Summary of Previous Lecture

A homography transforms one 3d plane to another 3d plane, under perspective projections. Those planes can be camera imaging planes or planes in the scene.

If a camera is stationary, a homography can perfectly relate two photos taken at different rotations and focal lengths.

We covered how to capture and build a panorama.
Automatic Image Alignment

© Mike Nese

Slides from Alexei Efros, Steve Seitz, and Rick Szeliski

cs129: Computational Photography
James Hays, Brown, Fall 2012
Image Alignment

How do we align two images automatically?

Two broad approaches:

• Feature-based alignment
  – Find a few matching features in both images
  – compute alignment

• Direct (pixel-based) alignment
  – Search for alignment where most pixels agree
Direct Alignment

The simplest approach is a brute force search (project 1)

- Need to define image matching function
  - SSD, Normalized Correlation, edge matching, etc.
- Search over all parameters within a reasonable range:

  e.g. for translation:
  
  for tx=x0:step:x1,
    for ty=y0:step:y1,
      compare image1(x,y) to image2(x+tx,y+ty)
    end;
  end;

Need to pick correct $x_0, x_1$ and step

- What happens if step is too large?
Direct Alignment (brute force)

What if we want to search for more complicated transformation, e.g. homography?

\[
\begin{bmatrix}
w x' \\
w y' \\
w \\
\end{bmatrix}
= \begin{bmatrix}
a & b & c \\
d & e & f \\
g & h & i \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1 \\
\end{bmatrix}
\]

for \(a=a_0:a_{\text{step}}:a_1\),

for \(b=b_0:b_{\text{step}}:b_1\),

for \(c=c_0:c_{\text{step}}:c_1\),

for \(d=d_0:d_{\text{step}}:d_1\),

for \(e=e_0:e_{\text{step}}:e_1\),

for \(f=f_0:f_{\text{step}}:f_1\),

for \(g=g_0:g_{\text{step}}:g_1\),

for \(h=h_0:h_{\text{step}}:h_1\),

compare image1 to \(H(\text{image2})\)

end; end; end; end; end; end; end; end; end;
Problems with brute force

Not realistic

• Search in $O(N^8)$ is problematic
• Not clear how to set starting/stopping value and step

What can we do?

• Use pyramid search to limit starting/stopping/step values
• For special cases (rotational panoramas), can reduce search slightly to $O(N^4)$:
  $H = K_1R_1R_2^{-1}K_2^{-1}$ (4 DOF: $f$ and rotation)

Alternative: gradient descent on the error function

• i.e. how do I tweak my current estimate to make the SSD error go down?
• Can do sub-pixel accuracy
• BIG assumption?
  – Images are already almost aligned (<2 pixels difference!)
  – Can improve with pyramid
• Same tool as in motion estimation
Image alignment
Feature-based alignment (project 6)

1. Find a few important features (aka Interest Points)
2. Match them across two images
3. Compute image transformation

How do we choose good features?

- They must prominent in both images
- Easy to localize
- Think how you would do it by hand
- Corners!
Feature Detection
Feature Matching

How do we match the features between the images?

• Need a way to describe a region around each feature
  – e.g. image patch around each feature
• Use successful matches to estimate homography
  – Need to do something to get rid of outliers

Issues:

• What if the image patches for several interest points look similar?
  – Make patch size bigger
• What if the image patches for the same feature look different due to scale, rotation, exposure etc.
  – Need an invariant descriptor
Invariant Feature Descriptors

Invariant Local Features

Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters.
Applications

Feature points are used for:

• Image alignment (homography, fundamental matrix)
• 3D reconstruction
• Motion tracking
• Object recognition
• Scene categorization
• Indexing and database retrieval
• Robot navigation
• ... other
Today’s lecture

- 1 Feature detector
  - scale invariant Harris corners
- 1 Feature descriptor
  - patches, oriented patches

Reading:


(linked from Project 6)
Harris corner detector

The Basic Idea

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

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

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

1 in window, 0 outside

or

Gaussian
Harris Detector: Mathematics

For small shifts \([u,v]\) we have a *bilinear* approximation:

\[
E(u,v) \approx u, v \quad M \begin{bmatrix} u \\ v \end{bmatrix}
\]

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

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

\[
A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T
\]
Harris Detector: Mathematics

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**
  - $\lambda_2 \gg \lambda_1$;
  - $\lambda_1$ and $\lambda_2$ are large,
  - $E$ increases in all directions

- **Flat** region
  - $\lambda_1$ and $\lambda_2$ are small;
  - $E$ is almost constant in all directions

But eigenvalues are expensive to compute
Measure of corner response:

\[ R = \frac{\det M}{\text{Trace } M} \]

\[
\begin{align*}
\det M &= \lambda_1 \lambda_2 \\
\text{trace } M &= \lambda_1 + \lambda_2
\end{align*}
\]
Harris Detector

The Algorithm:

- Find points with large corner response function $R$ ($R >$ threshold)
- Take the points of local maxima of $R$
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: Some Properties

Rotation invariance

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

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

Partial invariance to *affine intensity* change

✓ Only derivatives are used \(\Rightarrow\) invariance to intensity shift \(I \rightarrow I + b\)
✓ Intensity scale: \(I \rightarrow a I\)
Harris Detector: Some Properties

But: non-invariant to *image scale*!

All points will be classified as *edges*
Scale Invariant Detection

Consider regions (e.g. circles) of different sizes around a point. Regions of corresponding sizes will look the same in both images.
Scale Invariant Detection

The problem: how do we choose corresponding circles \textit{independently} in each image?

Choose the scale of the “best” corner