Image stitching

Digital Visual Effects, Spring 2005

Yung-Yu Chuang

2005/3/30

with slides by Richard Szeliski, Steve Seitz, Matthew Brown and Vaclav Hlavac

Announcements

- Project #1 was due yesterday.
- Project #2 handout will be available on the web later tomorrow.
- I will set up a webpage for artifact voting soon.

Outline

- Image stitching
- Motion models
- Direct methods
- Feature-based methods
- Applications
- Project #2

Image stitching

- Stitching = alignment + blending

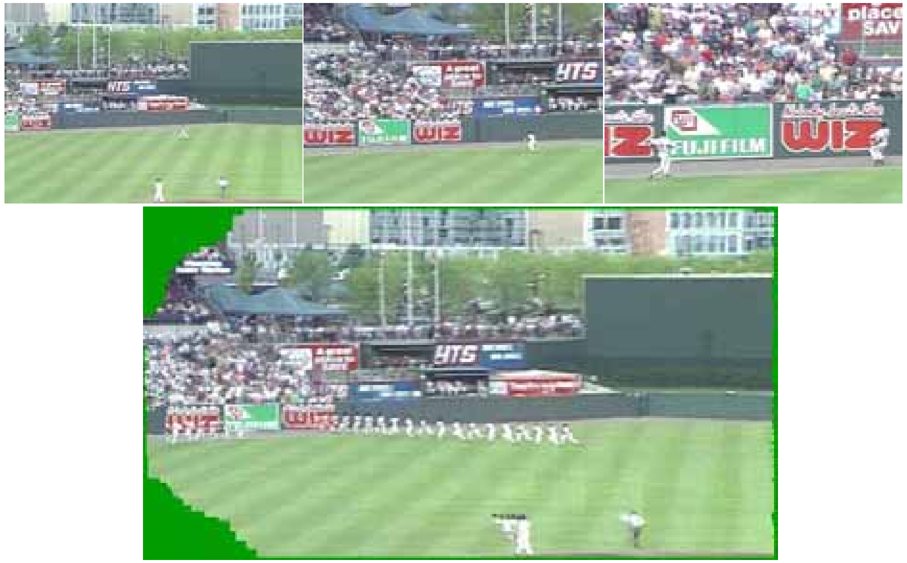
alignment ← geometrical registration

blending ← photometric registration

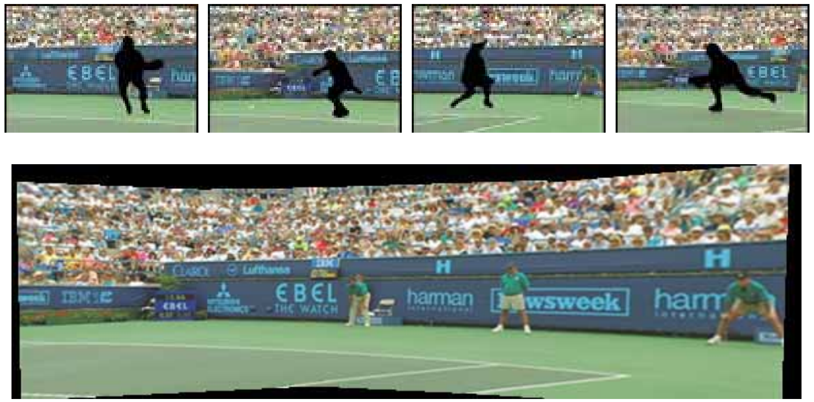
Applications of image stitching

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

Video summarization



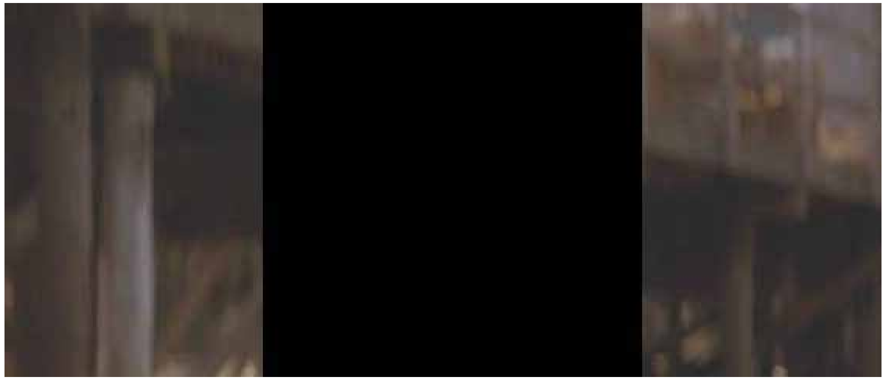
Video compression



Video matting

input video

Video matting



remove foreground

Video matting

estimate background

Video matting

background estimation

Video matting

alpha matte

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

- Panorama mode in consumer cameras
- Mars:

http://www.panoramas.dk/fullscreen3/f2_mars97.html

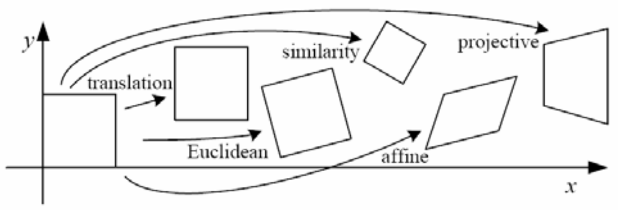
- Earth:

http://earthobservatory.nasa.gov/Newsroom/BlueMarble/

2D motion models

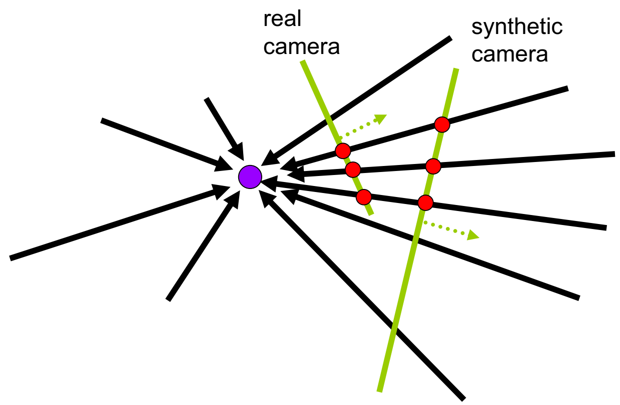
- translation: $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{t}$ $\quad \boldsymbol{x} = (x,y)$
- rotation: $\boldsymbol{x}' = \boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- similarity: $\boldsymbol{x}' = s\,\boldsymbol{R}\,\boldsymbol{x} + \boldsymbol{t}$
- affine: $\boldsymbol{x}' = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{t}$
- perspective: $\underline{\boldsymbol{x}}' \cong \boldsymbol{H}\,\underline{\boldsymbol{x}}$ $\quad \underline{\boldsymbol{x}} = (x,y,1)$
  ($\underline{\boldsymbol{x}}$ is a *homogeneous* coordinate)
- These all form a nested *group* (closed under composition w/ inv.)

2D image transformations



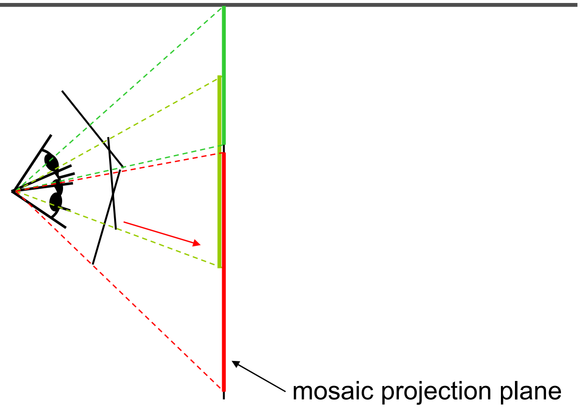
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$	8	straight lines	

A pencil of rays contains all views



Can generate any synthetic camera view
as long as it has **the same center of projection**!

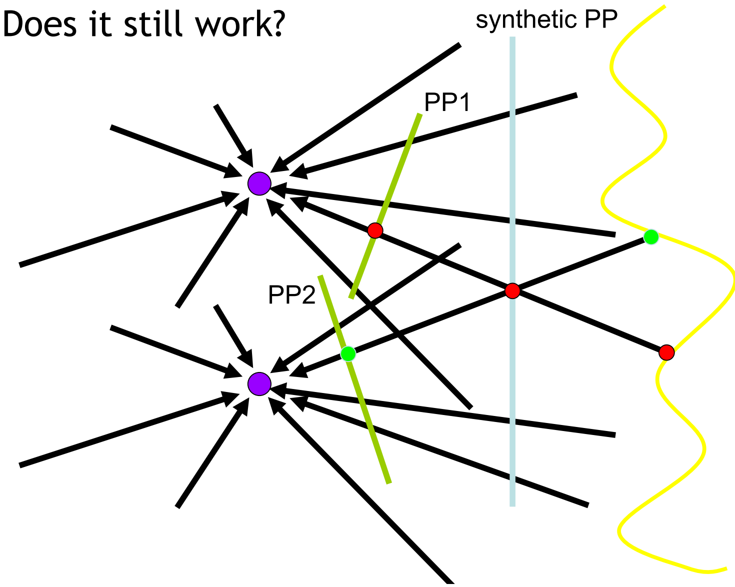
Mosaic as Image Reprojection



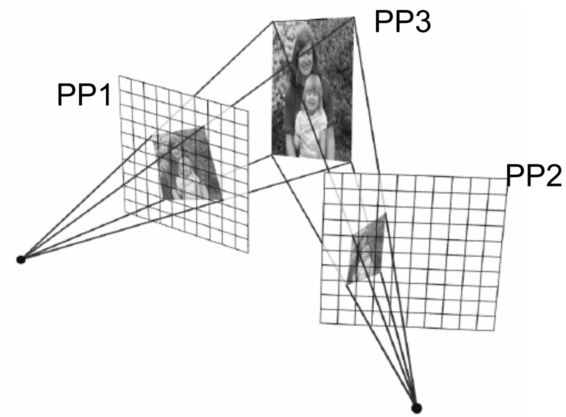
- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*

Changing camera center

- Does it still work?

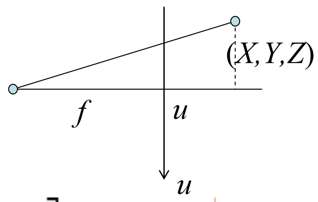


Planar scene (or far away)



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

3D motion models



$$\begin{bmatrix}X\\Y\\Z\end{bmatrix} = \begin{bmatrix}\mathbf{R}\end{bmatrix}_{3\times3}\begin{bmatrix}X\\Y\\Z\end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix}u\\v\\1\end{bmatrix} \sim \begin{bmatrix}U\\V\\W\end{bmatrix} = \begin{bmatrix}f&0&u\\0&f&v\\0&0&1\end{bmatrix}\begin{bmatrix}X\\Y\\Z\end{bmatrix}$$

3D motion models

- Rotational
- Cylindrical

Direct methods

- Select a motion model and estimate parameters
- Direct methods use pixel-to-pixel matching
- We have covered this last time actually.
- We will show a case study on constructing cylindrical panorama using a direct method.

A case study: cylindrical panorama

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

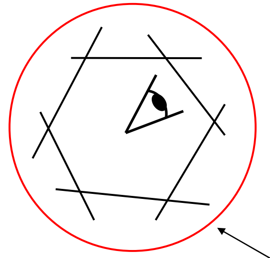
Taking pictures

Kaidan panoramic tripod head

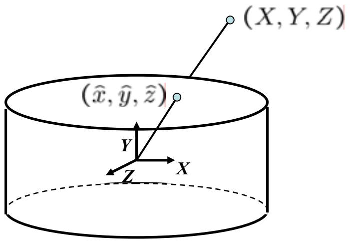
Translation model

A case study: cylindrical panorama

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



Cylindrical projection



- Map 3D point (X,Y,Z) onto cylinder

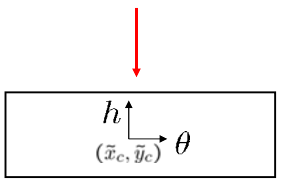
$$(\hat{x},\hat{y},\hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X,Y,Z)$$

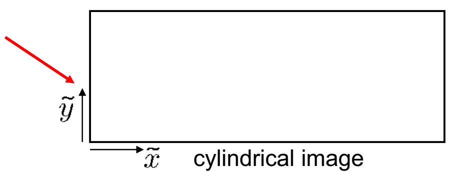
- Convert to cylindrical coordinates

$$(sin\theta, h, cos\theta) = (\hat{x},\hat{y},\hat{z})$$

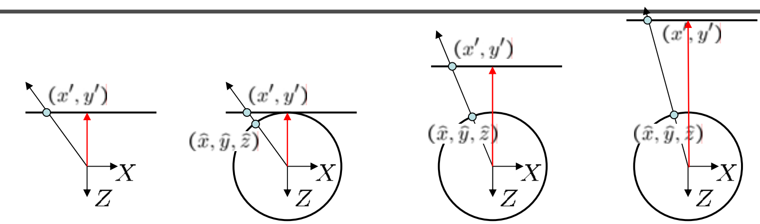
- Convert to cylindrical image coordinates

$$(\tilde{x},\tilde{y}) = (f\theta, fh) + (\tilde{x}_c,\tilde{y}_c)$$





Cylindrical reprojection



Focal length – the dirty secret…

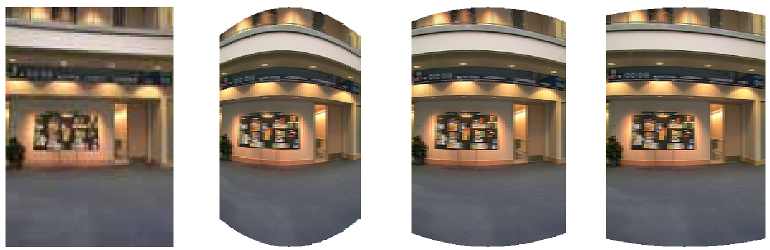


Image 384x300 f = 180 (pixels) f = 280 f = 380

A simple method for estimating f

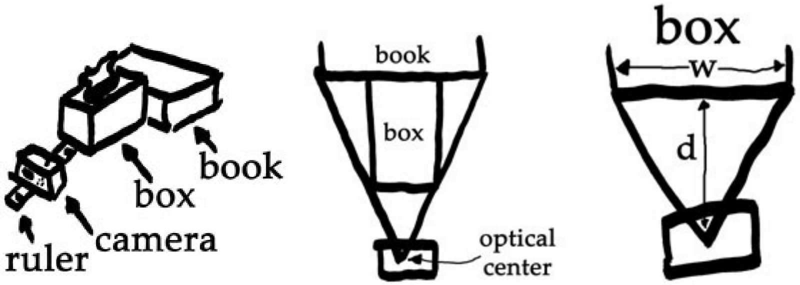
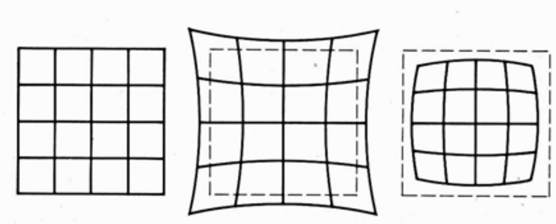


Figure A Figure B Figure C

We will discuss more accurate methods next time

Distortion

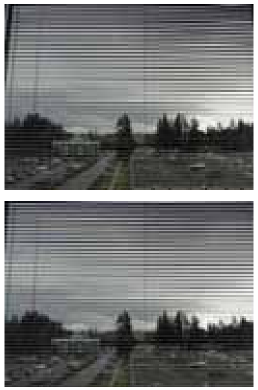


No distortion Pin cushion Barrel

- 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



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

Input images

Cylindrical warping

Alignment

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

$$\begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_{x,y} I_x \left(J(x,y) - I(x,y)\right) \\ \sum_{x,y} I_y \left(J(x,y) - I(x,y)\right) \end{bmatrix}$$

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++) {
         // If both have full alphas, then compute and accumulate the error
         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;
      }
   }
```

LucasKanadeStep (cont.)

```
      // Solve for the update At=b and update the vector

      double det = 1.0 / (A[0][0]*A[1][1] - A[1][0]*A[1][0]);

      t[0] += (A[1][1]*b[0] - A[1][0]*b[1]) * det;
      t[1] += (A[0][0]*b[1] - A[1][0]*b[0]) * det;
}
```

PyramidLucasKanade

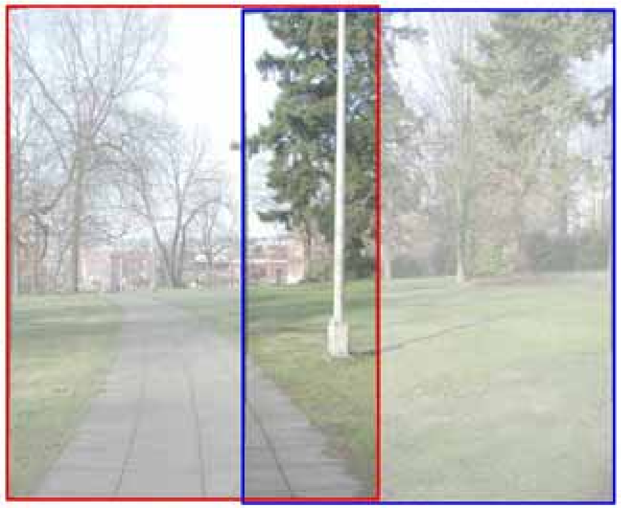
```
void PyramidalLucasKanade(CByteImage& img1, CByteImage& img2, float t[2],
                          int nLevels, int nLucasKanadeSteps)
{
   CBytePyramid p1(img1);     // Form the two pyramids
   CBytePyramid p2(img2);

   // Process in a coarse-to-fine hierarchy
   for (int l = nLevels-1; l >= 0; l--)
   {
      t[0] /= (1 << l);   // scale the t vector
      t[1] /= (1 << l);
      CByteImage& i1 = p1[l];
      CByteImage& i2 = p2[l];

      for (int k = 0; k < nLucasKanadeSteps; k++)
            LucasKanadeStep(i1, i2, t);
      t[0] *= (1 << l);   // restore the full scaling
      t[1] *= (1 << l);
   }
}
```

Gaussian pyramid

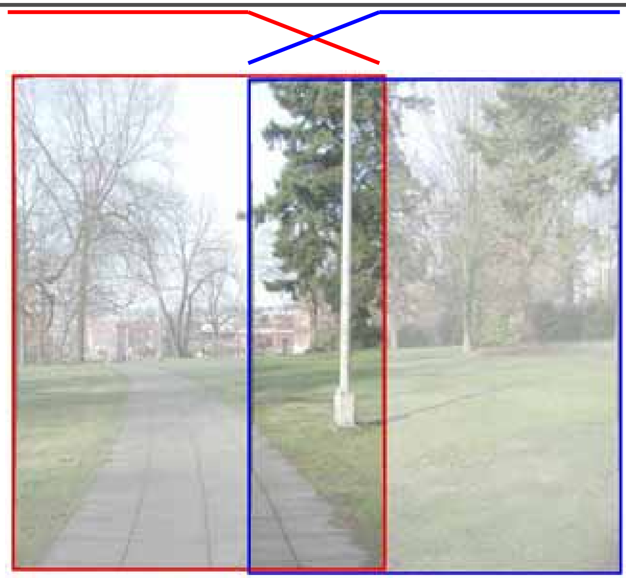
Blending

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

Blending

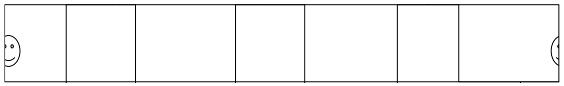


Blending



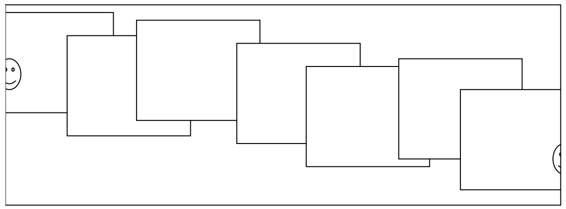
Blending

Assembling the panorama



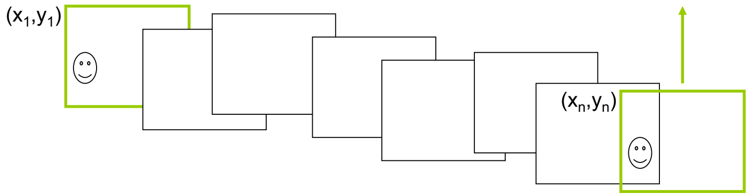
- Stitch pairs together, blend, then crop

Problem: Drift



- Error accumulation
  - small errors accumulate over time

Problem: Drift



- copy of first image

- 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/dougz/index.html

Viewer: texture mapped model

example: http://www.panoramas.dk/

Feature-based methods

- Only use feature points to estimate parameters
- We will study the "Recognising panorama" paper published in ICCV 2003

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 mjority of them are generated from a model with parameters Θ, try to recover Θ.

RANSAC algorithm

Run k times: ← How many times?

(1) draw n samples randomly ← How big? Smaller is better

(2) fit parameters Θ 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 ← How to define? Depends on the problem.

Output Θ with the largest c

How to determine k

p: probability of real inliers

P: probability of success after k trials

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

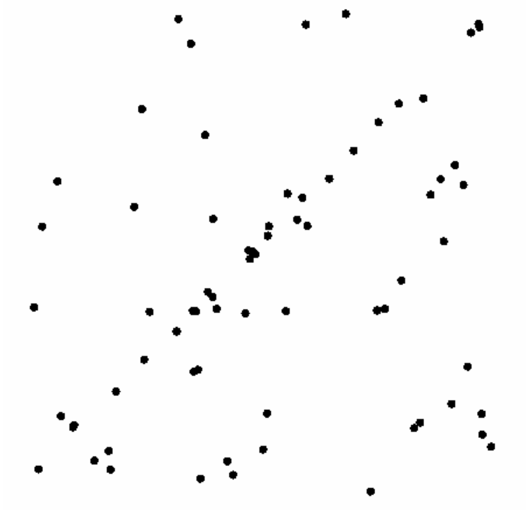
- p^n: n samples are all inliers
- $1 - p^n$: a failure
- $(1 - p^n)^k$: failure after k trials

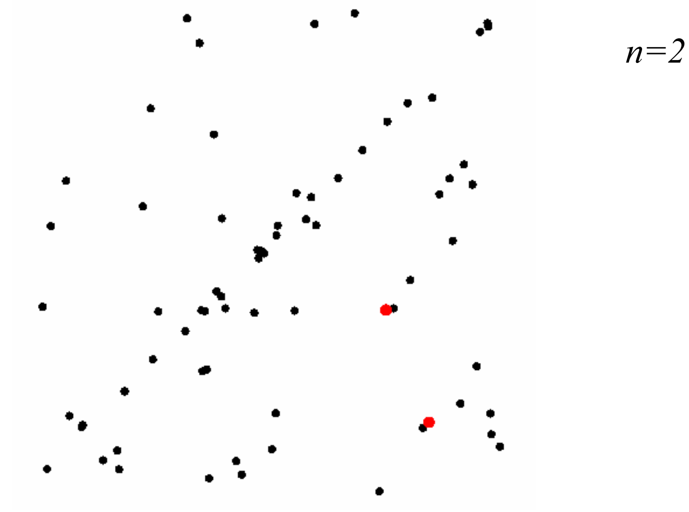
$$k = \frac{\log(1-P)}{\log(1-p^n)}$$

for P=0.99

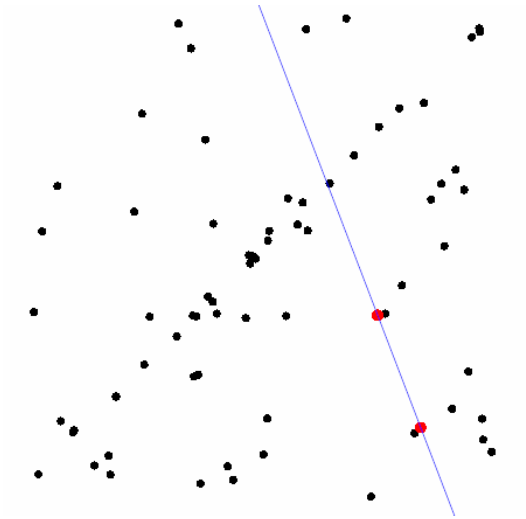
n	p	k
3	0.5	35
6	0.6	97
6	0.5	293

Example: line fitting

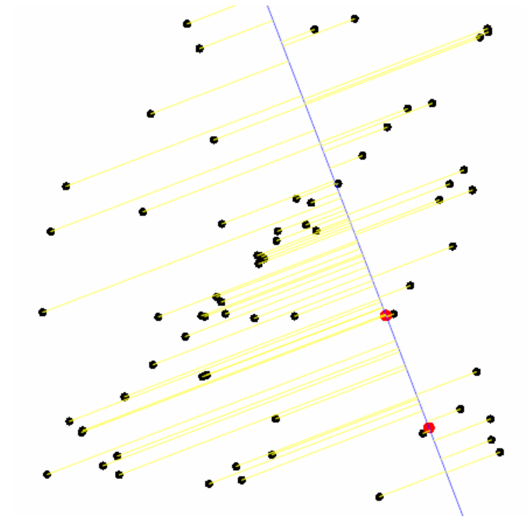




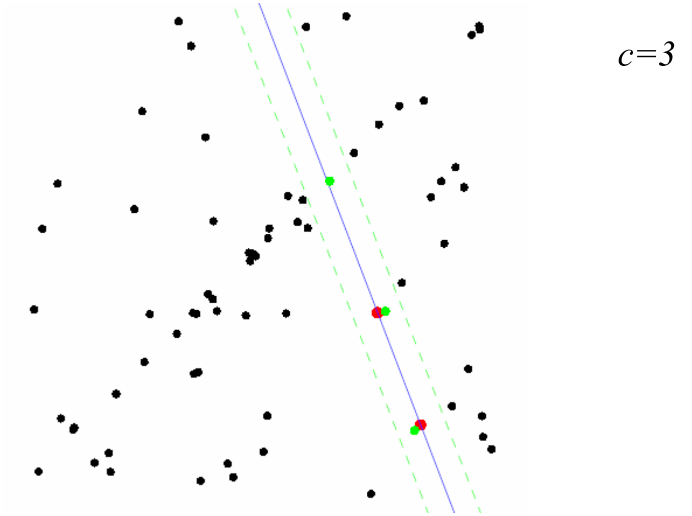
Model fitting



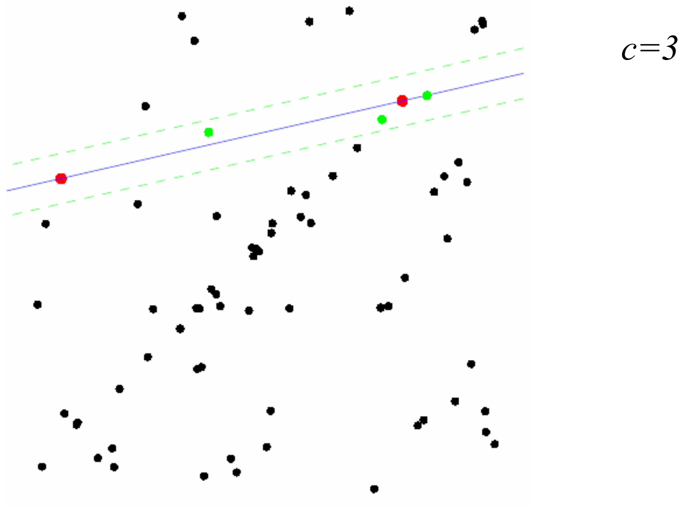
Measure distances



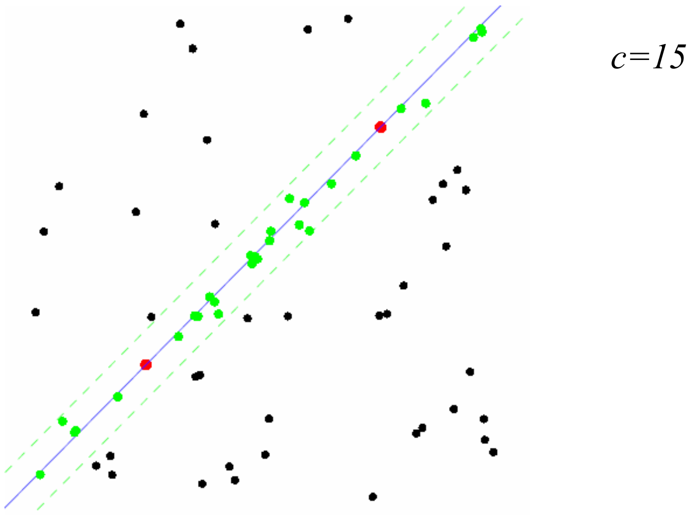
Count inliers



Another trial



The best model



Recognising Panoramas

- 1D Rotations (θ)
  - Ordering ⇒ matching images

Recognising Panoramas

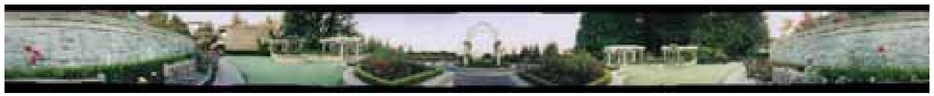
- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

Recognising Panoramas

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

Recognising Panoramas

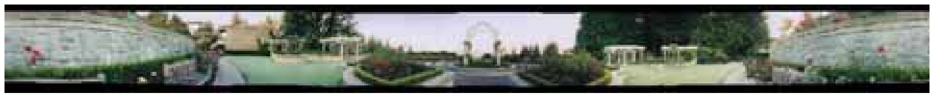
- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images



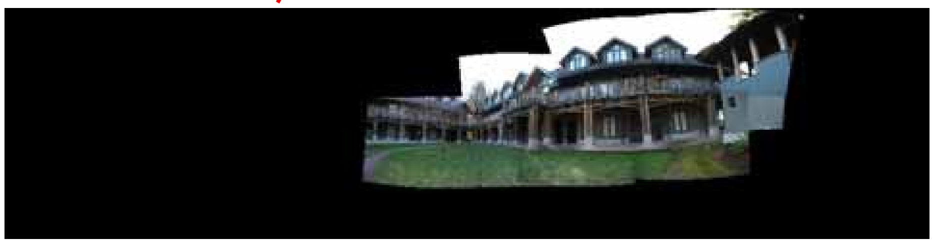
- 2D Rotations (q, f)
  - Ordering $\not\Rightarrow$ matching images

Recognising Panoramas

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



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

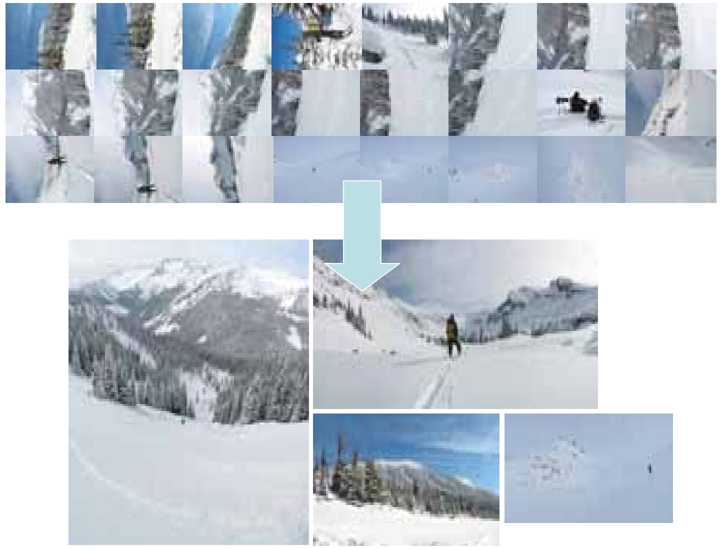


Recognising Panoramas

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

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

Recognising Panoramas



Overview

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

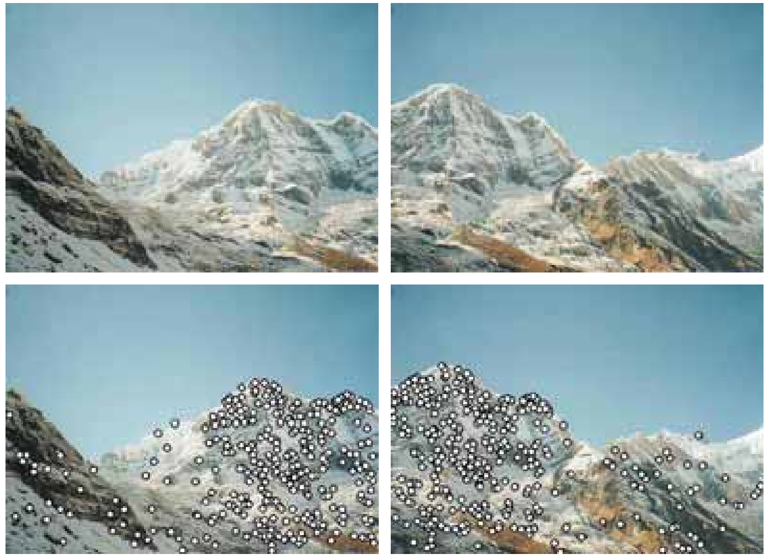
Nearest Neighbour Matching

- 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(nlogn)

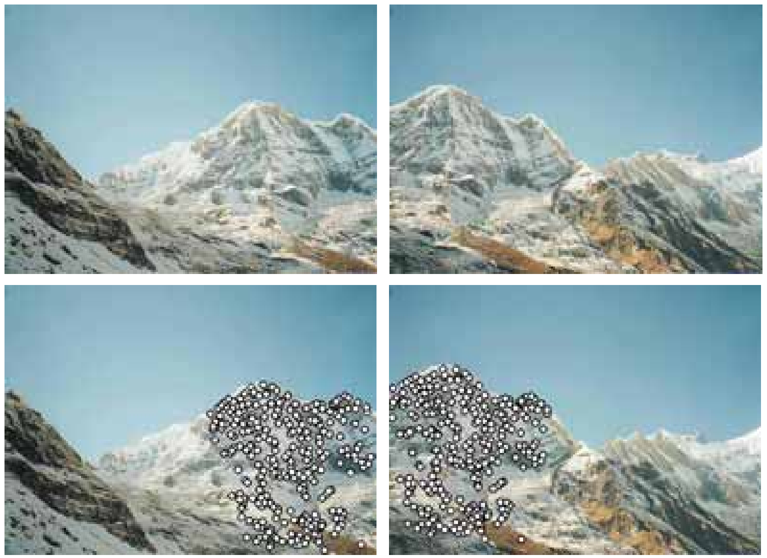
Overview

- 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

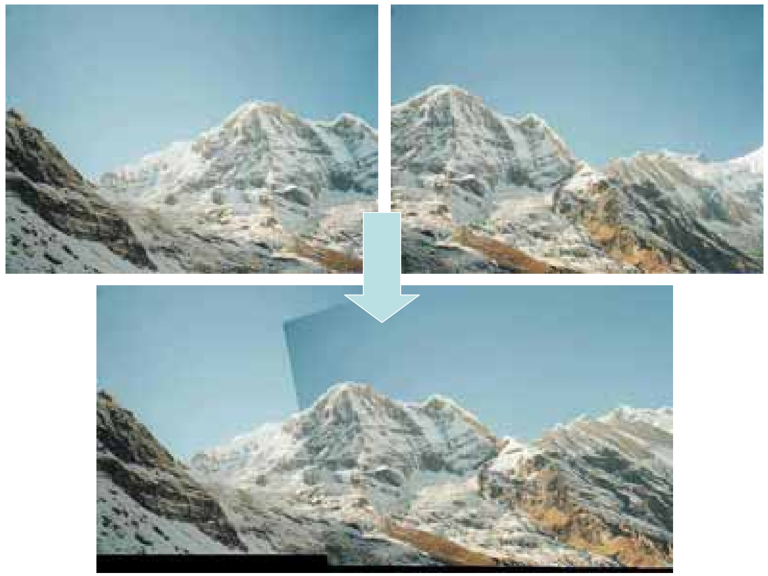
RANSAC for Homography



RANSAC for Homography



RANSAC for Homography

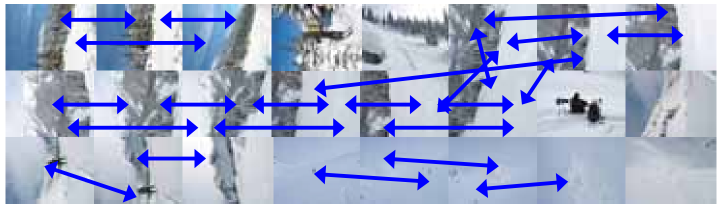


Probabilistic model for verification

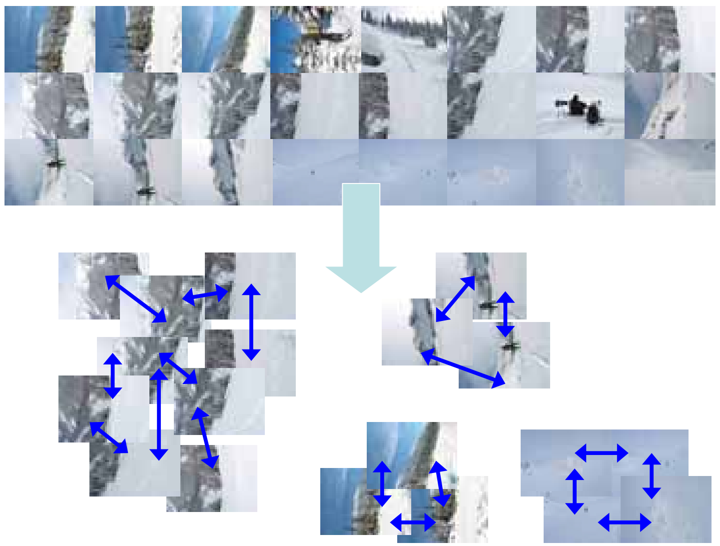
- Compare probability that this set of RANSAC inliers/outliers was generated by a correct/false image match
- Choosing values for p_1, p_0 and p_{min}

$$n_i > 5.9 + 0.22 n_f$$

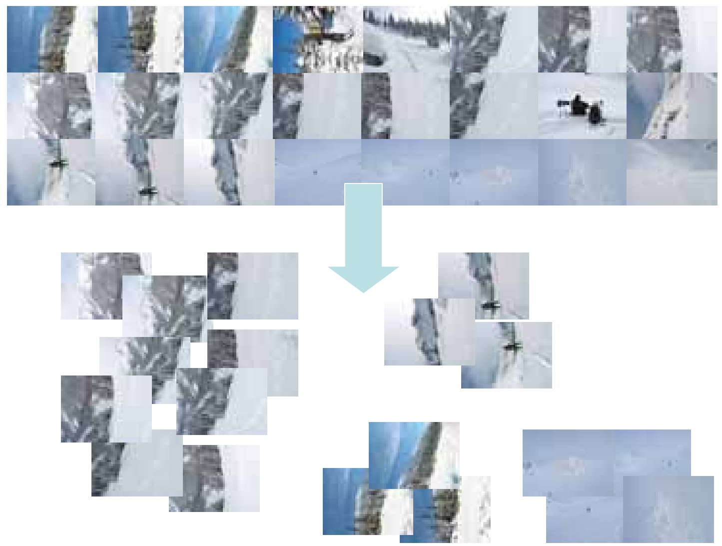
Finding the panoramas



Finding the panoramas



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

$$\mathbf{R}_i = e^{[\boldsymbol{\theta}_i]_\times}, \ [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}$$

$$\mathbf{K}_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- This gives pairwise homographies

$$\tilde{\mathbf{u}}_i = \mathbf{H}_{ij}\tilde{\mathbf{u}}_j, \ \mathbf{H}_{ij} = \mathbf{K}_i\mathbf{R}_i\mathbf{R}_j^T\mathbf{K}_j^{-1}$$

Error function

- Sum of squared projection errors

$$e = \sum_{i=1}^{n} \sum_{j \epsilon \mathcal{I}(i)} \sum_{k \epsilon \mathcal{F}(i,j)} f(\mathbf{r}_{ij}^k)^2$$

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

- Robust err $f(\mathbf{x}) = \begin{cases} |\mathbf{x}|, & \text{if } |\mathbf{x}| < x_{max} \\ x_{max}, & \text{if } |\mathbf{x}| \geq x_{max} \end{cases}$

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

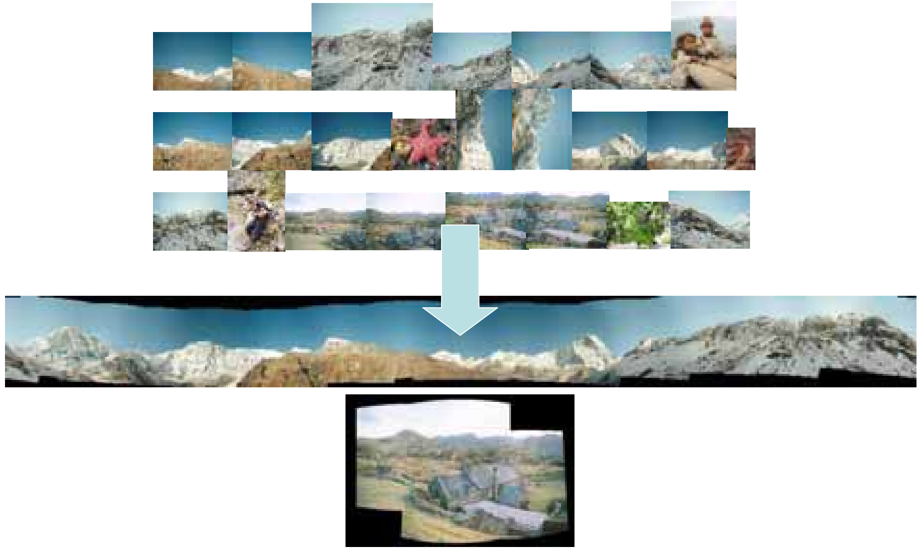
Low frequency (λ > 2 pixels)

High frequency (λ < 2 pixels)

Linear Blending

2-band Blending

Results



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

Applications of panorama in VFX

- Background plates
- Image-based lighting

Spiderman 2 (background plate)

Troy (image-based lighting)

http://www.cgnetworks.com/story_custom.php?story_id=2195&page=4

Project #2 Image stitching

- Assigned: 3/30
- Due: 11:59pm 4/19
- Work in pairs

Reference software

- Autostitch

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

- Many others are available online.

Bells & whistles

- Full SIFT implementation
- Recognizing panorama
- Bundle adjustment
- Handle dynamic objects
- Better blending techniques

Artifacts

- Take your own pictures and generate a stitched image, be creative.
- http://www.cs.washington.edu/education/courses/cse590ss/01wi/projects/project1/students/allen/index.html

Tips for taking pictures

- Common focal point
- Rotate your camera to increase vertical FOV
- Tripod
- Fixed exposure?

Submission

- You have to turn in your complete source, the executable, a html report and an artifact.
- Report page contains:

  description of the project, what do you learn, algorithm, implementation details, results, bells and whistles…
- Artifacts must be made using your own program.

  artifacts voting on forum.

Reference

- Richard Szeliski, Image Alignment and Stitching, unpublished draft, 2005.
- R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and texture-mapped models, SIGGRAPH 1997, pp251-258.
- M. Brown, D. G. Lowe, Recognising Panoramas, ICCV 2003.