

Photo Stitching

Panoramas from Multiple Images

Computer Vision
CS 543 / ECE 549
University of Illinois

Derek Hoiem

So far, we've looked at what can be done with one image

- Recover basic geometry using vanishing points
- Find image boundaries and segment objects
- Categorize images
- Find specific objects and detect objects that are part of some category

What can we get from multiple images?

What can we get from multiple images?

- Bigger, Better, Brighter, Sharper images
 - Panoramas
 - Increased dynamic range
 - Super-resolution
 - Reduced noise/blur



Product example: <http://www.vreveal.com/>

What can we get from multiple images?

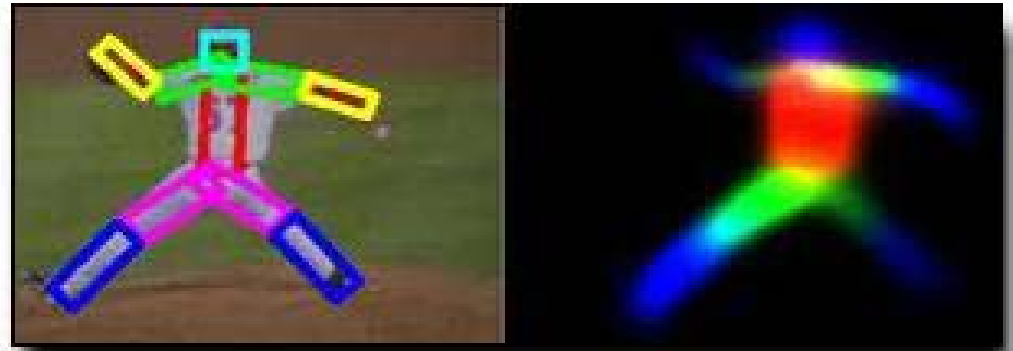
- Bigger, Better, Brighter, Sharper images
 - **Panoramas** ← today
 - Increased dynamic range
 - Super-resolution
 - Reduced noise/blur



Product example: <http://www.vreveal.com/>

What can we get from multiple images?

- Motion
 - Tracking
 - Optical flow
 - Action/activity recognition

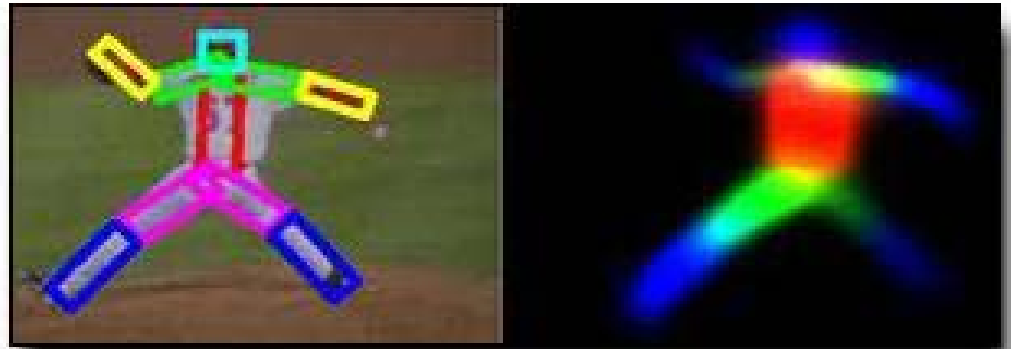


Tracking (from Deva Ramanan)

What can we get from multiple images?

- Motion
 - Tracking
 - Optical flow
 - Action/activity recognition

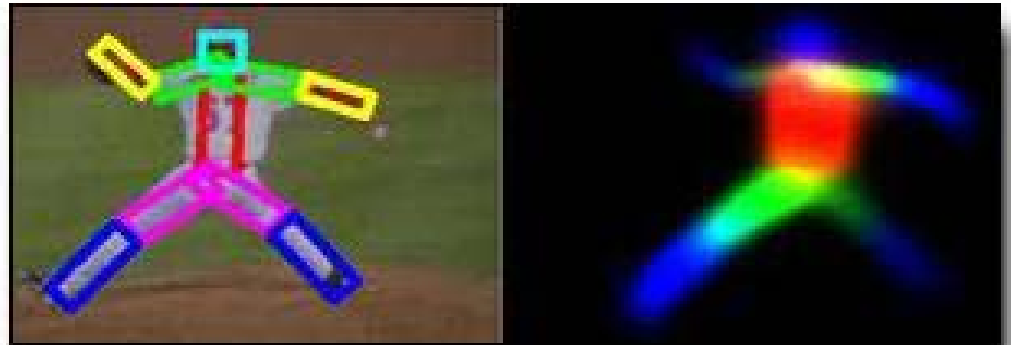
← Thursday



Tracking (from Deva Ramanan)

What can we get from multiple images?

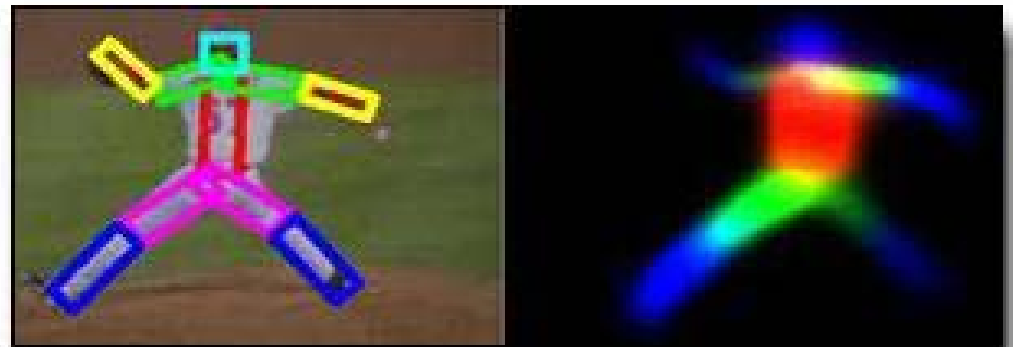
- Motion
 - Tracking ← April 19
 - Optical flow
 - Action/activity recognition



Tracking (from Deva Ramanan)

What can we get from multiple images?

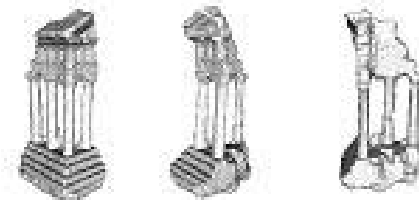
- Motion
 - Tracking
 - Optical flow
 - **Action/activity recognition** ← April 21



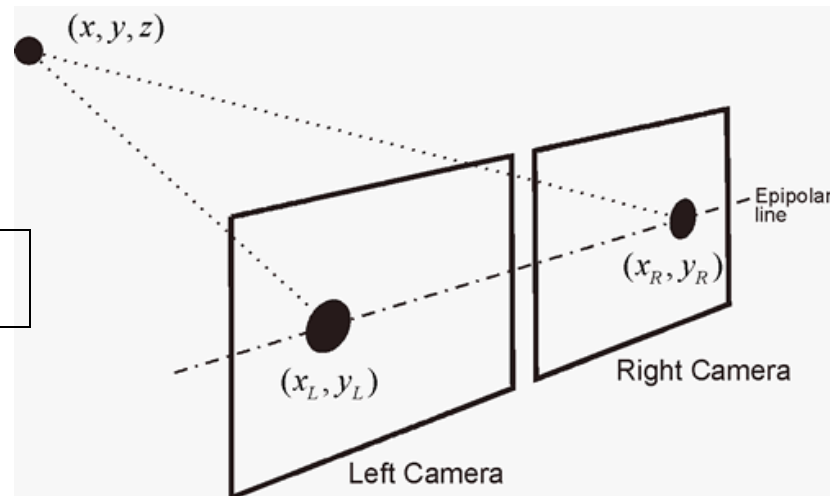
Tracking (from Deva Ramanan)

What can we get from multiple images?

- Depth and 3D structure
 - Two-view stereo
 - Multi-view stereo
 - Shape carving
 - Structure from motion



Next week



Key ideas

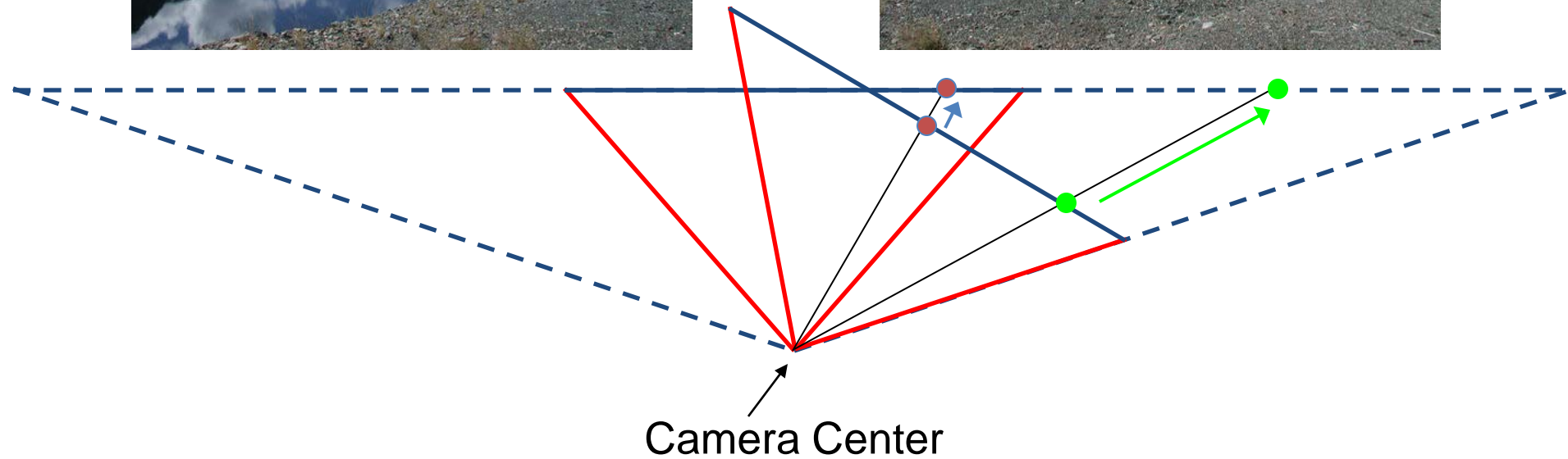
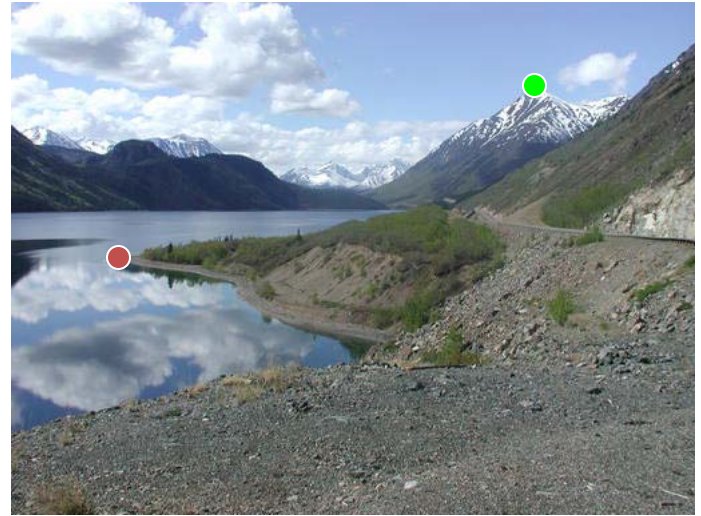
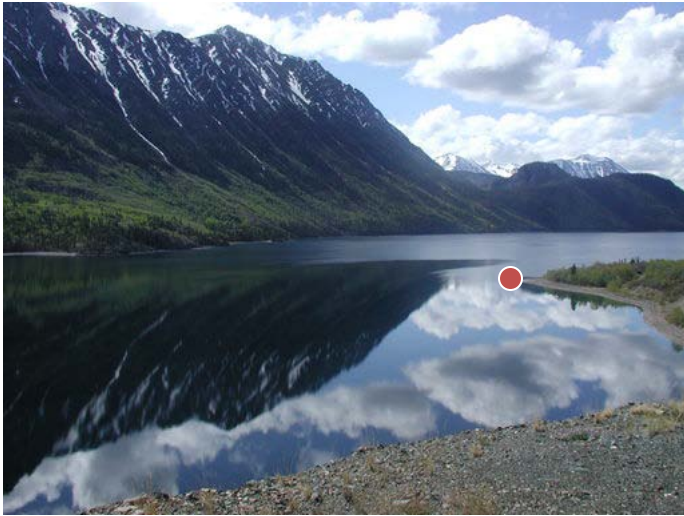
- Correspondence
 - Detect interest points
 - Match two patches
- Alignment and model fitting
 - Solve for motion or depth or camera movement from multiple matched points

Today: Image Stitching

- Combine two or more overlapping images to make one larger image



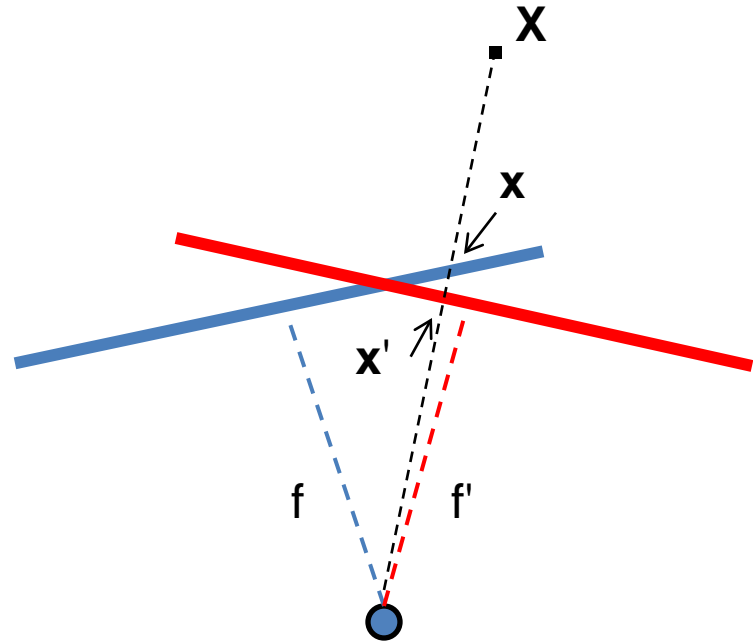
Example



Review of projective geometry

Problem set-up

- $x = K [R \ t] X$
- $x' = K' [R' \ t'] X$
- $t=t'=0$



- $x' = Hx$ where $H = K' R' R^{-1} K^{-1}$
- Typically only R and f will change (4 parameters), but, in general, H has 8 parameters

Image Stitching Algorithm Overview

1. Detect keypoints
2. Match keypoints
3. Estimate homography with four matched keypoints (using RANSAC)
4. Combine images

Computing homography

Assume we have four matched points: How do we compute homography \mathbf{H} ?

Direct Linear Transformation (DLT)

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \quad \mathbf{x}' = \begin{bmatrix} w'u' \\ w'v' \\ w' \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

$$\begin{bmatrix} -u & -v & -1 & 0 & 0 & 0 & uu' & vu' & u' \\ 0 & 0 & 0 & -u & -v & -1 & uv' & vv' & v' \end{bmatrix} \mathbf{h} = \mathbf{0}$$

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix}$$



Computing homography

Direct Linear Transform

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1 u'_1 & v_1 u'_1 & u'_1 \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1 v'_1 & v_1 v'_1 & v'_1 \\ & & & \vdots & & & & & \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_n v'_n & v_n v'_n & v'_n \end{bmatrix} \mathbf{h} = \mathbf{0} \Rightarrow \mathbf{A} \mathbf{h} = \mathbf{0}$$

- Apply SVD: $\mathbf{U} \mathbf{D} \mathbf{V}^T = \mathbf{A}$
- $\mathbf{h} = \mathbf{V}_{\text{smallest}}$ (column of \mathbf{V} corr. to smallest singular value)

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_9 \end{bmatrix} \quad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Matlab

```
[U, S, V] = svd(A);  
h = V(:, end);
```

Computing homography

- Assume we have four matched points: How do we compute homography \mathbf{H} ?

Normalized DLT

1. Normalize coordinates for each image

- a) Translate for zero mean
- b) Scale so that average distance to origin is $\sqrt{2}$

$$\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x} \quad \tilde{\mathbf{x}}' = \mathbf{T}'\mathbf{x}'$$

- This makes problem better behaved numerically (see HZ p. 107-108)

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

3. Unnormalize: $\mathbf{H} = \mathbf{T}'^{-1}\tilde{\mathbf{H}}\mathbf{T}$

$$\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$$

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

For probability p of no outliers:

$$N = \log(1 - p) / \log(1 - (1 - \epsilon)^s)$$

- N , number of samples
- s , size of sample set
- ϵ , proportion of outliers


e.g. for $p = 0.95$

Sample size	Proportion of outliers ϵ						
s	5%	10%	20%	25%	30%	40%	50%
2	2	2	3	4	5	7	11
3	2	3	5	6	8	13	23
4	2	3	6	8	11	22	47
5	3	4	8	12	17	38	95
6	3	4	10	16	24	63	191
7	3	5	13	21	35	106	382
8	3	6	17	29	51	177	766

Computing homography

- Assume we have matched points with outliers: How do we compute homography \mathbf{H} ?

Automatic Homography Estimation with RANSAC

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

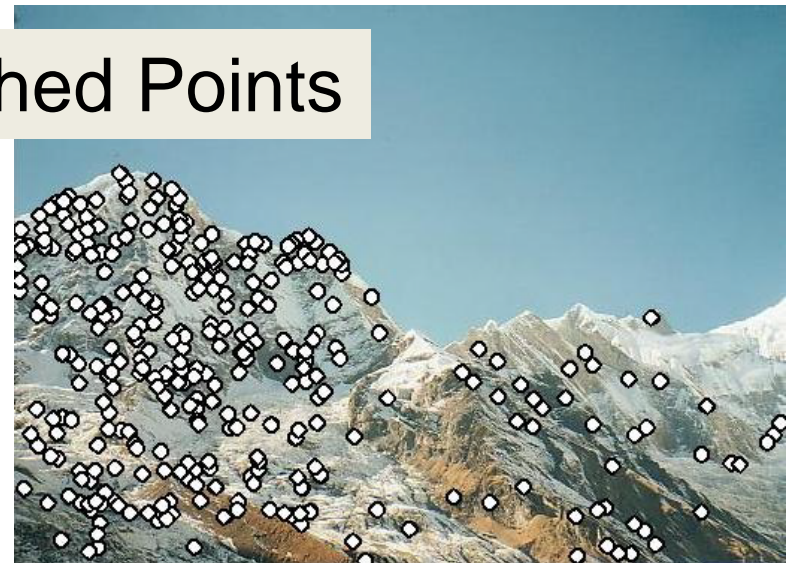
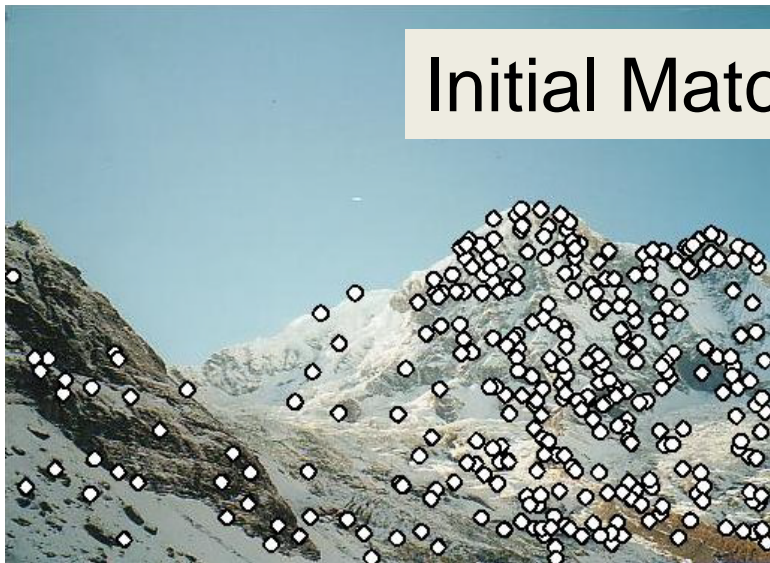
Automatic Image Stitching

1. Compute interest points on each image
2. Find candidate matches
3. Estimate homography **H** using matched points and RANSAC with normalized DLT
4. Transform second image and blend the two images
 - Matlab: maketform, imtransform

RANSAC for Homography



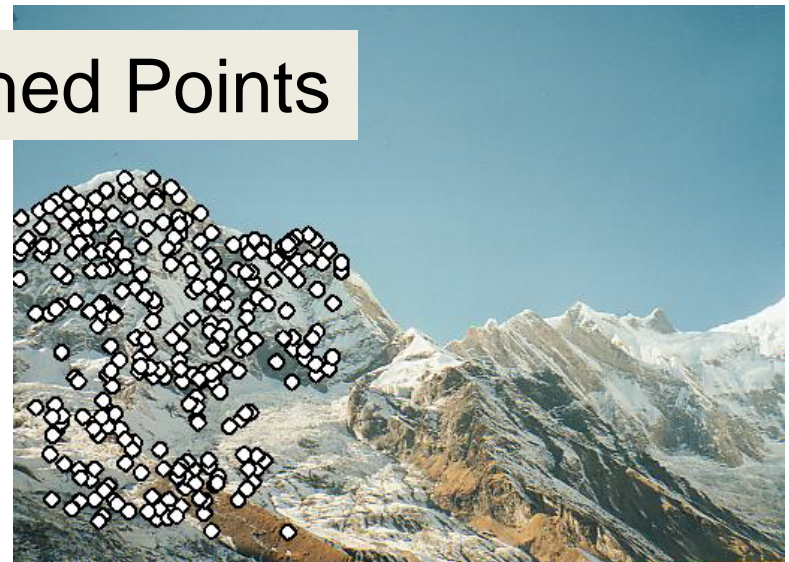
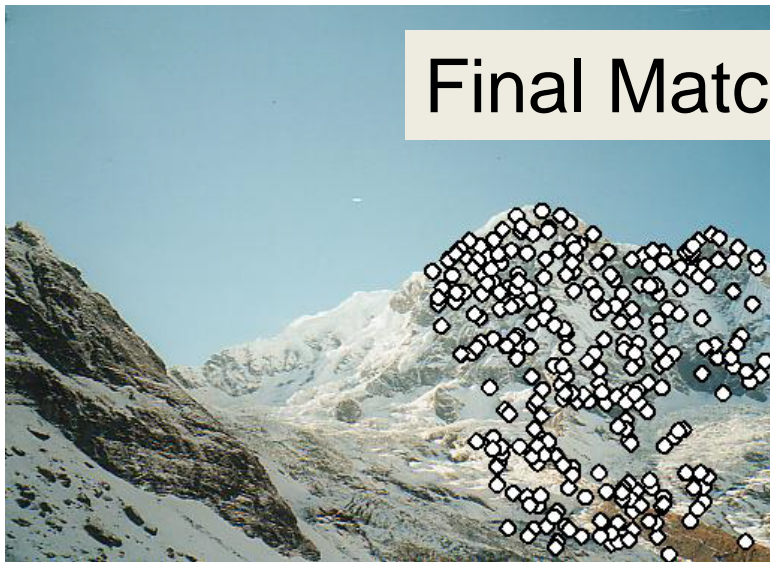
Initial Matched Points



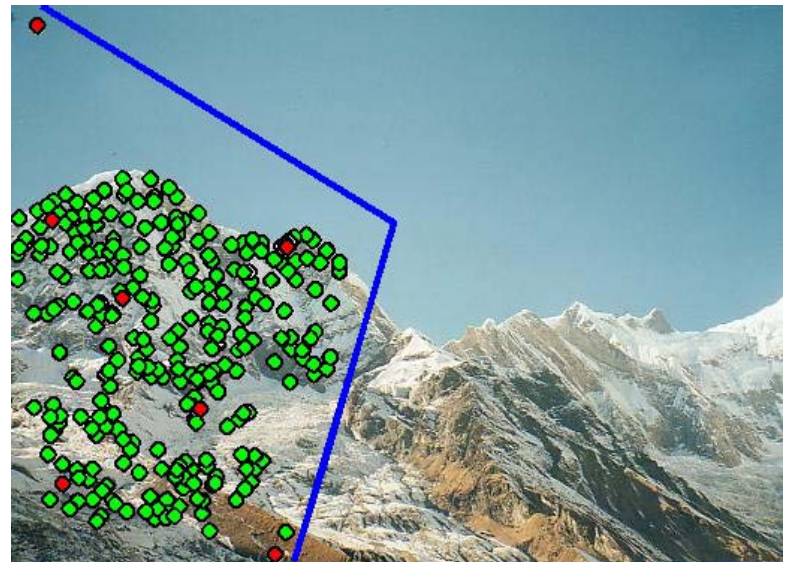
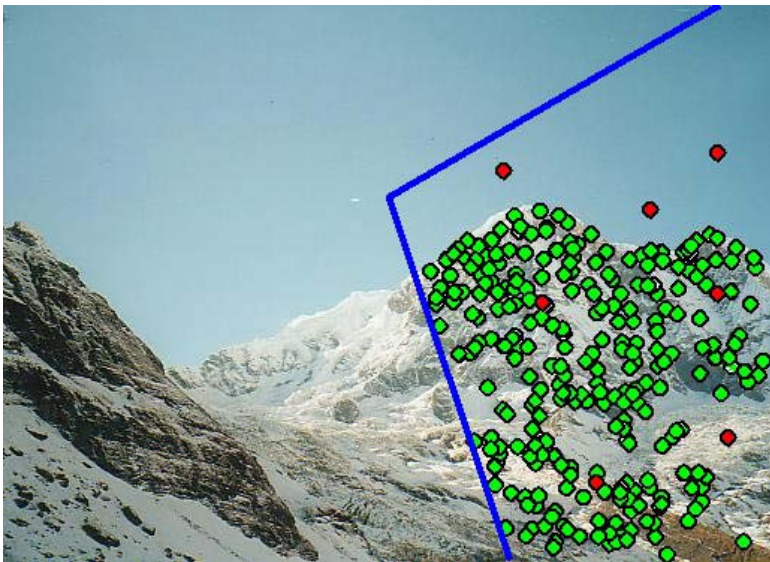
RANSAC for Homography



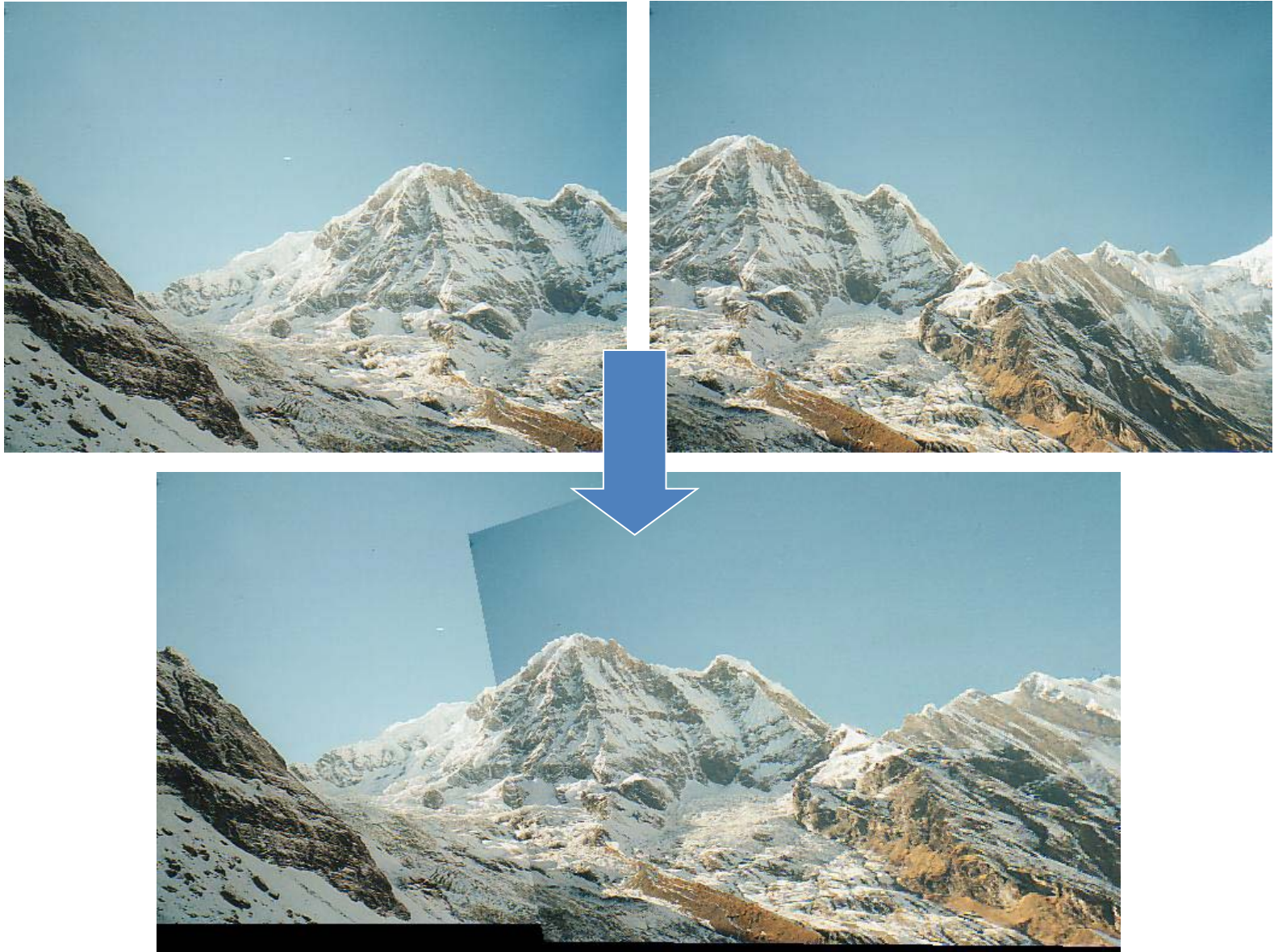
Final Matched Points



Verification

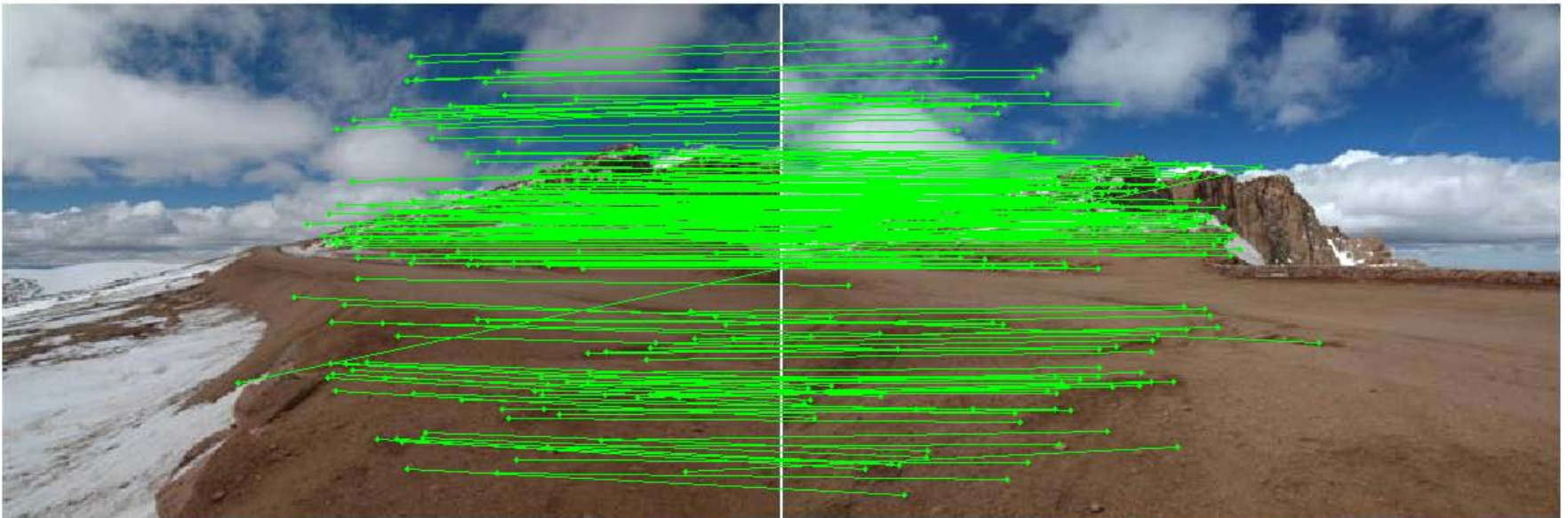


RANSAC for Homography



Choosing a Projection Surface

Many to choose: planar, cylindrical, spherical, cubic, etc.



Planar vs. Cylindrical Projection



Planar

Planar vs. Cylindrical Projection

Planar



Planar vs. Cylindrical Projection

Cylindrical



Planar vs. Cylindrical Projection

Cylindrical





Planar

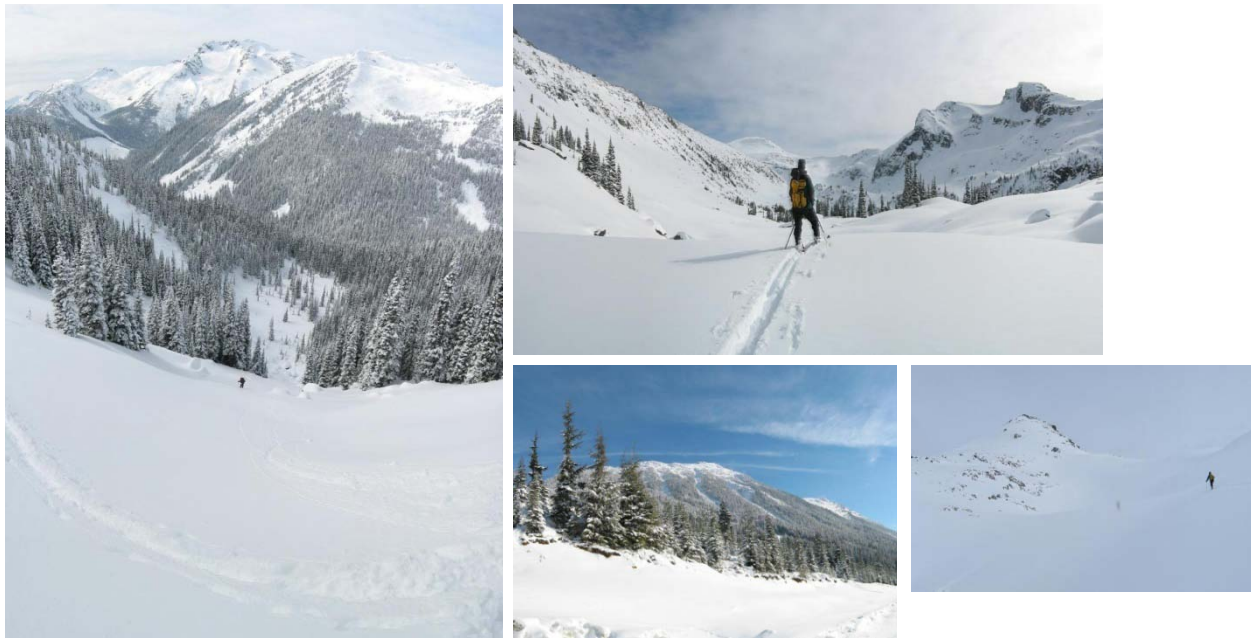
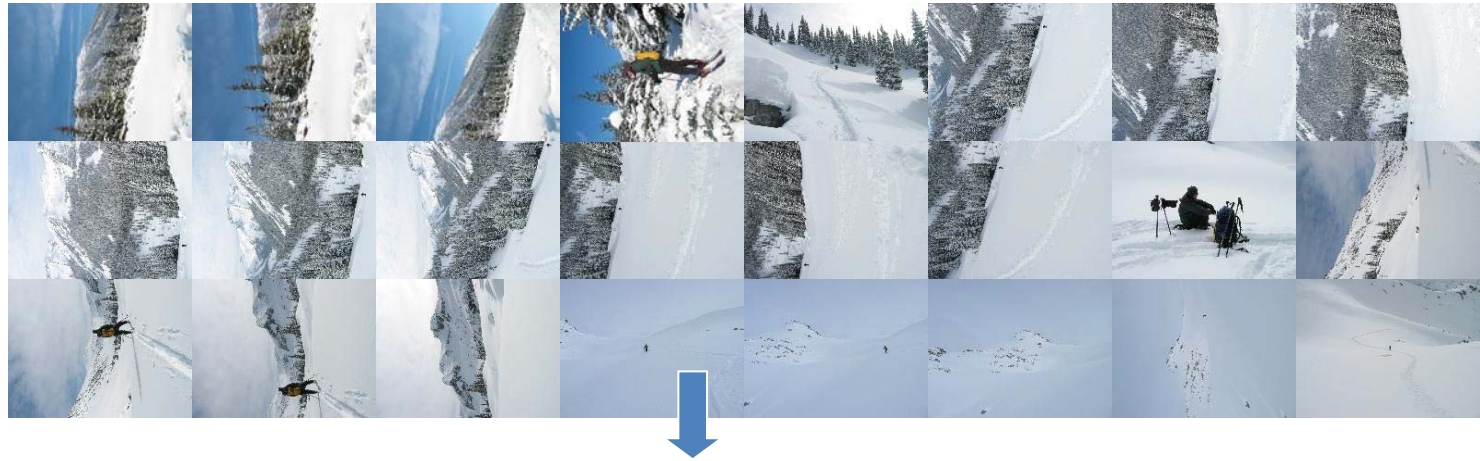


Cylindrical

Simple gain adjustment



Recognizing Panoramas



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 \mathbf{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 \mathbf{H}_{ij} for each matched image
 - c) Decide if match is valid ($n_i > 8 + 0.3 n_f$)



inliers



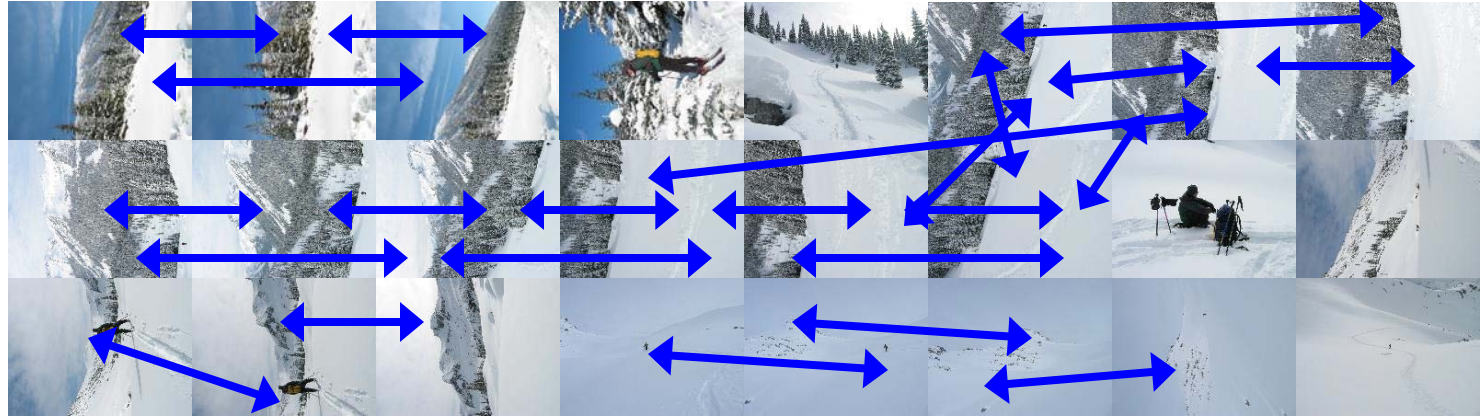
keypoints in overlapping area

Recognizing Panoramas (cont.)

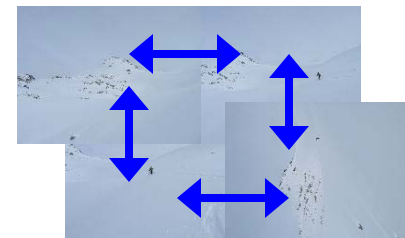
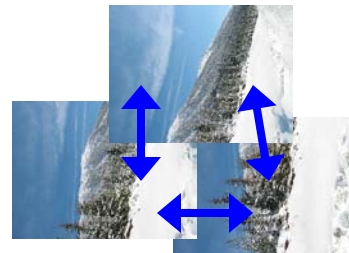
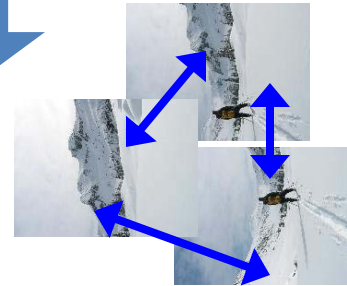
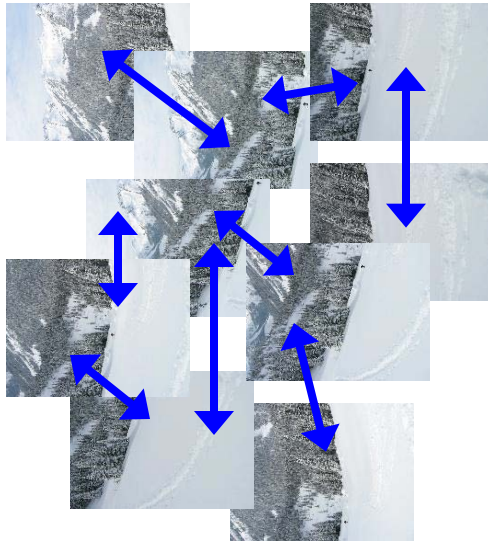
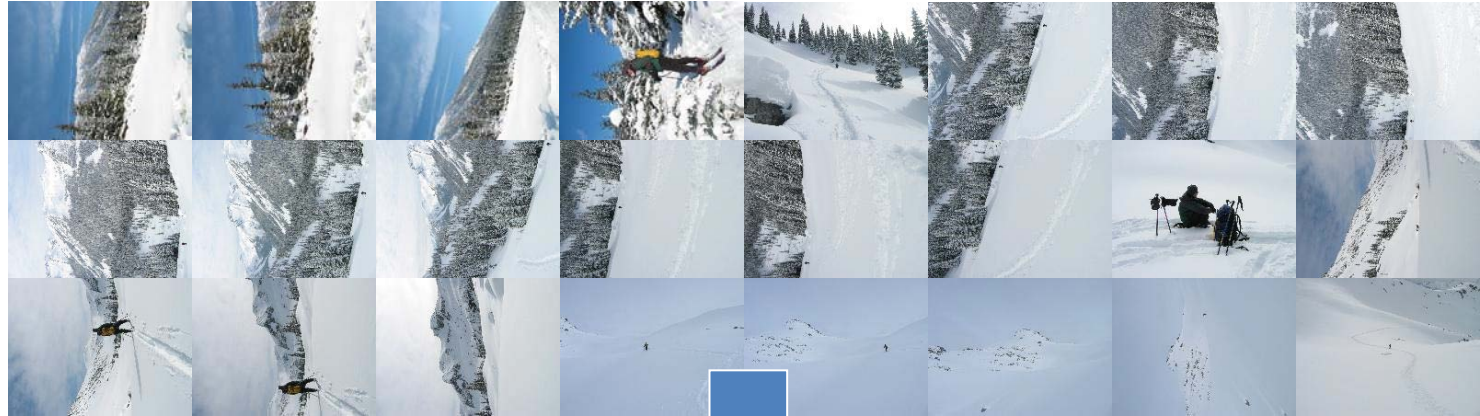
(now we have matched pairs of images)

4. Find connected components

Finding the panoramas



Finding the panoramas



Recognizing Panoramas (cont.)

(now we have matched pairs of images)

4. Find connected components
5. For each connected component
 - a) Perform bundle adjustment to solve for rotation $(\theta_1, \theta_2, \theta_3)$ and focal length f of all cameras
 - b) Project to a surface (plane, cylinder, or sphere)
 - c) Render with multiband blending

Bundle adjustment for stitching

- Non-linear minimization of re-projection error

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

- $\hat{\mathbf{x}}' = \mathbf{H}\mathbf{x}$ where $\mathbf{H} = \mathbf{K}' \mathbf{R}' \mathbf{R}^{-1} \mathbf{K}^{-1}$

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

$$error = \sum_1^N \sum_j^{M_i} \sum_k dist(\mathbf{x}', \hat{\mathbf{x}}')$$

- Solve non-linear least squares (Levenberg-Marquardt algorithm)
 - See paper for details

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



Blending

- Gain compensation: minimize intensity difference of overlapping pixels
- Blending
 - Pixels near center of image get more weight
 - Multiband blending to prevent blurring

Multi-band Blending

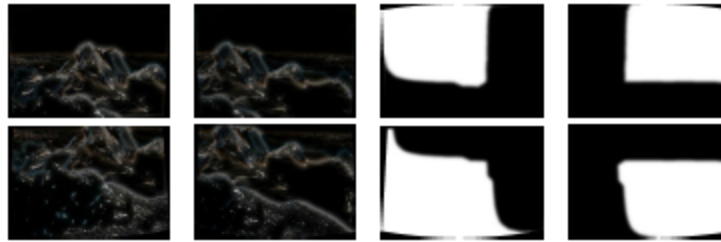
- Burt & Adelson 1983
 - Blend frequency bands over range $\propto \lambda$



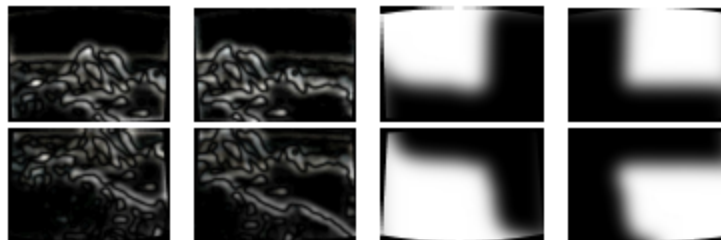
Multiband blending



(a) Original images and blended result



(b) Band 1 (scale 0 to σ)



(c) Band 2 (scale σ to 2σ)



(d) Band 3 (scale lower than 2σ)

Blending comparison (IJCV 2007)



(a) Linear blending



(b) Multi-band blending

Blending Comparison



(b) Without gain compensation



(c) With gain compensation



(d) With gain compensation and multi-band blending

Further reading

- DLT algorithm: HZ p. 91 (alg 4.2), p. 585
- Normalization: HZ p. 107-109 (alg 4.2)
- RANSAC: HZ Sec 4.7, p. 123, alg 4.6
- [Recognising Panoramas](#): Brown and Lowe, IJCV 2007 (also bundle adjustment)

Things to remember

- Homography relates rotating cameras
- Recover homography using RANSAC and normalized DLT
- Bundle adjustment minimizes reprojection error for set of related images
- Details to make it look nice (e.g., blending)

Next class

- Corner tracking and optical flow