Image stitching

Digital Visual Effects

Yuanchuang

Applications of image stitching

- Video stabilization
- Video summarization
- Video compression
- Video matting
- Panorama creation

Image stitching

- Stitching = alignment + blending
  - Geometrical registration
  - Photometric registration

Video summarization
Video compression

Object removal

Object removal

Object removal
Object removal

background estimation

Panorama creation

Why panorama?

• Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°

Why panorama?

• Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
Why panorama?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  - Panoramic Mosaic = 360 x 180°

Panorama examples

- Like HDR, it is a topic of computational photography, seeking ways to build a better camera mostly in software.
- Most consumer cameras have a panorama mode
- Earth: http://www.panoramas.dk/new-year-2006/taipei.html
  http://www.360cities.net/

What can be globally aligned?

- In image stitching, we seek for a matrix to globally warp one image into another. Are any two images of the same scene can be aligned this way?
  - Images captured with the same center of projection
  - A planar scene or far-away scene

A pencil of rays contains all views

Can generate any synthetic camera view as long as it has the same center of projection!
Mosaic as an image reprojection

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

Changing camera center

- Does it still work?

Planar scene (or a faraway one)

- PP3 is a projection plane of both centers of projection, so we are OK!
- This is how big aerial photographs are made

Motion models

- Parametric models as the assumptions on the relation between two images.
2D Motion models

- Translation: 2 unknowns
- Affine: 6 unknowns
- Perspective: 8 unknowns
- 3D rotation: 3 unknowns

A case study: cylindrical panorama

- What if you want a 360° field of view?

Cylindrical panoramas

- Steps
  - Reproject each image onto a cylinder
  - Blend
  - Output the resulting mosaic
Cylindrical panorama

1. Take pictures on a tripod (or handheld)
2. Warp to cylindrical coordinate
3. Compute pairwise alignments
4. Fix up the end-to-end alignment
5. Blending
6. Crop the result and import into a viewer

It is required to do radial distortion correction for better stitching results!

Translation model

Try to align this in PaintShop Pro

Taking pictures

Kaidan panoramic tripod head

Where should the synthetic camera be

- The projection plan of some camera
- Onto a cylinder
Cylindrical projection


Cylindrical projection

(\sin \theta, h, \cos \theta) \propto (x, y, f)

\theta = \tan^{-1} \frac{x}{f}

unwrapped cylinder

Cylindrical projection
Cylindrical projection

\[(\sin \theta, h, \cos \theta) \propto (x, y, f)\]

\[h = \frac{y}{\sqrt{x^2 + f^2}}\]

Cylindrical reprojection

A simple method for estimating \( f \)

Or, you can use other software, such as AutoStitch, to help.
Input images

Cylindrical warping

Blending

- Why blending: parallax, lens distortion, scene motion, exposure difference

Blending
Blending

Assembling the panorama

- Stitch pairs together, blend, then crop

Problem: Drift

- Error accumulation
  - small errors accumulate over time
Problem: Drift

- Solution
  - add another copy of first image at the end
  - there are a bunch of ways to solve this problem
    - add displacement of \((y_1 - y_n)/(n-1)\) to each image after the first
    - compute a global warp: \(y' = y + ax\)
    - run a big optimization problem, incorporating this constraint
      - best solution, but more complicated
      - known as “bundle adjustment”

End-to-end alignment and crop

Viewer: panorama

Example: http://www.cs.washington.edu/education/courses/cse590ss/01wi/projects/project1/students/doug/index.html

Viewer: texture mapped model

Example: http://www.panoramas.dk/
Cylindrical panorama

1. Take pictures on a tripod (or handheld)
2. Warp to cylindrical coordinate
3. Compute pairwise alignments
4. Fix up the end-to-end alignment
5. Blending
6. Crop the result and import into a viewer

Determine pairwise alignment?

- Feature-based methods: only use feature points to estimate parameters

- We will study the “Recognising panorama” paper published in ICCV 2003

- Run SIFT (or other feature algorithms) for each image, find feature matches.

Determine pairwise alignment

- \( p' = Mp \), where \( M \) is a transformation matrix, \( p \) and \( p' \) are feature matches
- It is possible to use more complicated models such as affine or perspective
- For example, assume \( M \) is a 2x2 matrix

\[
\begin{pmatrix}
  x' \\
  y'
\end{pmatrix} =
\begin{pmatrix}
  m_{11} & m_{12} \\
  m_{21} & m_{22}
\end{pmatrix}
\begin{pmatrix}
  x \\
  y
\end{pmatrix}
\]

- Find \( M \) with the least square error

\[
\sum_{i=1}^{n} (Mp - p')^2
\]
Normal equation

Given an overdetermined system

$$Ax = b$$

the normal equation is that which minimizes the sum of the square differences between left and right sides

$$A^T Ax = A^T b$$

Why?

$$E(x) = (Ax - b)^2$$

$$E(x) = \left(\begin{array}{c} a_{11} \ldots a_{1m} \\ \vdots \\ a_{n1} \ldots a_{nm} \end{array}\right) \left(\begin{array}{c} x_1 \\ \vdots \\ x_m \end{array}\right) = \left(\begin{array}{c} b_1 \\ \vdots \\ b_n \end{array}\right)$$

$$n \times m, \ n \text{ equations, } m \text{ variables}$$

$$Ax - b = \left[ \sum_{j=1}^{m} a_{ij} x_j \right] - \left[ \begin{array}{c} b_1 \\ \vdots \\ b_n \end{array}\right] = \left[ \sum_{j=1}^{m} a_{ij} x_j \right] - \left[ \begin{array}{c} b_1 \\ \vdots \\ b_n \end{array}\right]$$

$$E(x) = (Ax - b)^2 = \sum_{i=1}^{n} \left(\sum_{j=1}^{m} a_{ij} x_j - b_i \right)^2$$

$$0 = \frac{\partial E}{\partial x_i} = \sum_{i=1}^{n} 2 \left(\sum_{j=1}^{m} a_{ij} x_j \right) - b_i a_{i1}$$

$$= 2 \sum_{i=1}^{n} a_{i1} \sum_{j=1}^{m} a_{ij} x_j - 2 \sum_{i=1}^{n} a_{i1} b_i$$

$$0 = \frac{\partial E}{\partial x} = 2(A^T Ax - A^T b) \rightarrow A^T Ax = A^T b$$
Normal equation

\[
(Ax - b)^2 = (Ax - b)^T (Ax - b) = (Ax - b)^T (Ax - b) = (x^T A^T - b^T) (Ax - b) = x^T A^T Ax - b^T Ax - x^T A^T b + b^T b = x^T A^T Ax - (A^T b)^T x - (A^T b)^T x + b^T b
\]

\[
\frac{\partial E}{\partial x} = 2A^T Ax - 2A^T b
\]

Determine pairwise alignment

• \( p' = Mp \), where \( M \) is a transformation matrix, \( p \) and \( p' \) are feature matches
• For translation model, it is easier.

\[
E = \sum_{i=1}^{n} \left[ (m_1 + x_i - x_i)^2 + (m_2 + y_i - y_i)^2 \right]
\]

\[
0 = \frac{\partial E}{\partial m_i}
\]

• What if the match is false? Avoid impact of outliers.

RANSAC

• RANSAC = Random Sample Consensus
• An algorithm for robust fitting of models in the presence of many data outliers
• Compare to robust statistics

• Given \( N \) data points \( x_i \), assume that majority of them are generated from a model with parameters \( \Theta \), try to recover \( \Theta \).

RANSAC algorithm

Run \( k \) times:

1. draw \( n \) samples randomly
2. fit parameters \( \Theta \) with these \( n \) samples
3. for each of other \( N-n \) points, calculate its distance to the fitted model, count the number of (inlier points) \( c \)
Output \( \Theta \) with the largest \( c \)

How many times? How big? Smaller is better

How to define? Depends on the problem.
How to determine $k$

$p$: probability of real inliers
$P$: probability of success after $k$ trials

$$P = 1 - (1 - p^n)^k$$

$n$ samples are all inliers
a failure
failure after $k$ trials

$$k = \frac{\log(1 - P)}{\log(1 - p^n)}$$

for $P=0.99$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$p$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.5</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>0.6</td>
<td>97</td>
</tr>
<tr>
<td>6</td>
<td>0.5</td>
<td>293</td>
</tr>
</tbody>
</table>

Example: line fitting

$n=2$

Model fitting
Measure distances

Count inliers

c=3

The best model

c=15

Another trial
RANSAC for Homography

Applications of panorama in VFX

- Background plates
- Image-based lighting
Troy (image-based lighting)


Spiderman 2 (background plate)

Cylindrical projection

- Map 3D point \((X, Y, Z)\) onto a unit cylinder
  \[
  (\tilde{x}, \tilde{y}, \tilde{z}) = \frac{1}{\sqrt{X^2 + Z^2}}(X, Y, Z)
  \]
- Convert to cylindrical coordinates
  \[
  (\sin \theta, h, \cos \theta) = (\tilde{x}, \tilde{y}, \tilde{z})
  \]
- Convert to cylindrical image coordinates
  \[
  (\bar{x}, \bar{y}) = (f \theta, fh) + (\bar{x}_c, \bar{y}_c)
  \]
### Direct vs feature-based

- Direct methods use all information and can be very accurate, but they depend on the fragile “brightness constancy” assumption.
- Iterative approaches require initialization.
- Not robust to illumination change and noise images.
- In early days, direct method is better.

- Feature based methods are now more robust and potentially faster.
- Even better, it can recognize panorama without initialization.

### TODO

- Bundle adjustment
- LM method
- Direct method vs feature-based method
- Frame-rate image alignment for stabilization
- Rick’s CGA 1995 paper? LM method
**Project #2 Image stitching**

- camera availability
- Tripod?
- [http://www.cs.washington.edu/education/courses/cse590ss/CurrentQtr/projects.htm](http://www.cs.washington.edu/education/courses/cse590ss/CurrentQtr/projects.htm)
- [http://www.cs.ubc.ca/~mbrown/panorama/panorama.html](http://www.cs.ubc.ca/~mbrown/panorama/panorama.html)

**blending**

- Alpha-blending
- Photomontage
- Poisson blending
- Adelson’s pyramid blending
- Hdr?

**3D interpretation**

\[
\begin{bmatrix}
  u \\
  v
\end{bmatrix} =
\begin{bmatrix}
  K & 0 & R & t \\
  0 & 1 & 0 & 1
\end{bmatrix} p
\]

**Cylindrical warping**

- Given focal length \( f \) and image center \((x_c, y_c)\)

\[
\begin{align*}
\theta &= \frac{(x_{cyl} - x_c)}{f} \\
h &= \frac{(y_{cyl} - y_c)}{f} \\
\hat{x} &= \sin \theta \\
\hat{y} &= h \\
\hat{z} &= \cos \theta \\
x &= f\hat{x}/\hat{z} + x_c \\
y &= f\hat{y}/\hat{z} + y_c
\end{align*}
\]
Cylindrical projection

- Map 3D point \((X, Y, Z)\) onto cylinder
  \[
  (\bar{x}, \bar{y}, \bar{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X, Y, Z)
  \]
- Convert to cylindrical coordinates
  \[
  (\sin \theta, h \cos \theta) = (\bar{x}, \bar{y})
  \]
- Convert to cylindrical image coordinates
  \[
  (\bar{x}, \bar{y}) = (f \theta, fh) + (\bar{x}_c, \bar{y}_c)
  \]

Cylindrical reprojection

- How to map from a cylinder to a planar image?
  - Apply camera projection matrix
    \[
    \begin{pmatrix}
      w x' \\
      w y' \\
      w
    \end{pmatrix}
    =
    \begin{pmatrix}
      -f & 0 & w/2 & 0 \\
      0 & -f & h/2 & 0 \\
      0 & 0 & 1 & 0
    \end{pmatrix}
    \begin{pmatrix}
      \bar{x} \\
      \bar{y} \\
      \bar{z} \\
      1
    \end{pmatrix}
    \]
  - Convert to image coordinates
    - divide by third coordinate (w)

Levenberg-Marquardt Method
Alignment

- a rotation of the camera is a translation of the cylinder!

\[
\begin{bmatrix}
\sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\
\sum_{x,y} I_x I_y & \sum_{x,y} I_y^2
\end{bmatrix}
\begin{bmatrix}
u_x \\
v_y
\end{bmatrix} = \begin{bmatrix}
\sum_{x,y} (J(x,y) - I(x,y)) I_x \\
\sum_{x,y} (J(x,y) - I(x,y)) I_y
\end{bmatrix}
\]

LucasKanadeStep

void LucasKanadeStep(CByteImage& img1, CByteImage& img2, float t[2]) {
    // Transform the image
    Translation(img2, img2t, t);

    // Compute the gradients and summed error by comparing img1 and img2t
    double A[2][2], b[2];
    for (int y = 1; y < height-1; y++) {  // ignore borders
        for (int x = 1; x < width-1; x++) {
            double e = img2t.Pixel(x, y, k) - img1.Pixel (x, y, k);// Accumulate the matrix entries
            double gx = 0.5*(img2t.Pixel(x+1, y, k) - img2t.Pixel(x-1, y, k));
            double gy = 0.5*(img2t.Pixel(x, y+1, k) - img2t.Pixel(x, y-1, k));
            A[0][0] += gx*gx; A[0][1] += gx*gy;
            A[1][0] += gx*gy; A[1][1] += gy*gy;
            b[0] += e*gx; b[1] += e*gy;
        }
    }

    // Solve for the update At=b and update the vector
    double det = 1.0 / (A[0][0]*A[1][1] - A[0][1]*A[1][0]);
    t[0] += (A[1][1]*b[0] - A[0][1]*b[1]) * det;
    t[1] += (A[0][0]*b[1] - A[1][0]*b[0]) * det;
}

LucasKanadeStep (cont.)

PyramidLucasKanade

void PyramidalLucasKanade(CByteImage& img1, CByteImage& img2, float t[2],
                          int nLevels, int nLucasKanadeSteps) {
    CBytePyramid p1(img1);     // Form the two pyramids
    CBytePyramid p2(img2);

    for (int l = nLevels-1; l >= 0; l--){
        t[0] /= (1 << l);   // scale the t vector
        t[1] /= (1 << l);
        LucasKanadeStep(p1[l], p2[l], t);
        t[0] *= (1 << l);   // restore the full scaling
        t[1] *= (1 << l);
    }
}
Gaussian pyramid

2D Motion models

- translation: \( \mathbf{x}' = \mathbf{x} + \mathbf{t} \) \( \mathbf{x} = (x, y) \)
- rotation: \( \mathbf{x}' = R \mathbf{x} + \mathbf{t} \)
- similarity: \( \mathbf{x}' = s R \mathbf{x} + \mathbf{t} \)
- affine: \( \mathbf{x}' = A \mathbf{x} + \mathbf{t} \)
- perspective: \( \mathbf{x}' \approx H \mathbf{x} \) \( \mathbf{x} = (x, y, 1) \)
  \( \mathbf{x} \) is a \textit{homogeneous} coordinate

- These all form a nested \textit{group} (closed under composition w/ inv.)

Video matting

alpha matte

Recognising Panoramas

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

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

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

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

2D Rotations ($q$, $f$)
- Ordering $\Rightarrow$ matching images

Probabilistic model for verification

- Compare probability that this set of RANSAC inliers/outliers was generated by a correct/false image match

- Choosing values for $p_1$, $p_0$ and $p_{\text{min}}$

$\quad n_i > 5.9 + 0.22n_f$

Overview

- SIFT Feature Matching
- Image Matching
- Bundle Adjustment
- Multi-band Blending
Nearest Neighbour Matching

- Find k-NN for each feature
  - $k \approx$ number of overlapping images (we use $k = 4$)
- Use k-d tree
  - k-d tree recursively bi-partitions data at mean in the dimension of maximum variance
  - Approximate nearest neighbours found in $O(n \log n)$

Overview

- SIFT Feature Matching
- Image Matching
  - For each image, use RANSAC to select inlier features from 6 images with most feature matches
- Bundle Adjustment
- Multi-band Blending

Finding the panoramas
Finding the panoramas

Overview

- SIFT Feature Matching
- Image Matching
- Bundle Adjustment
- Multi-band Blending

Homography for Rotation

- Parameterise each camera by rotation and focal length

\[ R_i = e^{[\theta_i]_\times}, \quad [\theta_i]_\times = \begin{bmatrix} 0 & -\theta_i^3 & \theta_i^2 \\ \theta_i^3 & 0 & -\theta_i \\ -\theta_i & \theta_i & 0 \end{bmatrix} \]

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

- This gives pairwise homographies

\[ \tilde{u}_i = H_{ij}\tilde{u}_j, \quad H_{ij} = K_iR_iR^T_jK_j^{-1} \]
**Error function**

- Sum of squared projection errors

\[ e = \sum_{i=1}^{n} \sum_{j \in I(i)} \sum_{k \in F(i, j)} \left( r_{ij}^k \right)^2 \]

- \( n = \#\text{images} \)
- \( I(i) = \text{set of image matches to image } i \)
- \( F(i, j) = \text{set of feature matches between images } i, j \)
- \( r_{ij}^k = \text{residual of } k^{th} \text{ feature match between images } i, j \)

- **Robust error**

\[ \text{err}_f(x) = \begin{cases} 
|x|, & \text{if } |x| < x_{max} \\
 x_{max}, & \text{if } |x| \geq x_{max} 
\end{cases} \]

---

**Overview**

- SIFT Feature Matching
- Image Matching
- Bundle Adjustment
- Multi-band Blending

---

**Multi-band Blending**

- Burt & Adelson 1983
  - Blend frequency bands over range \( \propto \lambda \)

---

**2-band Blending**

- Low frequency (\( \lambda > 2 \text{ pixels} \))
- High frequency (\( \lambda < 2 \text{ pixels} \))
Linear Blending

2-band Blending

Results

Distortion

- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens
Radial correction

- Correct for “bending” in wide field of view lenses

\[
\begin{align*}
\hat{r}^2 &= \hat{x}^2 + \hat{y}^2 \\
\hat{x}' &= \hat{x}/(1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
\hat{y}' &= \hat{y}/(1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
x &= f \hat{x}'/\hat{z} + x_c \\
y &= f \hat{y}'/\hat{z} + y_c
\end{align*}
\]