Image Stitching
• Combine two or more overlapping images to make one larger image
How to do it?

• Basic Procedure
  1. Take a sequence of images from the same position
     1. Rotate the camera about its optical center
  2. Compute transformation between second image and first
  3. Shift the second image to overlap with the first
  4. Blend the two together to create a mosaic
  5. If there are more images, repeat
1. Take a sequence of images from the same position

   • Rotate the camera about its optical center
2. Compute transformation between images

- Extract interest points
- Find Matches (are they all correct?)
- Compute transformation?
3. Shift the images to overlap
4. Blend the two together to create a mosaic
5. Repeat for all images
How to do it?

• Basic Procedure

✓ 1. Take a sequence of images from the same position
   1. Rotate the camera about its optical center
  2. Compute transformation between second image and first
  3. Shift the second image to overlap with the first
  4. Blend the two together to create a mosaic
  5. If there are more images, repeat
Compute Transformations

✓ • Extract interest points
✓ • Find good matches
• Compute transformation

Let’s assume we are given a set of good matching interest points
Image reprojection

• The mosaic has a natural interpretation in 3D
  – The images are reprojected onto a common plane
  – The mosaic is formed on this plane
Example
• Observation
  – Rather than thinking of this as a 3D reprojection, think of it as a 2D image warp from one image to another
Motion models

• What happens when we take two images with a camera and try to align them?
  • translation?
  • rotation?
  • scale?
  • affine?
  • Perspective?
Projective transformations

• (aka *homographies*)

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

\[x' = \frac{u}{w}\]
\[y' = \frac{v}{w}\]
Parametric (global) warping

- Examples of parametric warps:
  - translation
  - rotation
  - aspect
  - affine
  - perspective
2D coordinate transformations

• translation: \( x' = x + t \) \( x = (x,y) \)
• add rotation: \( x' = R \cdot x + t \)
• similarity: \( x' = s \cdot R \cdot x + t \)
• affine: \( x' = A \cdot x + t \)
• perspective: \( x' \approx H \cdot x \) \( x = (x,y,1) \)  
  \( (x) \) is in *homogeneous* coordinates)
Image Warping

• Given a coordinate transform \( x' = h(x) \) and a source image \( f(x) \), how do we compute a transformed image \( g(x') = f(h(x)) \)?
Forward Warping

• Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$
• What if pixel lands “between” two pixels?
Forward Warping

• Send each pixel $f(x)$ to its corresponding location $x' = h(x)$ in $g(x')$

• What if pixel lands “between” two pixels?
• Answer: add “contribution” to several pixels, normalize later (splatting)
Inverse Warping

• Get each pixel $g(x')$ from its corresponding location $x' = h(x)$ in $f(x)$

• What if pixel comes from “between” two pixels?
Inverse Warping

• Get each pixel \( g(x') \) from its corresponding location \( x' = h(x) \) in \( f(x) \)

• What if pixel comes from "between" two pixels?
  • Answer: resample color value from interpolated source image
Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)
Related: Descriptors
Implementation Concern: How do you rotate a patch?

• Start with an “empty” patch whose dominant direction is “up”.
• For each pixel in your patch, compute the position in the detected image patch. It will be in floating point and will fall between the image pixels.
• Interpolate the values of the 4 closest pixels in the image, to get a value for the pixel in your patch.
Using Bilinear Interpolation

- Use all 4 adjacent pixels, weighted.
Transformation models

Translation: 2 unknowns
Affine: 6 unknowns
Perspective: 8 unknowns
Finding the transformation

• Translation = 2 degrees of freedom
• Similarity = 4 degrees of freedom
• Affine = 6 degrees of freedom
• Homography = 8 degrees of freedom

• How many corresponding points do we need to solve?
Simple case: translations

How do we solve for $(x_t, y_t)$?
Simple case: translations

Displacement of match $i = (x'_i - x_i, y'_i - y_i)$

$$(x_t, y_t) = \left( \frac{1}{n} \sum_{i=1}^{n} x'_i - x_i, \frac{1}{n} \sum_{i=1}^{n} y'_i - y_i \right)$$
Simple case: translations

- System of linear equations
  - What are the knowns? Unknowns?
  - How many unknowns? How many equations (per match)?

\[
x_i + x_t = x'_i
\]
\[
y_i + y_t = y'_i
\]
Simple case: translations

- Problem: more equations than unknowns
  - “Overdetermined” system of equations
  - We will find the least squares solution

\[
\begin{align*}
    x_i + x_t &= x_i' \\
    y_i + y_t &= y_i'
\end{align*}
\]
Least squares formulation

- For each point \((x_i, y_i)\)
  \[
x_i + x_t = x'_i
  \]
  \[
y_i + y_t = y'_i
  \]

- We define the *residuals* as
  \[
r_{x_i}(x_t) = (x_i + x_t) - x'_i
  \]
  \[
r_{y_i}(y_t) = (y_i + y_t) - y'_i
  \]
Least squares formulation

- Goal: minimize sum of squared residuals

\[ C(x_t, y_t) = \sum_{i=1}^{n} \left( r_{x_i}(x_t)^2 + r_{y_i}(y_t)^2 \right) \]

- “Least squares” solution

- For translations, is equal to mean displacement
Least squares

\[ A^t b = b \]

• Find \( t \) that minimizes

\[ ||A^t b - b||^2 \]

• To solve, form the normal equations

\[ A^T A^t = A^T b \]

\[ t = (A^T A)^{-1} A^T b \]
Solving for translations

• Using least squares

\[
\begin{bmatrix}
1 & 0 \\
0 & 1 \\
1 & 0 \\
0 & 1 \\
\vdots \\
1 & 0 \\
0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x_t \\
y_t \\
\end{bmatrix}
= 
\begin{bmatrix}
x'_1 - x_1 \\
y'_1 - y_1 \\
x'_2 - x_2 \\
y'_2 - y_2 \\
\vdots \\
x'_n - x_n \\
y'_n - y_n \\
\end{bmatrix}
\]
Affine transformations

\[
\begin{bmatrix}
    x' \\
    y' \\
    1
\end{bmatrix}
= 
\begin{bmatrix}
    a & b & c \\
    d & e & f \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\]

- How many unknowns?
- How many equations per match?
- How many matches do we need?
Affine transformations

- Residuals:
  \[ r_{x_i}(a, b, c, d, e, f) = (ax_i + by_i + c) - x_i' \]
  \[ r_{y_i}(a, b, c, d, e, f) = (dx_i + ey_i + f) - y_i' \]

- Cost function:
  \[ C(a, b, c, d, e, f) = \sum_{i=1}^{n} (r_{x_i}(a, b, c, d, e, f)^2 + r_{y_i}(a, b, c, d, e, f)^2) \]
Affine transformations

• Matrix form

\[
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_1 & y_1 & 1 \\
  x_2 & y_2 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_2 & y_2 & 1 \\
  \vdots
  \end{bmatrix}
  \begin{bmatrix}
    a \\
    b \\
    c \\
    d \\
    e \\
    f
  \end{bmatrix}
  =
  \begin{bmatrix}
    x'_1 \\
    y'_1 \\
    x'_2 \\
    y'_2 \\
    \vdots
    \end{bmatrix}
\]

\[
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_2 & y_2 & 1 \\
  x_n & y_n & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & x_n & y_n & 1
\end{bmatrix}
\begin{bmatrix}
  a \\
  b \\
  c \\
  d \\
  e \\
  f
\end{bmatrix}
= \begin{bmatrix}
  x'_1 \\
  y'_1 \\
  x'_2 \\
  y'_2 \\
  \vdots
\end{bmatrix}
\]

\[
A_{2n \times 6} \quad t_{6 \times 1} = b_{2n \times 1}
\]
Solving for homographies

\[
\begin{bmatrix}
x'_{i} \\
y'_{i} \\
1
\end{bmatrix} = 
\begin{bmatrix}
h_{00} & h_{01} & h_{02} \\
h_{10} & h_{11} & h_{12} \\
h_{20} & h_{21} & h_{22}
\end{bmatrix}
\begin{bmatrix}
x_{i} \\
y_{i} \\
1
\end{bmatrix}
\]

\[
x'_{i} = \frac{h_{00}x_{i} + h_{01}y_{i} + h_{02}}{h_{20}x_{i} + h_{21}y_{i} + h_{22}}
\]

\[
y'_{i} = \frac{h_{10}x_{i} + h_{11}y_{i} + h_{12}}{h_{20}x_{i} + h_{21}y_{i} + h_{22}}
\]

\[
x'_{i}(h_{20}x_{i} + h_{21}y_{i} + h_{22}) = h_{00}x_{i} + h_{01}y_{i} + h_{02}
\]

\[
y'_{i}(h_{20}x_{i} + h_{21}y_{i} + h_{22}) = h_{10}x_{i} + h_{11}y_{i} + h_{12}
\]
Solving for homographies

\[
x'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02} \\
y'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}
\]

\[
\begin{bmatrix}
x_i & y_i & 1 & 0 & 0 & 0 & -x'_ix_i & -x'_iy_i & -x'_i \\
0 & 0 & 0 & x_i & y_i & 1 & -y'_ix_i & -y'_iy_i & -y'_i
\end{bmatrix}
\begin{bmatrix}
h_{00} \\
h_{01} \\
h_{02} \\
h_{10} \\
h_{11} \\
h_{12} \\
h_{20} \\
h_{21} \\
h_{22}
\end{bmatrix}
= \begin{bmatrix}
0 \\
0
\end{bmatrix}
\]
Direct Linear Transforms

\[
\begin{bmatrix}
  x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 & x_1 & -x'_1 y_1 & -x'_1 \\
  0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 & x_1 & -y'_1 y_1 & -y'_1 \\
  \vdots \\
  x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\
  0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \\
\end{bmatrix}
\begin{bmatrix}
  h_{00} \\
  h_{01} \\
  h_{02} \\
  h_{10} \\
  h_{11} \\
  h_{12} \\
  h_{20} \\
  h_{21} \\
  h_{22}
\end{bmatrix}
= \begin{bmatrix}
  0 \\
  0 \\
  \vdots \\
  0 \\
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
  h & 0 \\
\end{bmatrix}
\]

\[
A = \begin{bmatrix}
  h & 0 \\
\end{bmatrix}
\]

Defines a least squares problem:

\[
\text{minimize} \quad \|Ah - 0\|^2
\]

- Since \(h\) is only defined up to scale, solve for unit vector \(\hat{h}\)
- Solution: \(\hat{h}\) = eigenvector of \(A^T A\) with smallest eigenvalue
- Works with 4 or more points
Matching features

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

Select *one* match, count *inliers*
RAndom SAmple Consensus

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 $\|p_i', Hp_i\| < \varepsilon$

• Keep largest set of inliers
• Re-compute least-squares $H$ estimate using all of the inliers
Simple example: fit a line

• Rather than homography $H$ (8 numbers)
  fit $y=ax+b$ (2 numbers $a$, $b$) to 2D pairs
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers

4 inliers
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers

9 inliers
Simple example: fit a line

• Pick 2 points
• Fit line
• Count inliers
Simple example: fit a line

- Use biggest set of inliers
- Do least-square fit
Red:
  rejected by 2nd nearest neighbor criterion
Blue:
  Ransac outliers
Yellow:
  inliers
Computing homography

- Assume we have four matched points: How do we compute homography $H$?

Normalized DLT (direct linear transformation)

1. Normalize coordinates for each image
   a) Translate for zero mean
   b) Scale so that average distance to origin is $\sim$sqrt(2)
      \[
      \tilde{x} = Tx \quad \tilde{x}' = T'x'
      \]
      – This makes problem better behaved numerically

2. Compute $\tilde{H}$ using DLT in normalized coordinates

3. Unnormalize:
   \[
   H = T'^{-1}\tilde{H}T
   \]
   \[
   x_i' = Hx_i
   \]
Computing homography

• Assume we have matched points with outliers: How do we compute homography $H$?

Automatic Homography Estimation with RANSAC
1. Choose number of samples $N$
2. Choose 4 random potential matches
3. Compute $H$ using normalized DLT
4. Project points from $x$ to $x'$ for each potentially matching pair: $x'_i = Hx_i$
5. Count points with projected distance $< t$
   – E.g., $t = 3$ pixels
6. Repeat steps 2-5 $N$ times
   – Choose $H$ with most inliers
Automatic Image Stitching

1. Compute interest points on each image

2. Find candidate matches

3. Estimate homography $H$ using matched points and RANSAC with normalized DLT

4. Project each image onto the same surface and blend
RANSAC for Homography

Initial Matched Points
RANSAC for Homography

Final Matched Points
RANSAC for Homography
Image Blending
Feathering

1 0

+ 1 0

=
Pyramid blending

Create a Laplacian pyramid, blend each level

Poisson Image Editing

For more info: Perez et al, SIGGRAPH 2003

http://research.microsoft.com/vision/cambridge/papers/perez_siggraph03.pdf
Recognizing Panoramas

Some of following material from Brown and Lowe 2003 talk

Recognizing Panoramas

Input: N images

1. Extract SIFT points, descriptors from all images

2. Find K-nearest neighbors for each point (K=4)

3. For each image
   a) Select M candidate matching images by counting matched keypoints (m=6)
   b) Solve homography $H_{ij}$ for each matched image
Recognizing Panoramas

Input: N images

1. Extract SIFT points, descriptors from all images

2. Find K-nearest neighbors for each point (K=4)

3. For each image
   a) Select M candidate matching images by counting matched keypoints (m=6)
   b) Solve homography $H_{ij}$ for each matched image
   c) Decide if match is valid ($n_i > 8 + 0.3 \cdot n_f$)

# inliers
# keypoints in overlapping area
Recognizing Panoramas (cont.)

(now we have matched pairs of images)

4. Find connected components
Finding the panoramas
Finding the panoramas
Finding the panoramas
Recognizing Panoramas (cont.)

(now we have matched pairs of images)

4. Find connected components

5. For each connected component
   a) Solve for rotation and f
   b) Project to a surface (plane, cylinder, or sphere)
   c) Render with multiband blending