Automatic Image Alignment

CS194: Intro to Comp. Vision and Comp. Photo
Alexei Efros, UC Berkeley, Fall 2022
Project Proposals due in a month (11/11)

for cs294-26 and others interested to do more
Alpha blending for Panorama Stitching

Alpha = .5 in overlap region
Setting alpha: center seam

$\text{Distance Transform bwdist}$

$\text{Alpha} = \text{logical}(dtrans1 > dtrans2)$
Simplification: Two-band Blending

Brown & Lowe, 2003

- Only use two bands: high freq. and low freq.
- Blends low freq. smoothly
- Blend high freq. with no smoothing: use binary alpha
2-band “Laplacian Stack” Blending

Low frequency ($\lambda > 2$ pixels)

High frequency ($\lambda < 2$ pixels)
2-band Blending
Live Homography...
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 (hw1)

- 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?

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

for \(a=a_0:astep:a_1,\)
for \(b=b_0:bstep:b_1,\)
for \(c=c_0:cstep:c_1,\)
for \(d=d_0:dstep:d_1,\)
for \(e=e_0:estep:e_1,\)
for \(f=f_0:fstep:f_1,\)
for \(g=g_0:gstep:g_1,\)
for \(h=h_0:hstep:h_1,\)
compare image1 to \(H(\text{image2})\)
end; 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

Alternative: gradient decent 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

1. **Feature Detection**: find a few important features (aka Interest Points) in each image separately
2. **Feature Matching**: match them across two images
3. **Compute image transformation**: as per Project 5, Part I

How do we **choose** good features automatically?

- They must be prominent in both images
- Easy to localize
- Think how you did that by hand in Project #6 Part I
- Corners!
A hard feature matching problem

NASA Mars Rover images
NASA Mars Rover images with SIFT feature matches
Figure by Noah Snavely
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, 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
- Indexing and database retrieval
- Robot navigation
- … other
Today’s lecture

• 1 Feature detector
  • scale invariant Harris corners

• 1 Feature descriptor
  • patches, oriented patches

Reading:

  Multi-image Matching using Multi-scale image patches, CVPR 2005
Feature Detector – Harris Corner
Harris corner detector

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.
Harris Detector: Basic Idea

“flat” region: no change in all directions

“edge”: no change along the edge direction

“corner”: significant change in all directions
Corner Detection: Mathematics

Change in appearance of window $W$ for the shift $[u,v]$:

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

Change in appearance of window $W$ for the shift $[u,v]$:

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

Change in appearance of window $W$ for the shift $[u, v]$:

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

We want to find out how this function behaves for small shifts
Corner Detection: Mathematics

- First-order Taylor approximation for small motions \([u, v]\):

\[
I(x+u, y+v) = I(x, y) + I_x u + I_y v + \text{higher order terms}
\]

\[
\approx I(x, y) + I_x u + I_y v
\]

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

- Let’s plug this into

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

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

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

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

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

The quadratic approximation simplifies to

\[ E(u, v) \approx [u \quad v] M [u \\ v] \]

where \( M \) is a *second moment matrix* computed from image derivatives:

\[ M = \sum_{(x, y) \in W} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \]
Interpreting the second moment matrix

- The surface $E(u, v)$ is locally approximated by a quadratic form. Let’s try to understand its shape.

- Specifically, in which directions does it have the smallest/greatest change?

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

\[
M = \sum_{(x, y) \in W} \begin{bmatrix}
I_x^2 & I_x I_y \\
I_x I_y & I_y^2
\end{bmatrix}
\]
Interpreting the second moment matrix

First, consider the axis-aligned case (gradients are either horizontal or vertical)

\[ M = \sum_{(x, y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \]

If either \( a \) or \( b \) is close to 0, then this is **not** a corner, so look for locations where both are large.
Interpreting the second moment matrix

Consider a horizontal “slice” of $E(u, v)$: 

$$
[u \ v] \ M \ \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}
$$

This is the equation of an ellipse.
Visualization of second moment matrices
Visualization of second moment matrices
Interpreting the second moment matrix

Consider a horizontal “slice” of $E(u, v)$: $[u \ v] \ M \ [u \ v] = \text{const}$

This is the equation of an ellipse.

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

The axis lengths of the ellipse are determined by the eigenvalues and the orientation is determined by $R$.
Interpreting the eigenvalues

Classification of image points using eigenvalues of $M$:

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

- **“Corner”**
  - $\lambda_1$ and $\lambda_2$ are large,
  - $\lambda_1 \sim \lambda_2$;
  - $E$ increases in all directions

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

- $\lambda_1$ and $\lambda_2$ are small; $E$ is almost constant in all directions
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: Steps

1. Compute Gaussian derivatives at each pixel
2. Compute second moment matrix $M$ in a Gaussian window around each pixel
3. Compute corner response function $R$
4. Threshold $R$
5. Find local maxima of response function (nonmaximum suppression)

Harris Detector: Workflow
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: $I \rightarrow a \ I$

![Graph showing Harris detector properties](image)
Harris Detector: Some Properties

But: non-invariant to *image scale*!

All points will be classified as *edges*

Corner!
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 independently in each image?

Choose the scale of the “best” corner
Feature selection

Distribute points evenly over the image
Adaptive Non-maximal Suppression

Desired: Fixed # of features per image
- Want evenly distributed spatially…
- Sort points by non-maximal suppression radius

[Brown, Szeliski, Winder, CVPR’05]