Panoramic stitching

Harder case

by Diva Sian

by scgbt
What makes a good feature?
Want uniqueness

Look for image regions that are unusual
  • Lead to unambiguous matches in other images

How to define “unusual”? 
Local measures of uniqueness

Suppose we only consider a small window of pixels

• What defines whether a feature is a good or bad candidate?

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.
Feature detection

Local measure of feature uniqueness

- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*

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

Slide adapted from Darya Frolova, Denis Simakov, Weizmann Institute.
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 $E(u,v)$:

$$E_{x_0,y_0}(u,v) = \sum_{(x,y) \in W(x_0,y_0)} \left[ I(x+u, y+v) - I(x, y) \right]^2$$

$$\text{corner}(x_0,y_0) = \min_{u^2+v^2=1} E_{x_0,y_0}(u,v)$$

We can approximate corner-score with a Taylor expansion of $E(u,v)$

Many existing implementation of Harris-corner detector
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…
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
- this defines an “error” of $E(u,v)$:

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

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

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

This can be rewritten:

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

Eigenvalues and eigenvectors of \( H \)

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

\[
\begin{align*}
H x_+ &= \lambda_+ x_+ \\
H x_- &= \lambda_- x_-
\end{align*}
\]
Feature detection summary

Here’s what you do

- Compute the gradient at each point in the image
- Create the $H$ matrix from the entries in the gradient
- Compute the eigenvalues.
- Find points with large response ($\lambda_- >$ threshold)
- Choose those points where $\lambda_-$ is a local maximum as features
The Harris operator

\( \lambda \) is a variant of the “Harris operator” for feature detection

\[
f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{\text{determinant}(H)}{\text{trace}(H)}
\]

- The trace is the sum of the diagonals, i.e., \( \text{trace}(H) = h_{11} + h_{22} \)
- Very similar to \( \lambda \) but less expensive (no square root)
- Called the “Harris Corner Detector” or “Harris Operator”
- Lots of other detectors, this is one of the most popular
The Harris operator
Harris detector example
f value (red high, blue low)
Threshold \((f > \text{value})\)
Find local maxima of $f$
Harris features (in red)
Invariance

Suppose you rotate the image by some angle
  • Will you still pick up the same features?

What if you change the brightness?

Scale?
Scale invariant detection

Suppose you’re looking for corners

Key idea: find scale that gives local maximum of f

- f is a local maximum in both position and scale
- Common definition of f: Laplacian (or difference between two Gaussian filtered images with different sigmas)
Automatic scale selection

Lindeberg et al., 1996
Automatic scale selection

Function responses for increasing scale
Scale trace (signature)

\[ f(l_{h \ldots m}(x, \sigma)) \]
Automatic scale selection

Function responses for increasing scale
Scale trace (signature)
Automatic scale selection

Function responses for increasing scale
Scale trace (signature)
Automatic scale selection

Function responses for increasing scale
Scale trace (signature)
Automatic scale selection

Function responses for increasing scale

Scale trace (signature)
Automatic scale selection

Function responses for increasing scale
Scale trace (signature)
Automatic scale selection

Normalize: rescale to fixed size
Feature descriptors

We know how to detect good points
Next question: **How to match them?**

Lots of possibilities (this is a popular research area)

- Simple option: match square windows around the point
- State of the art approach: SIFT
Rotation invariance for feature descriptors

Find dominant orientation of the image patch

- This is given by $\mathbf{x}_+$, the eigenvector of $\mathbf{H}$ corresponding to $\lambda_+$
  - $\lambda_+$ is the *larger* eigenvalue
- Rotate the patch according to this angle

Figure by Matthew Brown
Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window

Adapted from slide by Matthew Brown
Detections at multiple scales

Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.
**Scale Invariant Feature Transform**

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

Adapted from slide by David Lowe
SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

Adapted from slide by David Lowe
Properties of 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
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
Code

- [http://www.vlfeat.org/overview/sift.html](http://www.vlfeat.org/overview/sift.html)