Live Homography…

https://www.youtube.com/watch?v=QTOxOlj1q28

Also see OpenPhoto VR: http://openphotovr.org/
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?

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

for \(a = a0:astep:a1,\)
for \(b = b0:bstep:b1,\)
for \(c = c0:cstep:c1,\)
for \(d = d0:dstep:d1,\)
for \(e = e0:estep:e1,\)
for \(f = f0:fstep:f1,\)
for \(g = g0:gstep:g1,\)
for \(h = h0:hstep:h1,\)
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 $O(N^4)$:
  \[ H = K_1 R_1 R_2^{-1} K_2^{-1} \]  
  (4 DOF: f and rotation)

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 #6 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!
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

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
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
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
Harris Detector: Mathematics

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 \(w(x, y) = \)

1 in window, 0 outside

or

Gaussian
Harris Detector: Mathematics

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

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

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

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

\[
A^T A = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I(\nabla I)^T
\]
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**
  - $\lambda_1 \gg \lambda_2$

- **Flat** region
  - $\lambda_1$ and $\lambda_2$ are small;
  - $E$ is almost constant in all directions
Measure of corner response:

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

\[ \det M = \lambda_1 \lambda_2 \]

\[ \text{trace } M = \lambda_1 + \lambda_2 \]
Harris Detector

The Algorithm:

- Find points with large corner response function $R$ ($R > \text{threshold}$)
- Take the points of local maxima of $R$
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 => invariance to intensity shift $I \rightarrow I + b$
- ✓ Intensity scale: $I \rightarrow a \, I$

![Diagram](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.
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]
Feature descriptors

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

Point descriptor should be:
1. Invariant
2. Distinctive
Feature Descriptor – MOPS
Multi-Scale Oriented Patches

Interest points
- Multi-scale Harris corners
- Orientation from blurred gradient
- Geometrically invariant to rotation

Descriptor vector
- Bias/gain normalized sampling of local patch (8x8)
- Photometrically invariant to affine changes in intensity

[Brown, Szeliski, Winder, CVPR’2005]
Detect Features, setup Frame

Orientation = blurred gradient

Rotation Invariant Frame

- Scale-space position \( (x, y, s) \) + orientation \( (\theta) \)
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.
MOPS descriptor vector

8x8 oriented patch
  - Sampled at 5 x scale

Bias/gain normalisation: $l' = (l - \mu)/\sigma$
Automatic Feature Matching
Feature matching
Feature matching

• Pick best match!
  • For every patch in image 1, find the most similar patch (e.g. by SSD).
  • Called “nearest neighbor” in machine learning

• Can do various speed ups:
  • Hashing
    – compute a short descriptor from each feature vector, or hash longer descriptors (randomly)
  • Fast Nearest neighbor techniques
    – $kd$-trees and their variants
  • Clustering / Vector quantization
    – So called “visual words”
What about outliers?
Feature-space outlier rejection

Let’s not match all features, but only those that have “similar enough” matches?

How can we do it?

- \( \text{SSD}(\text{patch1}, \text{patch2}) < \text{threshold} \)
- How to set threshold?

![Graph showing probability density against 1-NN squared error with correct matches and incorrect matches distinguished.](image-url)
Feature-space outlier rejection

Let’s not match all features, but only these that have “similar enough” matches?

How can we do it?

• Symmetry: x’s NN is y, and y’s NN is x
Feature-space outlier rejection

A better way [Lowe, 1999]:

- **1-NN**: SSD of the closest match
- **2-NN**: SSD of the second-closest match
- Look at how much better 1-NN is than 2-NN, e.g. $1$-NN/$2$-NN
- That is, is our best match so much better than the rest?

![Graph showing the comparison between correct matches and incorrect matches based on 1-NN/2-NN squared error.](image)
Can we now compute H from the blue points?

- No! Still too many outliers…
- What can we do?
Matching features

What do we do about the “bad” matches?
RAnandom SAmple Consensus

Select *one* match, count *inliers*
Select one match, count inliers
Least squares fit

Find “average” translation vector
RANSAC for estimating homography

RANSAC loop:
1. Select four feature pairs (at random)
2. Compute homography $H$ (exact)
3. Compute *inliers* where $SSD(p_i', H p_i) < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares $H$ estimate on all of the inliers
RANSAC
Example: Recognising Panoramas

M. Brown and D. Lowe,
University of British Columbia
Why “Recognising Panoramas”?
Why “Recognising Panoramas”?

1D Rotations ($\theta$)

- Ordering $\Rightarrow$ matching images
Why “Recognising Panoramas”?

1D Rotations ($\theta$)

- Ordering $\Rightarrow$ matching images
Why “Recognising Panoramas”?

1D Rotations ($\theta$)
  - Ordering $\Rightarrow$ matching images
Why “Recognising Panoramas”?

1D Rotations ($\theta$)
  - Ordering $\Rightarrow$ matching images

2D Rotations ($\theta, \phi$)
  - Ordering $\nRightarrow$ matching images
Why “Recognising Panoramas”?  

1D Rotations ($\theta$)
- Ordering $\Rightarrow$ matching images

2D Rotations ($\theta$, $\phi$)
- Ordering $\nRightarrow$ matching images
Why “Recognising Panoramas”?

1D Rotations ($\theta$)
- Ordering $\Rightarrow$ matching images

2D Rotations ($\theta$, $\phi$)
- Ordering $\not\Rightarrow$ matching images
Why “Recognising Panoramas”? 
Overview

Feature Matching
Image Matching
Bundle Adjustment
Multi-band Blending
Results
Conclusions
RANSAC for Homography
RANSAC for Homography
RANSAC for Homography
Probabilistic model for verification
Finding the panoramas
Finding the panoramas
Finding the panoramas
Finding the panoramas
Homography for Rotation

Parameterise each camera by rotation and focal length

\[ R_i = e^{[\theta_i]_{\times}}, \quad [\theta_i]_{\times} = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix} \]

\[ K_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix} \]

This gives pairwise homographies

\[ \tilde{u}_i = H_{ij}\tilde{u}_j, \quad H_{ij} = K_i R_i R_j^T K_j^{-1} \]
Bundle Adjustment

New images initialised with rotation, focal length of best matching image
Bundle Adjustment

New images initialised with rotation, focal length of best matching image
Multi-band Blending

Burt & Adelson 1983

• Blend frequency bands over range $\propto \lambda$
Results