  – \(n\) orientation parameters specifying the angle of the patch
    • \([n \times 1\) vector]\]
  – \(n\) 2d points giving positions of the patches
    • \([n \times 2\) matrix]\]
More on feature detection/description

Affine Covariant Regions

Publications

Region detectors


Region descriptors


Performance evaluation

More feature detection/description

- FAST corner detector http://www.edwardrosten.com/work/fast.html
- AGAST http://www6.in.tum.de/Main/ResearchAgast
- BREALF
- DAISY
- SURF Speeded up Robust Features
  http://www.vision.ee.ethz.ch/~surf/
- **BRISK: Binary Robust Invariant Scalable Keypoints**
  http://www.asl.ethz.ch/people/lesefand/personal/BRISK
Main questions

• Where will the interest points come from?
  – What are salient features that we’ll detect in multiple views?

• How to *describe* a local region?

• How to establish *correspondences*, i.e., compute matches?
Feature descriptors

We know how to detect **and describe** good points

Next question: **How to match them?**
Feature matching

Given a feature in $I_1$, how to find the best match in $I_2$?

1. Define distance function that compares two descriptors
2. Test all the features in $I_2$, find the one with min distance
Feature distance

How to define the difference between two features $f_1$, $f_2$?

- Simple approach is $SSD(f_1, f_2)$
  - sum of square differences between entries of the two descriptors
  - can give good scores to very ambiguous (bad) matches
Feature distance

How to define the difference between two features $f_1$, $f_2$?

- Better approach: ratio distance $= \frac{\text{SSD}(f_1, f_2)}{\text{SSD}(f_1, f_2')}$
  - $f_2$ is best SSD match to $f_1$ in $I_2$
  - $f_2'$ is 2nd best SSD match to $f_1$ in $I_2$
  - gives small values for ambiguous matches
Feature distance sum of squared differences

\[
SSD = \sum_{l} \sum_{k} (f_1(k, l) - f_2(i + k, j + l))^2
\]

Minimize

\[
SSD = \sum_{l} \sum_{k} \left( f_1(k, l)^2 - 2 f_1(k, l) f_2(i + k, j + l) + f_2(i + k, j + l)^2 \right)
\]

Minimize

\[
SSD = \sum_{l} \sum_{k} \left( f_1(k, l)^2 - 2 f_1(k, l) f_2(i + k, j + l) + f_2(i + k, j + l)^2 \right)
\]

\[
SSD = \sum_{l} \sum_{k} (-2 f_1(k, l) f_2(i + k, j + l))
\]

\[
SSD = \sum_{l} \sum_{k} (2 f_1(k, l) f_2(i + k, j + l)) \quad \text{MAXIMIZE}
\]

\[
Correlation = \sum_{l} \sum_{k} (f_1(k, l) f_2(i + k, j + l)) \quad \text{MAXIMIZE}
\]
Evaluating the results

How can we measure the performance of a feature matcher?

feature distance
The distance threshold affects performance
  - True positives = # of detected matches that are correct
    - Suppose we want to maximize these—how to choose threshold?
  - False positives = # of detected matches that are incorrect
    - Suppose we want to minimize these—how to choose threshold?
Object Detection

- Create a database of keypoints from training images
- Match keypoints to a database
  - Nearest neighbor search
Figure 4.22  False positives and negatives: The black digits 1 and 2 are features being matched against a database of features in other images. At the current threshold setting (the solid circles), the green 1 is a *true positive* (good match), the blue 1 is a *false negative* (failure to match), and the red 3 is a *false positive* (incorrect match). If we set the threshold higher (the dashed circles), the blue 1 becomes a true positive but the brown 4 becomes an additional false positive.
Object Detection

Figure 4.24
At a fixed di incorrectly matches but $D_D$ incorrectly rejects matches for $D_D$.

NNDR = $\frac{d_1}{d_2} = \frac{||D_A - D_B||}{||D_A - D_C||}$, no matching. $D_B$ and $D_D$ matches $D_B$. ($D_D$ matching, $d_2'$ correctly
Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.
Matching
Logo Detection in Broadcast Videos

Fig. 3. Examples of 4 of the 32 logo classes of FlickrLogos-32 Dataset

(a) Heineken  (b) Esso

(c) Guinness  (d) Ferrari

Fig. 5. Screenshots taken from the broadcast videos considered in the experiments. In each picture is visible at least one instance of the searched logos.
(a) Dreher Lemon (b) Cariparma (c) Michelin
(d) Total Erg (e) Edison (f) Skoda

Fig. 6. Query logos for broadcast videos
Fig. 7. How the algorithm works to detect multiple logo occurrences in images (Color figure online)

Fig. 8. Example of the logo detection: the red stars represent inliers whereas the blue ones represent outliers (Color figure online)

Figure 8 better details how the algorithm works: the red stars in the yellow bounding box represents the inliers, contrarily, the blue stars represent the outliers or the wrong descriptors matching.

It is also possible to note how the algorithm is robust against heavy logo occlusions (see Fig. 9).

In Table 1 are reported the detection performance: the lowest detection rate was obtained for the Skoda logo (see Fig. 10d). We expected this results because the Skoda logo was mainly located on the pilot’s sleeves and this means that it was difficult to estimate an in-plane geometric transformation since the features points lie on no-plane surface. Same considerations for the detection rate experienced for the Cariparma logo that was, often, located on the players shirt. However in that case, the implemented geometric estimator correctly handled this quasi-planar surface, allowing successful detections of the logo areas (see Fig. 10c).

Logo Detection in Broadcast Videos

**Fig. 8.** Example of the logo detection: the red stars represent inliers whereas the blue ones represent outliers (Color figure online)

Logo Detection in Broadcast Videos

(a) Dreher Lemon  (b) Cariparma

(a) Total Erg  (b) Michelin  (c) Cariparma

(d) Skoda  (e) Edison  (f) Dreher Lemon

Today: Feature Detection and Matching

- Local features
- Pyramids for invariant feature detection
- Local descriptors
- Matching
Lots of applications

Features are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking
- Object recognition
- Indexing and database retrieval
- Robot navigation
- … other
Object recognition (David Lowe)