BBM 444 – Week 9
Panoramas & Mosaics

PIA16453: Panoramic View From 'Rocknest' Position of Curiosity Mars Rover
http://photojournal.jpl.nasa.gov/catalog/PIA16453

Aykut Erdem
Computer Vision Lab (CVL)
Hacettepe University

Slides from Steve Seitz, Rick Szeliski, Alexei Efros, Fredo Durand, Chuck Dyer, and Kristin Grauman and James Hays
Why Mosaic?

Are you getting the whole picture?

- Compact Camera FOV = 50 x 35°
Why Mosaic?

Are you getting the whole picture?

- Compact Camera FOV = 50 x 35°
- Human FOV = 200 x 135°
Why Mosaic?

Are you getting the whole picture?

- Compact Camera FOV = 50 x 35°
- Human FOV = 200 x 135°
- Panoramic Mosaic = 360 x 180°
The First Panoramas ...

Paris, c. 1845-50, photographer unknown

San Francisco from Rincon Hill, 1851, by Martin Behrmanx
... and Panoramic Cameras

The “AL-VISTA”
Panoramic Camera

FIVE Cameras in ONE for the Price of ONE.

The AL-VISTA

Sweeps the Field.

Sweeps the Field.

THE AL-VISTA

THE AL-VISTA

THE TRAVELLING LENS

DOES IT.

THE AL-VISTA PANORAMIC CAMERA

THE AL-VISTA PANORAMIC CAMERA

MULTISCOPE & FILM CO.

BURLINGTON, WIS., U.S.A.

23 JEFFERSON STREET.

Al-Vista, 1899 ($20)
Old panoramas in a modern viewer

Figure 2. The principle of an infinite rotation presentation of a circular panorama (circular room or a computer screen and panorama viewer).

http://www.eurofresh.se/history/
Traditional panoramas

Swing lens (1843 – 1980s)
Panorama Capture Hardware

0-360

Point Grey Ladybug

Panoscan MK-3
Kogeto Dot 360 Camera for iPhone
Mosaics: stitching images together
Today’s Agenda

• Manual correspondences
  – The user provides 4 correspondences
  – We *reproject* one image to match the other one
  – Creates a wider angle view
Today’s Agenda

• **Automatic correspondences**
  – Corner detection
  – Patch descriptor

• **Nice blending**
  – Smooth transition
  – 2 scale
Single view model

- Camera rotates around a single optical center
A pencil of rays contains all views

Can generate any synthetic camera view as long as it has the same center of projection!
How to do it?

Basic Procedure

• Take a sequence of images from the same position
  – Rotate the camera about its optical center
• Compute transformation between second image and first
• Transform the second image to overlap with the first
• Blend the two together to create a mosaic
• If there are more images, repeat

…but wait, why should this work at all?

• What about the 3D geometry of the scene?
• Why aren’t we using it?
Aligning images: Translation

left on top

right on top

Translations are not enough to align the images
The mosaic has a natural interpretation in 3D

- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a synthetic wide-angle camera
Image reprojection

Basic question
- How to relate two images from the same camera center?
  - how to map a pixel from PP1 to PP2

Answer
- Cast a ray through each pixel in PP1
- Draw the pixel where that ray intersects PP2

But don’t we need to know the geometry of the two planes with respect to the eye?

Observation:
Rather than thinking of this as a 3D reprojection, think of it as a 2D **image warp** from one image to another
Aligning images

- We have established that pairs of images from the same viewpoint can be aligned through a simple 2D spatial transformation (warp).
- What kind of transformation?
Image warping with homographies

image plane in front

black area where no pixel maps to
To unwarp (rectify) an image

- Find the homography $H$ given a set of $p$ and $p'$ pairs
- How many correspondences are needed?
- Tricky to write $H$ analytically, but we can solve for it!
  - Find such $H$ that “best” transforms points $p$ into $p'$
  - Use least-squares!
Example
Recall: Simple Perspective Projection

- Project all points to the $z = 1$ plane, viewpoint at the origin:
  - $x' = x/z$
  - $y' = y/z$

- Can we represent this with a matrix?
  - not directly (division)
  - add a third coordinate to the result (homogeneous coordinates)
    - $(x, y, w)$ represents $(x/w, y/w)$
    - allows us to represent projective transforms
    - Nice thing: projecting onto plane $z=1$ is just the strict interpretation of homogeneous coordinates
Homography

- Projective – mapping between any two PPs with the same center of projection
  - rectangle should map to arbitrary quadrilateral
  - parallel lines aren’t
  - but must preserve straight lines
  - same as: project, rotate, reproject

called **Homography**

\[
\begin{bmatrix}
wx' \\
w'y' \\
w
\end{bmatrix} = \begin{bmatrix}
* & * & * \\
* & * & * \\
* & * & * \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
l
\end{bmatrix}
\]

To apply a homography \( H \)

- Compute \( p' = Hp \) (regular matrix multiply)
- Convert \( p' \) from homogeneous to image coordinates
To apply a given homography $H$

- Compute $p' = Hp$ (regular matrix multiply)
- Convert $p'$ from homogeneous to image coordinates

$$\begin{bmatrix} wx' \\ wy' \\ w \\ p' \end{bmatrix} = \begin{bmatrix} * & * & * & 0 \\ * & * & * & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \\ p \end{bmatrix}$$
To **compute** the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of $H$ are the unknowns...
Homography equation

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

- We are given pairs of corresponding points
  - \( x, y, x', y' \) are known
- Unknowns: matrix coefficients and \( w' \)
Homography equation

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

- We are given pairs of corresponding points
  - \(x, y, x', y'\) are known
- Unknowns: matrix coefficients and \(w'\)
  - But \(w'\) is easy to get:

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

\[w' = gy + hx + i\]

- For a pair of points \((x, y) \rightarrow (x', y')\) we have
  \[
  ay + bx + c = y' (gy + hx + i)
  \]
  \[
  dy + ex + f = x' (gy + hx + i)
  \]
\[
\begin{pmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i \\
\end{pmatrix}
\begin{pmatrix}
  y \\
  x \\
  1 \\
\end{pmatrix}
=
\begin{pmatrix}
  y'w' \\
  x'w' \\
  w' \\
\end{pmatrix}
\]

\[
w' = gy + hx + i
\]

• For a pair of points \((x, y) \rightarrow (x', y')\) we have

\[
ay + bx + c = y'(gy + hx + i)
\]

\[
dy + ex + f = x'(gy + hx + i)
\]

• Unknowns: \(a, b, c, d, e, f, g, h, i\)
  • Linear!
How many pairs?

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

• Each correspondence pair gives us two equations

\[
ay + bx + c = y'(gy + hx + i)
\]
\[
dy + ex + f = x'(gy + hx + i)
\]

• How many unknowns?
  - 9
  - But H is defined up to scale. Four pairs are enough!
Forming the linear system

- We have 4x2 linear equations in our 8 unknowns
- Represent as a matrix system $Ax = B$:

\[
\begin{pmatrix}
a & b \\
 c & d \\
 e & f \\
g & h \\
i
\end{pmatrix}
= B
\]

- Now we need to fill matrix $A$ and vector $B$
Forming the matrix

\[ ay + bx + c = y'(gy + hx + i) \]

\[ dy + ex + f = x'(gy + hx + i) \]

\[
\begin{pmatrix}
a & b & c & d & e & f & g & h & i \\
\end{pmatrix}
\begin{pmatrix}
a \\
b \\
c \\
d \\
e \\
f \\
g \\
h \\
i \\
\end{pmatrix}
= 
\begin{pmatrix}
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\cdot \\
\end{pmatrix}
\]
Forming the matrix

\[ ay + bx + c = y' (gy + hx + i) \]
\[ dy + ex + f = x' (gy + hx + i) \]

\[
\begin{pmatrix}
\begin{bmatrix}
 a & b & c & d & e & f & g & h & i \\
\end{bmatrix} \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\end{pmatrix}
\begin{pmatrix}
\begin{bmatrix}
 y & x & 1 & 0 & 0 & 0 & -yy' & -xy' & -y' \\
\end{bmatrix} \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\end{pmatrix}
= 0
\]
Recap: Solving for homographies

• We have four pairs of points

• Looking for homography H

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

• Formed a big 8x9 linear system Ax=0
  - where x is the 9 homography coefficients
Dirty Solution

- Can set scale factor \( i = 1 \). So, there are 8 unknowns.
- Set up a system of linear equations:
  \[
  Ax = b
  \]
  where vector of unknowns \( h = [a, b, c, d, e, f, g, h]^T \)
- Need at least 8 eqs, but the more the better...
- Solve for \( h \). If overconstrained, solve using least-squares:
  \[
  \min \|Ah - b\|^2
  \]

>> help \texttt{lmdivide}

- A cleaner solution is to use SVD
  - The singular vector with singular value 0 is a solution

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

- Collect correspondences (manually)
- Solve for homography matrix H
  - Least squares solution
- Warp content from one image frame to the other to combine: say im1 into im2 reference frame
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2’s coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1.
  - Compute coordinates in im1’s reference frame (via homography) for all points in that range: \( H^{-1} \)
  - Lookup all colors for all these positions from im1
    - Inverse warp: \texttt{interp2} (watch for nans: \texttt{isnan})
- Overlay im2 content onto the warped im1 content.
  - Careful about new bounds of the output image: minx, miny
Mosaics: main steps

• Collect correspondences (manually)
• Solve for homography matrix $H$
  – Least squares solution
• Warp content from one image frame to the other to combine: say $im1$ into $im2$ reference frame
  – Determine bounds of the new combined image
    • Where will the corners of $im1$ fall in $im2$’s coordinate frame?
    • We will attempt to lookup colors for any of these positions we can get from $im1$. \( \text{meshgrid} \)
  – Compute coordinates in $im1$’s reference frame (via homography) for all points in that range: $H^{-1}$
  – Lookup all colors for all these positions from $im1$
    • Inverse warp: \( \text{interp2} \) (watch for nans: \( \text{isnan} \))
• Overlay $im2$ content onto the warped $im1$ content.
  – Careful about new bounds of the output image: minx, miny
Mosaics: main steps

• Collect correspondences (manually)
• Solve for homography matrix H
  – Least squares solution
• Warp content from one image frame to the other to combine: say im1 into im2 reference frame
  – Determine bounds of the new combined image
    • Where will the corners of im1 fall in im2’s coordinate frame?
    • We will attempt to lookup colors for any of these positions we can get from im1.
    – Compute coordinates in im1’s reference frame (via homography) for all points in that range: H\(^{-1}\)
    – Lookup all colors for all these positions from im1
      • Inverse warp: interp2 (watch for nans: isnan)
• Overlay im2 content onto the warped im1 content.
  – Careful about new bounds of the output image: minx, miny
Use `interp2` to ask for the colors (possibly interpolated) from `im1` at all the positions needed in `im2`’s reference frame.
Mosaics: main steps

- Collect correspondences (manually)
- Solve for homography matrix $H$
  - Least squares solution
- **Warp content from one image frame to the other to combine: say im1 into im2 reference frame**
  - Determine bounds of the new combined image
    - Where will the corners of im1 fall in im2’s coordinate frame?
    - We will attempt to lookup colors for any of these positions we can get from im1. : `meshgrid`
  - Compute coordinates in im1’s reference frame (via homography) for all points in that range: $H^{-1}$
  - Lookup all colors for all these positions from im1
    - Inverse warp : `interp2` (watch for nans : `isnan`)
- **Overlay im2 content onto the warped im1 content.**
  - Careful about new bounds of the output image: minx, miny
Fun with homographies

Original image

Virtual camera rotations

St. Petersburg
photo by A. Tikhonov
Analyzing patterns and shapes

What is the shape of the b/w floor pattern?

The floor (enlarged)

Automatically rectified floor

Slide from Criminisi
Analyzing patterns and shapes

From Martin Kemp *The Science of Art* (manual reconstruction)

2 patterns have been discovered!

Slide from Criminisi
Analyzing patterns and shapes

What is the (complicated) shape of the floor pattern?

Automatically rectified floor

St. Lucy Altarpiece, D. Veneziano
Slide from Criminisi
Analyzing patterns and shapes

From Martin Kemp, *The Science of Art* (manual reconstruction)

Automatic rectification
Analyzing patterns and shapes

The Ambassadors by Hans Holbein the Younger, 1533
Julian Beever: Manual Homographies

http://users.skynet.be/J.Beever/pave.htm
Recap

• Panorama = reprojection
• 3D rotation -> homography
  - Homogeneous coordinates are key
• Use feature correspondence
• Solve least square problem
  - Set of linear equations
• Warp all images to a reference one
• Use your favorite blending
Changing camera centers

3-D Scene

Rotation + translation

\[ p = \begin{bmatrix} X \\ Y \\ Z \\ d \end{bmatrix} \]
General projective model

Image Sequence:

1. Image
2. Image
3. Image
changing camera center

Does it still work?

PP1

PP2

synthetic PP
Planar scene (or far away)

PP3 is a projection plane of both centers of projection, so we are OK!
This is how big aerial photographs are made
<table>
<thead>
<tr>
<th>Image Compositing for Tele-Reality</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
</tr>
<tr>
<td>2. Previous work</td>
</tr>
<tr>
<td>3. Basic imaging equations</td>
</tr>
<tr>
<td>4. Planar image compositing</td>
</tr>
<tr>
<td>5. Panoramic</td>
</tr>
<tr>
<td>6. Piecewise-planar scenes</td>
</tr>
<tr>
<td>7. Scenes with arbitrary depth</td>
</tr>
<tr>
<td>8. 3-D model recovery</td>
</tr>
<tr>
<td>9. Applications</td>
</tr>
<tr>
<td>10. Discussion &amp; Conclusions</td>
</tr>
</tbody>
</table>

**Planar mosaic**
Live Homography DEMO

Check out panoramio.com “Look Around” feature!

Also see OpenPhoto VR:  http://openphotovr.org/ and Mars Gigapixel Panorama - Curiosity rover: Martian solar days 136-149 http://www.360cities.net/image/mars-gigapixel-panorama-curiosity-solar-days-136-149#110.16,3.12,36.0
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

- 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 \(a=a_0:a\text{step}:a_1,\)
  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 \text{image1} to \(H(\text{image2})\)
            \end{for}
          \end{for}
        \end{for}
      \end{for}
    \end{for}
  \end{for}
\end{for}
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 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

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
Local features: main components

1) **Detection:** Identify the interest points

2) **Description:** Extract vector feature descriptor surrounding each interest point.

3) **Matching:** Determine correspondence between descriptors in two views

\[ \mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}] \]

\[ \mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}] \]

Slide credit: K. Grauman
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
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
Harris Detector: Basic Idea

“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 \([u, v]\):

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

Window function

Shifted intensity

Intensity

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

1 in window, 0 outside

or

Gaussian
Harris Detector: Mathematics

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

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

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

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

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

Classification of image points using eigenvalues of $M$:

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

But eigenvalues are expensive to compute.
Harris Detector: Mathematics

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: $I \rightarrow a \cdot I$

![Diagram illustrating threshold and image coordinate](image.png)
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 \textit{independently} in each image?

Choose the scale of the “best” corner
Automatic scale selection

Intuition:
• Find scale that gives local maxima of some function $f$ in both position and scale.
Harris Detector – Responses
Automatic Scale Selection

How to find corresponding patch sizes?

K. Grauman, B. Leibe
Automatic Scale Selection

Function responses for increasing scale (scale signature)

\[ f(I_{i_1...i_m}(x, \sigma)) \quad \text{K. Grauman, B. Leibe} \]

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

Function responses for increasing scale (scale signature)

\[ f(I_{i_1 \ldots i_m}(x, \sigma)) \quad \text{K. Grauman, B. Leibe} \]

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

Function responses for increasing scale (scale signature)

$f(I_{i_1...i_m}(x, \sigma))$  
K. Grauman, B. Leibe  

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

Function responses for increasing scale (scale signature)

\[ f(I_{s_1 \ldots s_m}(x, \sigma)) \]

K. Grauman, B. Leibe

\[ f(I_{s_1 \ldots s_m}(x', \sigma)) \]
Automatic Scale Selection

Function responses for increasing scale (scale signature)

\[ f(I_{i_1 \ldots i_m}(x, \sigma)) \quad \text{K. Grauman, B. Leibe} \quad f(I_{i_1 \ldots i_m}(x', \sigma)) \]
Automatic Scale Selection

Function responses for increasing scale (scale signature)

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

K. Grauman, B. Leibe
What Is A Useful Signature Function?

Difference of Gaussian = “blob” detector
DoG – Efficient Computation

Computation in Gaussian scale pyramid

Sampling with step $s^4 = 2$

Original image

$\sigma = 2^4$

Scale (next octave)

Scale (first octave)

Gaussian

Difference of Gaussian (DOG)

K. Grauman, B. Leibe
Results: Lowe’s DoG

K. Grauman, B. Leibe
Making descriptor rotation invariant

• Rotate patch according to its dominant gradient orientation
• This puts the patches into a canonical orientation.
Descriptor Vector

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: \( I' = (I - \mu)/\sigma \)
Feature descriptors

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

Point descriptor should be:
1. Invariant
2. Distinctive
Feature matching

• Exhaustive search
  • for each feature in one image, look at all the other features in the other image(s)

• Hashing
  • compute a short descriptor from each feature vector, or hash longer descriptors (randomly)

• Fast Nearest neighbor techniques
  • $kd$-trees and their variants
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?

- \( \text{SSD}(\text{patch}_1, \text{patch}_2) < \text{threshold} \)
- How to set threshold?

```plaintext
Status
We have extracted \( N_1 \) corners from image 1, and \( N_2 \) from image 2
For each corner, we have a \( k \times k \) descriptor
The combination of a corner+descriptor is often called a feature point
Now we need to match feature points from image 1 to image 2
```
Problem

We have a match for each corner of an image
- But lots of wrong matches
- Even scene points that are not on the overlap between the images have a match!
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 probability density against 1-NN/2-NN squared error]
Can we now compute $H$ from the blue points?

- No! Still too many outliers…
- What can we do?
**RA**random **SA**mple **C**onsensus

- A general framework for model fitting in the presence of outliers

**Outline**
- Choose a small subset of points uniformly at random
- Fit a model to that subset
- Find all remaining points that are “close” to the model and reject the rest as outliers
- Do this many times and choose the best model

---

A new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data is introduced. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and is thus ideally suited for applications in automated image analysis where interpretation is based on the data provided by error-prone feature detectors. A major portion of this paper describes the application of RANSAC to the Location Determination Problem (LDP): Given an image depicting a set of landmarks with known locations, determine that point in space from which the image was obtained. In response to a RANSAC requirement, new results are derived on the minimum number of landmarks needed to obtain a solution, and algorithms are presented for computing these minimum-landmark solutions in closed form. These results provide the basis for an automatic system that can solve the LDP under difficult viewing conditions.
RANSAC for line fitting

• Repeat $N$ times:
• Draw $s$ points uniformly at random
• Fit line to these $s$ points
• Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than $t$)
• If there are $d$ or more inliers, accept the line and refit using all inliers
RANSAC for line fitting example
RANSAC for line fitting example
RANSAC for line fitting example

1. Randomly select minimal subset of points

Slide credit: R. Raguram
1. Randomly select minimal subset of points
2. Hypothesize a model

RANSAC for line fitting example
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Uncontaminated sample
1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop
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
Example: Autostitch


- **Goal**: Search a collection of photos for sets that can be stitched together completely automatically

http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html
Autostitch: Example

Input:

Output:

[Brown 2003]
Method

- Detect point features
- Match features between images
- Determine overlapping pairs of images
- Solve for homographies between all images
- Blend
Why “Recognising Panoramas”? 

1D Rotations (q) 

- Ordering ⇒ matching images
Why “Recognising Panoramas”?

1D Rotations (q)
• Ordering ⇒ matching images
Why “Recognising Panoramas”?  

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

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

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

1D Rotations (q)

• Ordering ⇒ matching images

2D Rotations (q, f)

– Ordering ⇒≠ matching images
Why “Recognising Panoramas”?

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

2D Rotations (q, f)
- 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 (=cliques)
Finding the panoramas
Finding the panoramas
Homography for Rotation

Parameterize 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 $\propto l$
Results
Reading Assignments


Figure 1: A multi-viewpoint panorama of a street in Antwerp composed from 107 photographs taken about one meter apart with a hand-held camera.

Figure 1: Working example. A user makes a query of ”West Lake, Hangzhou” to YouTube, and feeds retrieved video clips into our system. Our system selects useful frames from the given videos and synthesizes panoramas using the selected source frames.