Photo Stitching

Panoramas from Multiple Images

Computer Vision

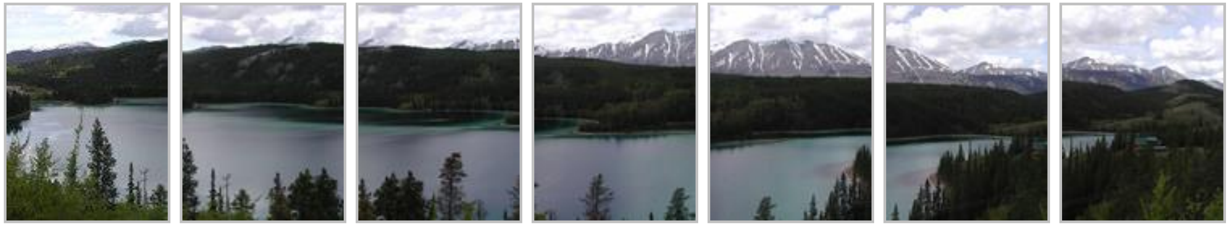
CS 543 / ECE 549

University of Illinois

Derek Hoiem

Image Stitching

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

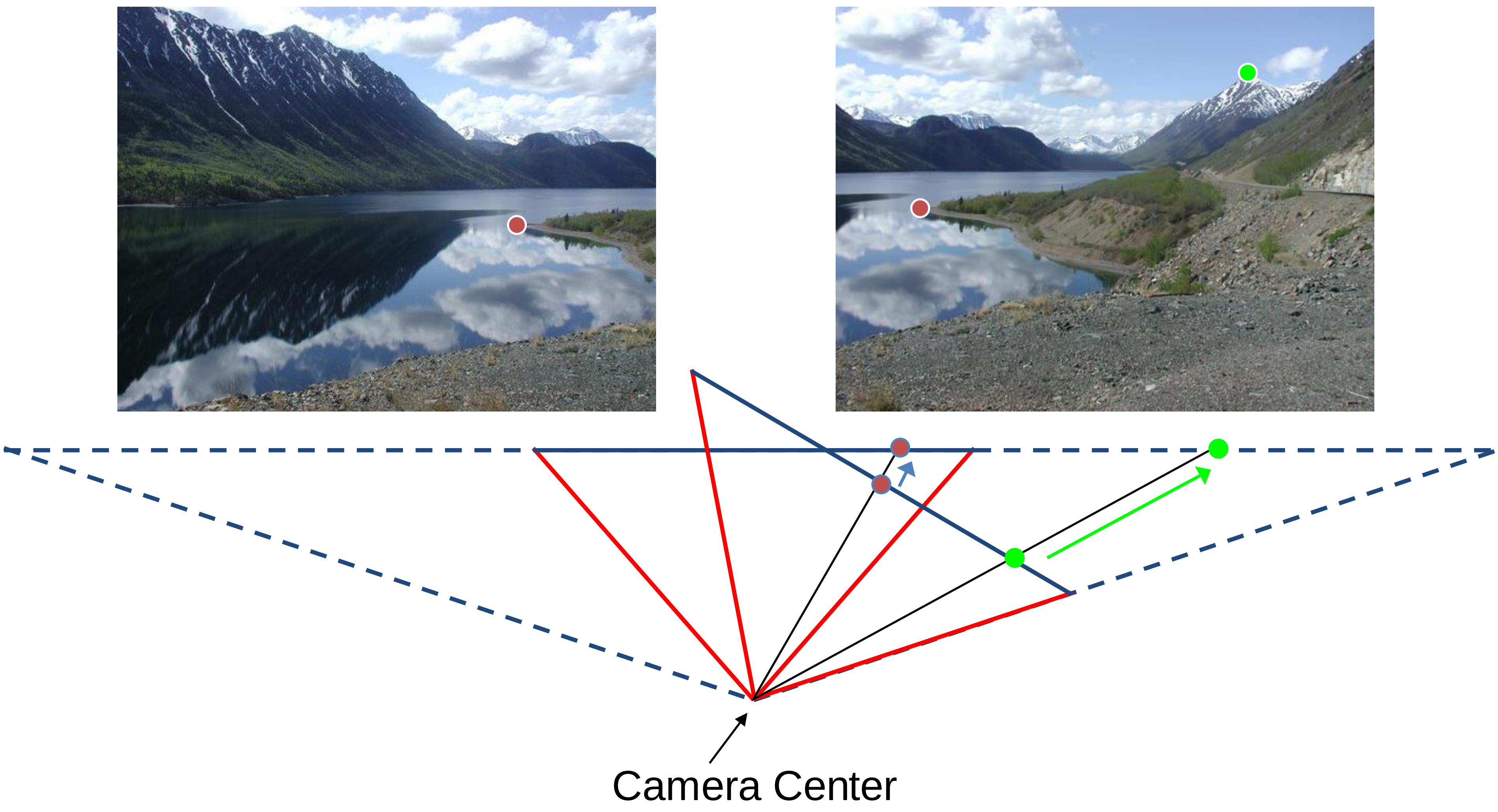


Slide credit: [Vaibhav Vaish]

Concepts introduced/reviewed in today's lecture

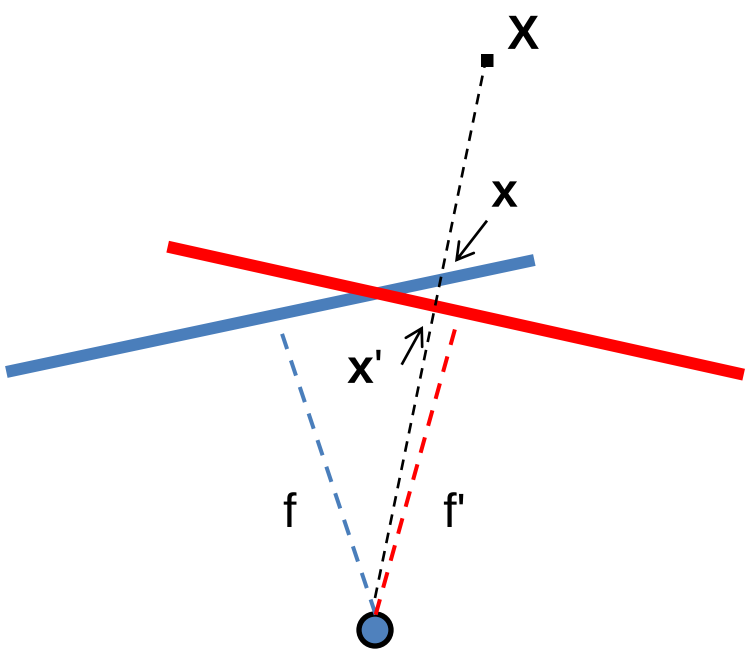
- Camera model
- Homographies
- Solving homogeneous systems of linear equations
- Keypoint-based alignment
- RANSAC
- Blending
- How the iphone stitcher works

Illustration



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

Homography

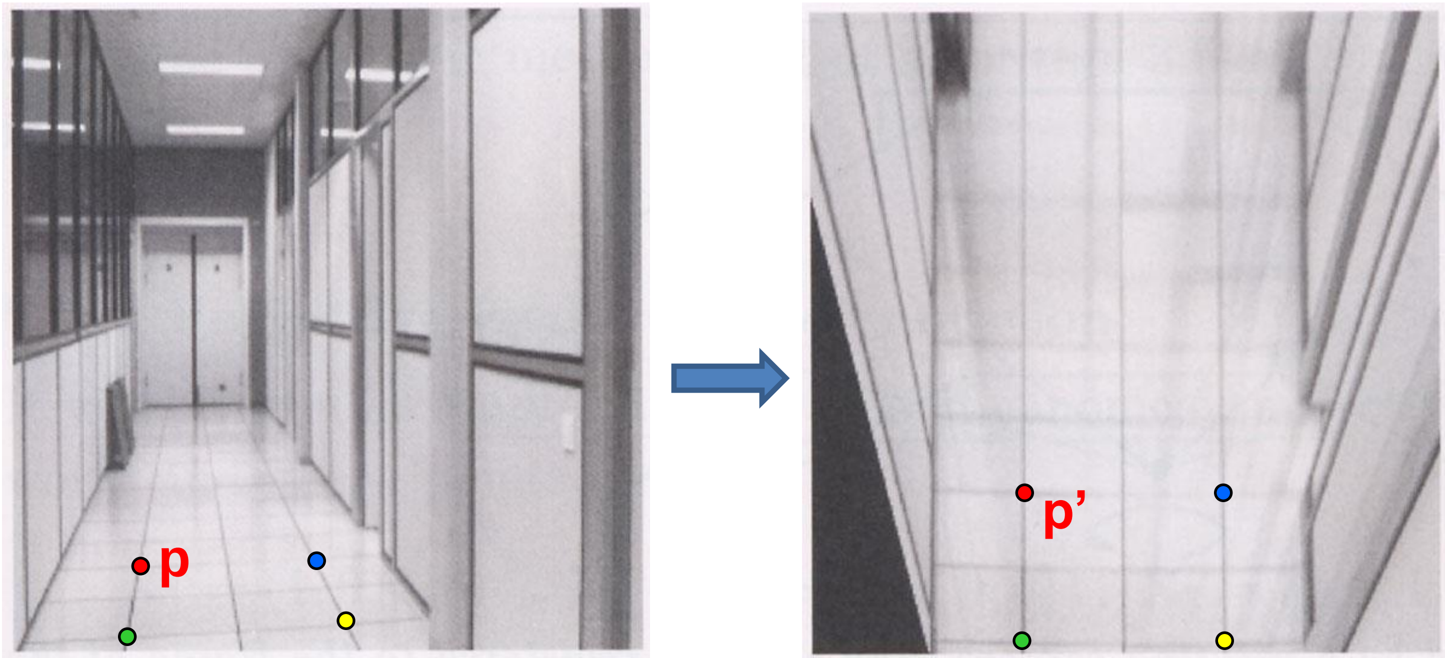
- Definition
  - General mathematics:

    homography = projective linear transformation
  - Vision (most common usage):

    homography = linear transformation between two image planes

- Examples
  - Project 3D surface into frontal view
  - Relate two views that differ only by rotation

Homography example: Image rectification



To unwarp (rectify) an image solve for homography **H** given **p** and **p'**: w**p'=Hp**

Image Stitching Algorithm Overview

1. Detect keypoints (e.g., SIFT)
2. Match keypoints (e.g., $1^{st}/2^{nd}$ NN < thresh)
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 **H**?

Direct Linear Transformation (DLT)

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

$$\mathbf{x}' = \begin{bmatrix} w'u' \\ w'v' \\ w' \end{bmatrix}$$

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

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

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

Computing homography

Direct Linear Transform

$$\begin{bmatrix} -u_1 & -v_1 & -1 & 0 & 0 & 0 & u_1u_1' & v_1u_1' & u_1' \\ 0 & 0 & 0 & -u_1 & -v_1 & -1 & u_1v_1' & v_1v_1' & v_1' \\ & & & & \vdots & & & & \\ 0 & 0 & 0 & -u_n & -v_n & -1 & u_nv_n' & v_nv_n' & v_n' \end{bmatrix} \mathbf{h} = \mathbf{0} \Rightarrow \mathbf{A}\mathbf{h} = \mathbf{0}$$

- Apply SVD: $\boldsymbol{UDV}^T = \boldsymbol{A}$
- $\boldsymbol{h} = \boldsymbol{V}_{\text{smallest}}$ (column of $\boldsymbol{V}$ 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);
```

Explanations of SVD and solving homogeneous linear systems

Computing homography

- Assume we have four matched points: How do we compute homography **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)

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

   – 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

HZ Tutorial ‘99

Computing homography

- Assume we have matched points with outliers: How do we compute homography **H**?

Automatic Homography Estimation with RANSAC

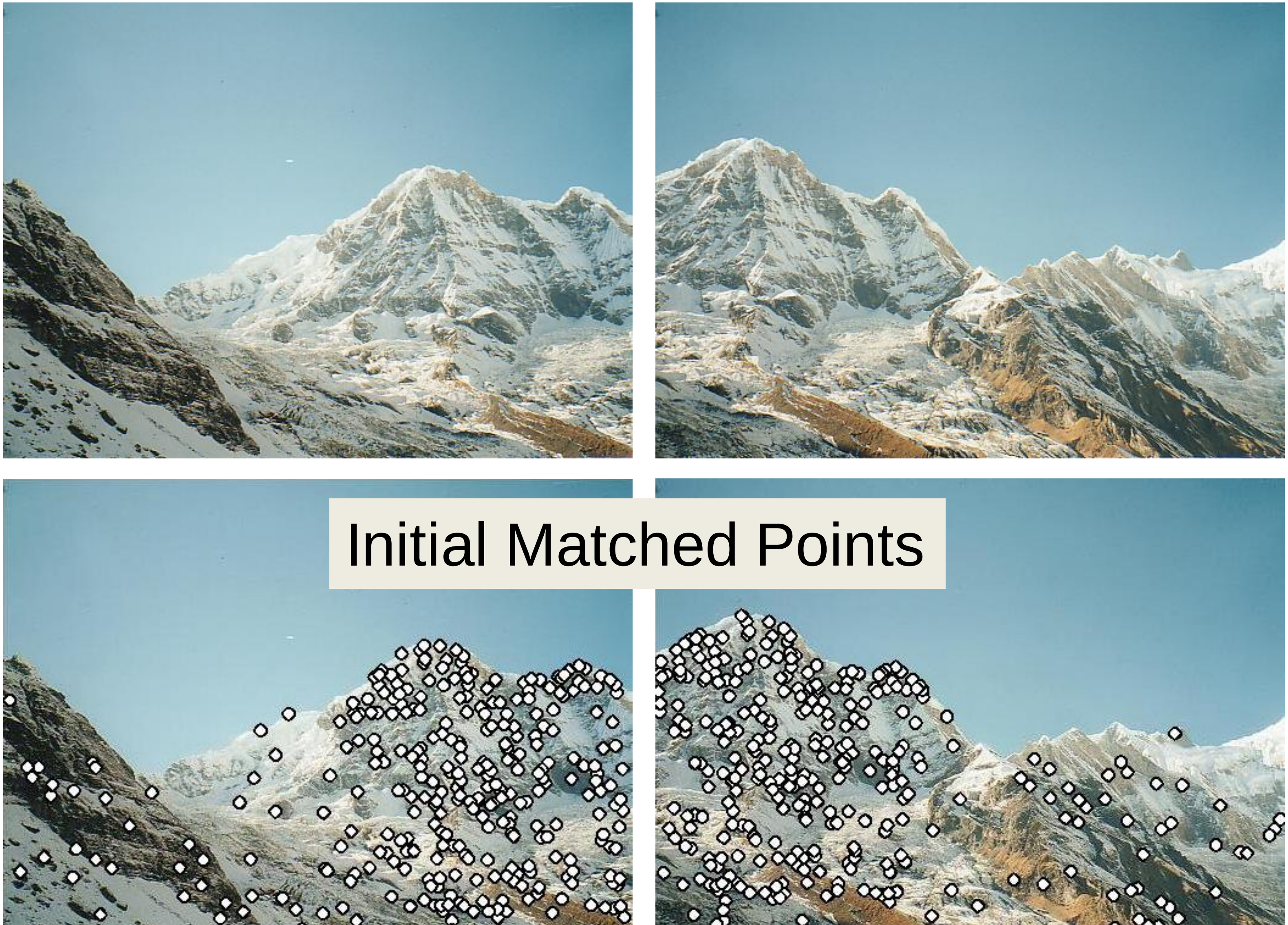
1. Choose number of samples *N*
2. Choose 4 random potential matches
3. Compute **H** using normalized DLT
4. Project points from **x** to **x**' for each potentially matching pair: $\mathbf{x}'_i = \mathbf{H}\mathbf{x}_i$
5. Count points with projected distance $< t$
   - E.g., t = 3 pixels
6. Repeat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

HZ Tutorial '99

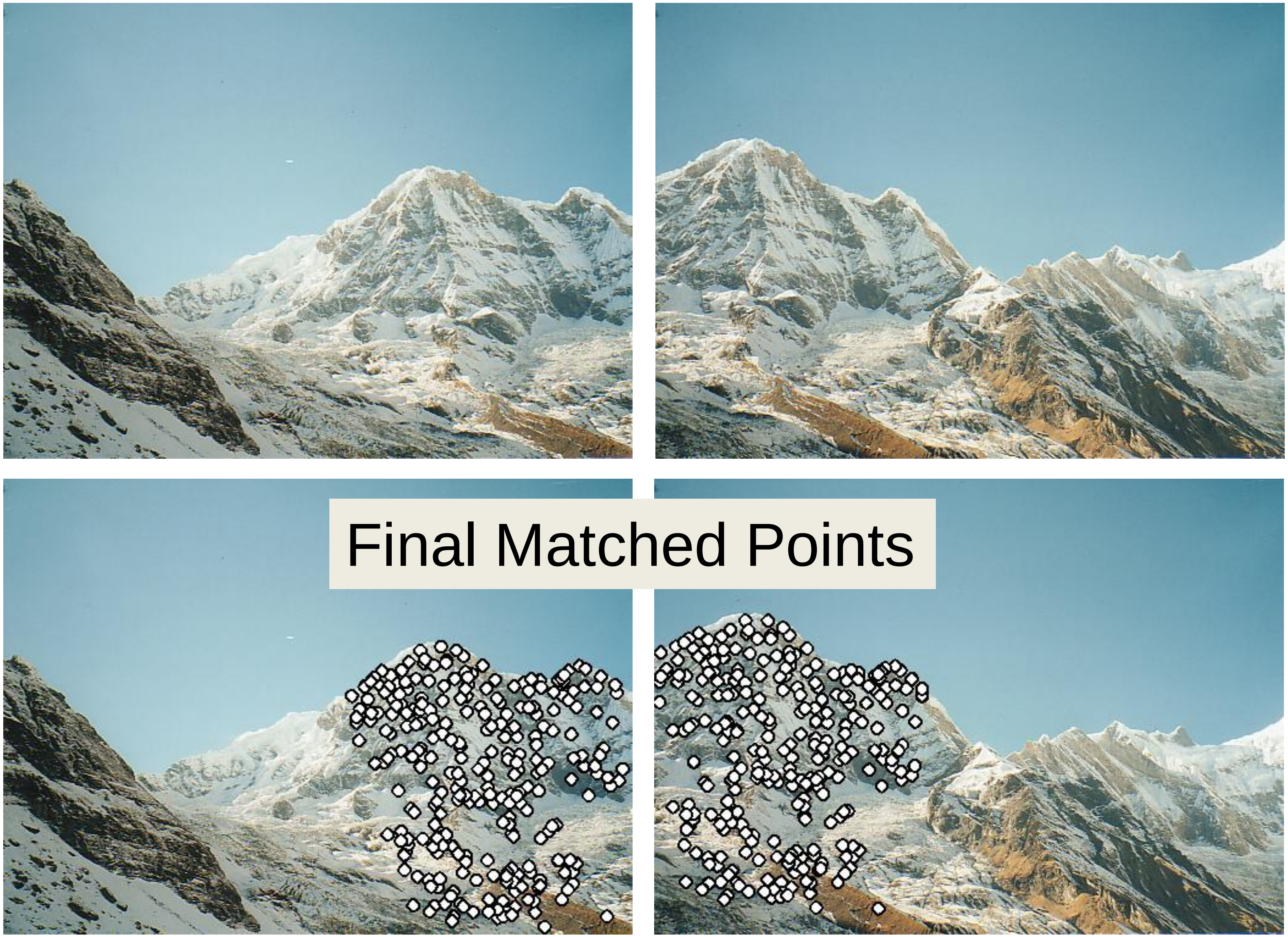
Automatic Image Stitching

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
   - Matlab: maketform, imtransform

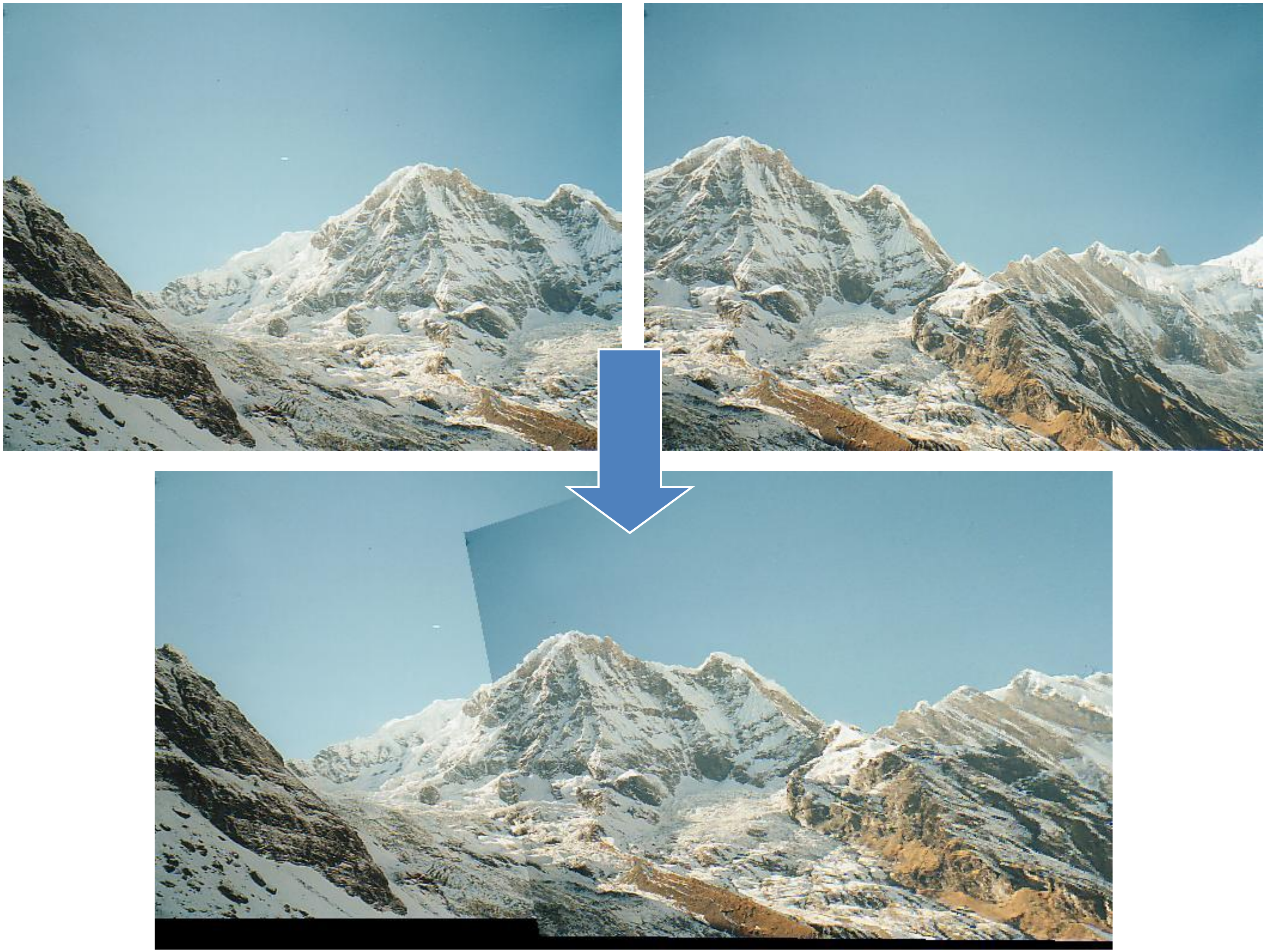
RANSAC for Homography



RANSAC for Homography

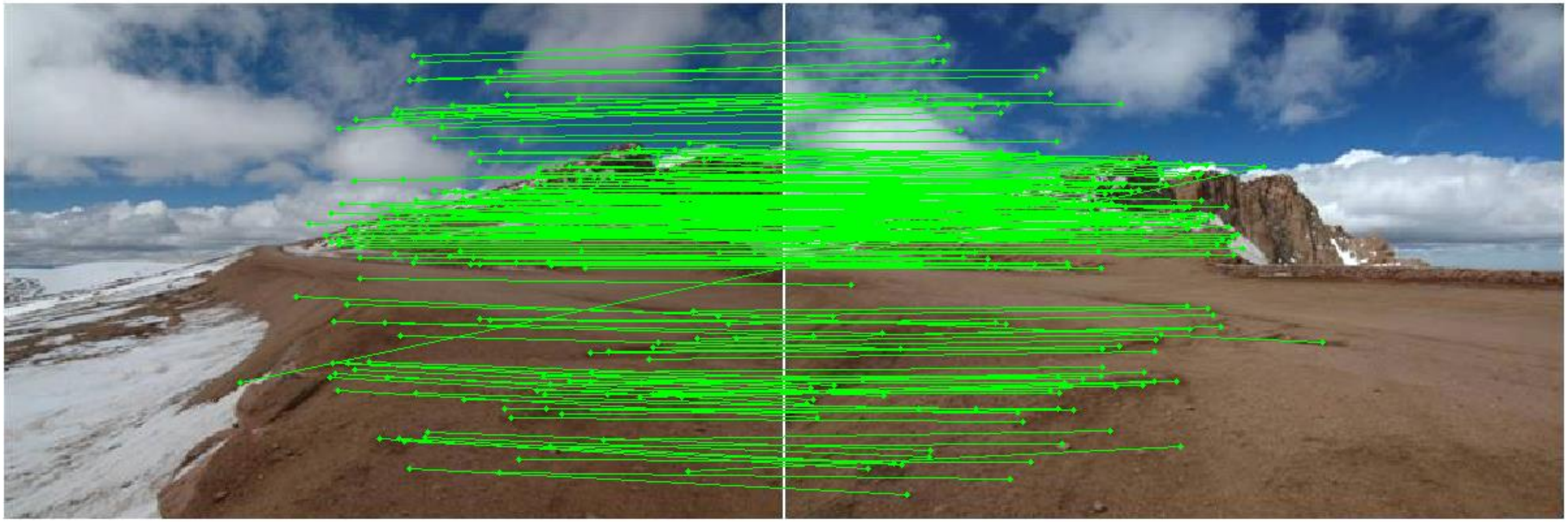


RANSAC for Homography

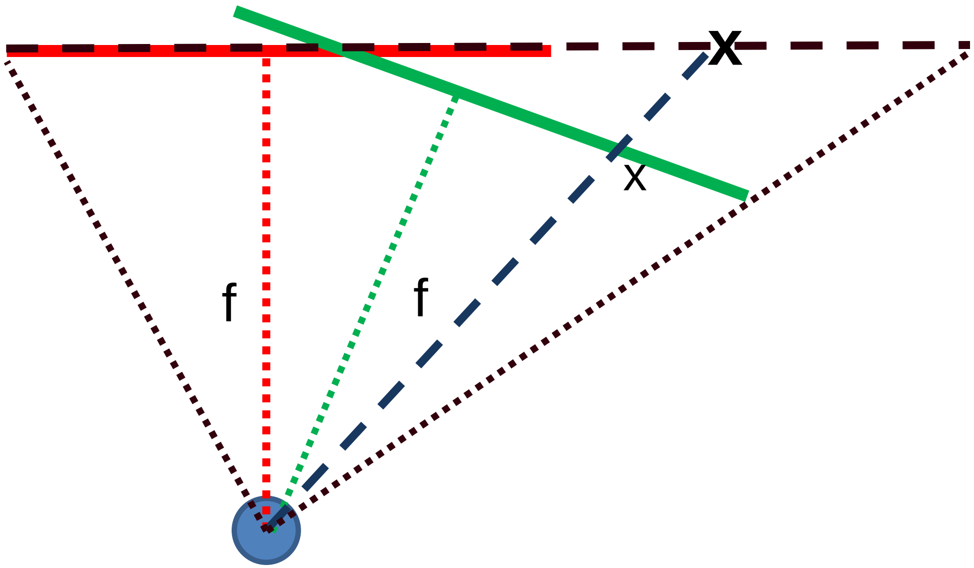


Choosing a Projection Surface

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



Planar Mapping



1) For red image: pixels are already on the planar surface
2) For green image: map to first image plane

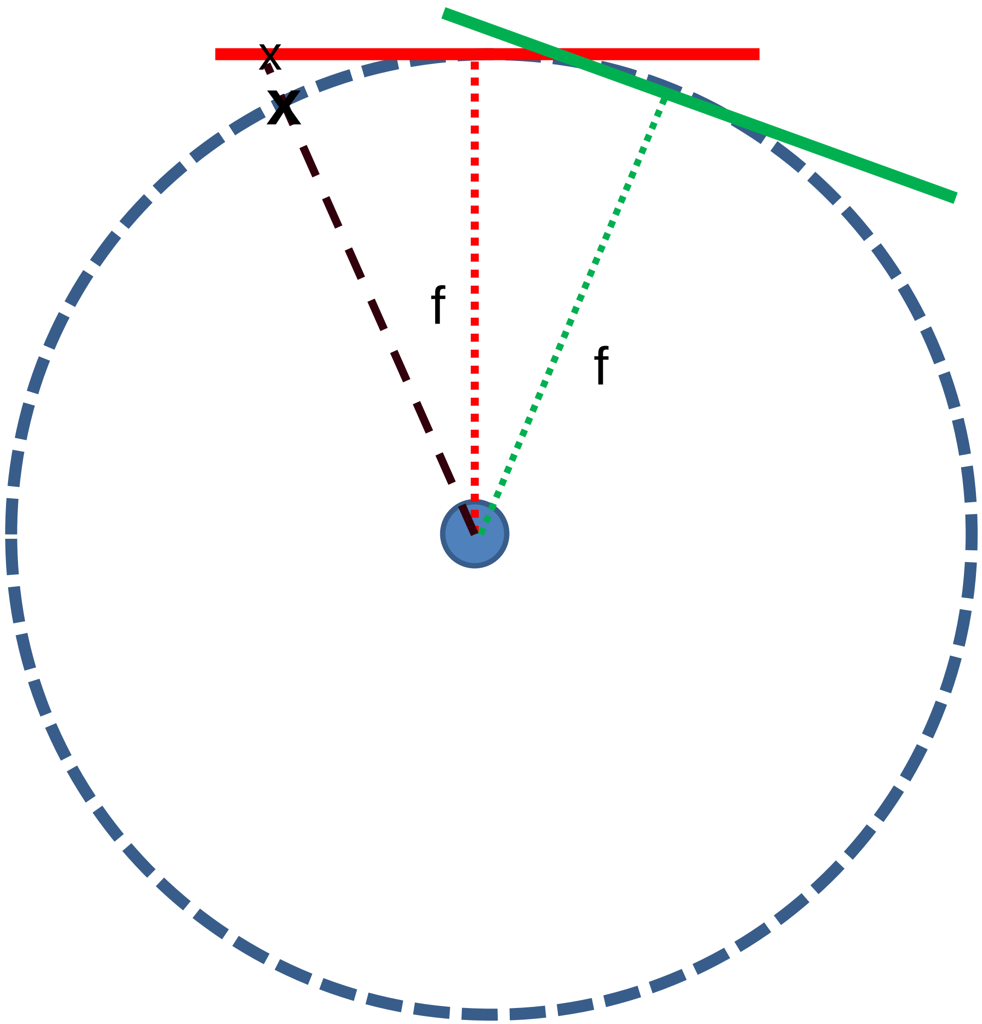
Planar Projection

Photos by Russ Hewett

Planar Projection

Planar

Cylindrical Mapping



1) For red image: compute h, theta on cylindrical surface from (u, v)
2) For green image: map to first image plane, than map to cylindrical surface

Cylindrical Projection

Cylindrical

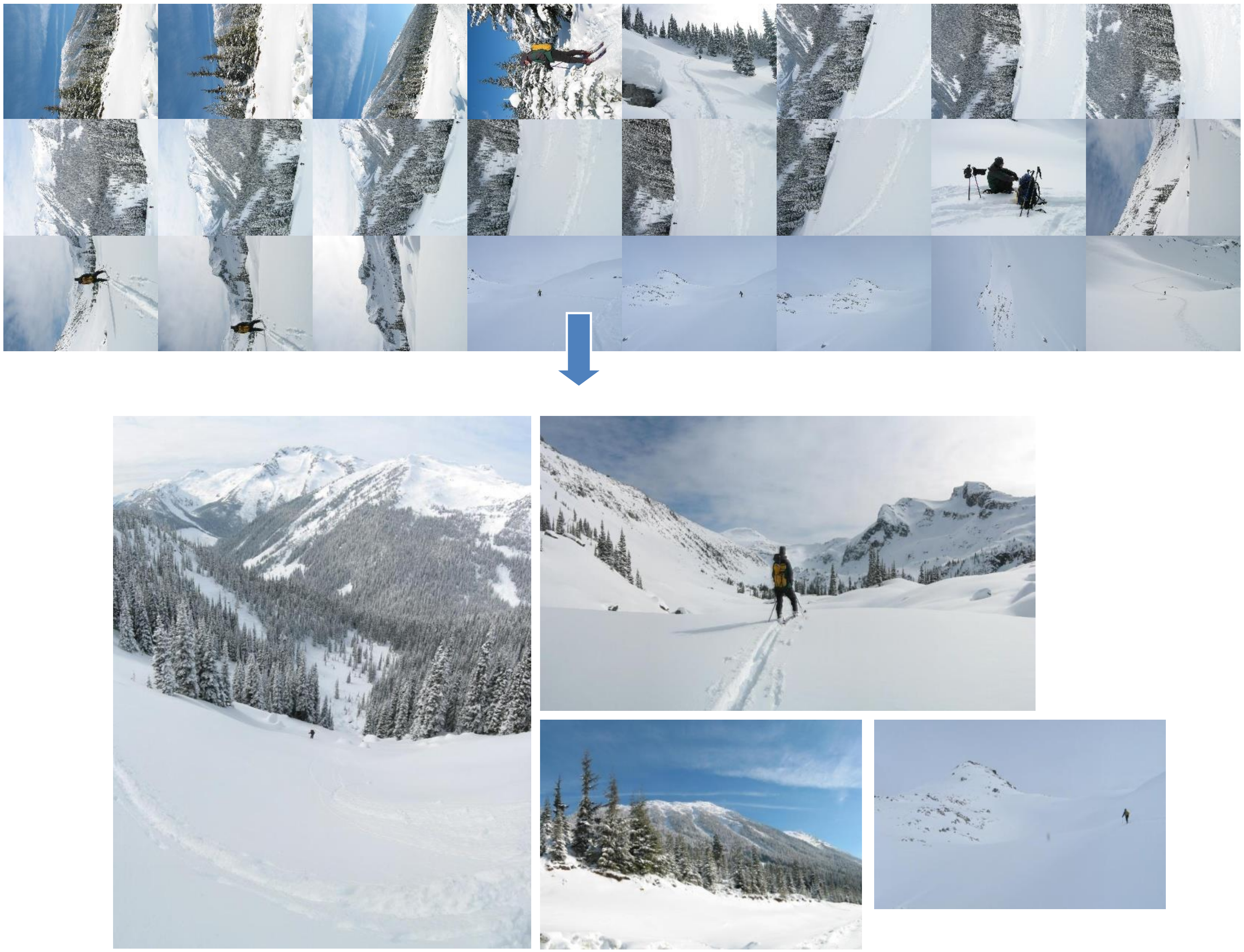
Cylindrical Projection

Cylindrical


Planar


Cylindrical

Recognizing Panoramas



Some of following material from Brown and Lowe 2003 talk

Brown and Lowe 2003, 2007

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

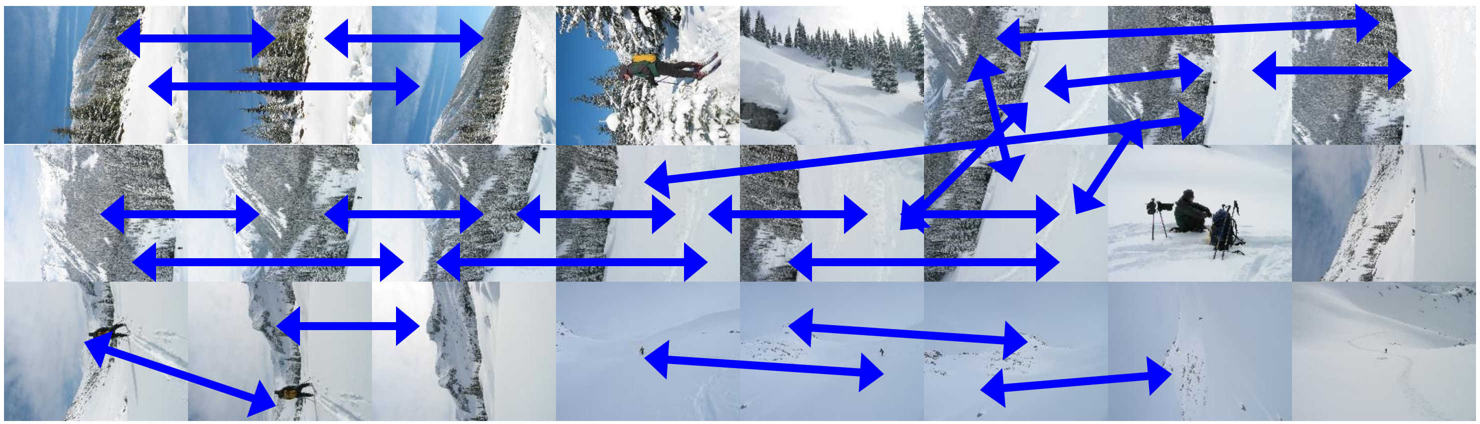
keypoints in overlapping area

Recognizing Panoramas (cont.)

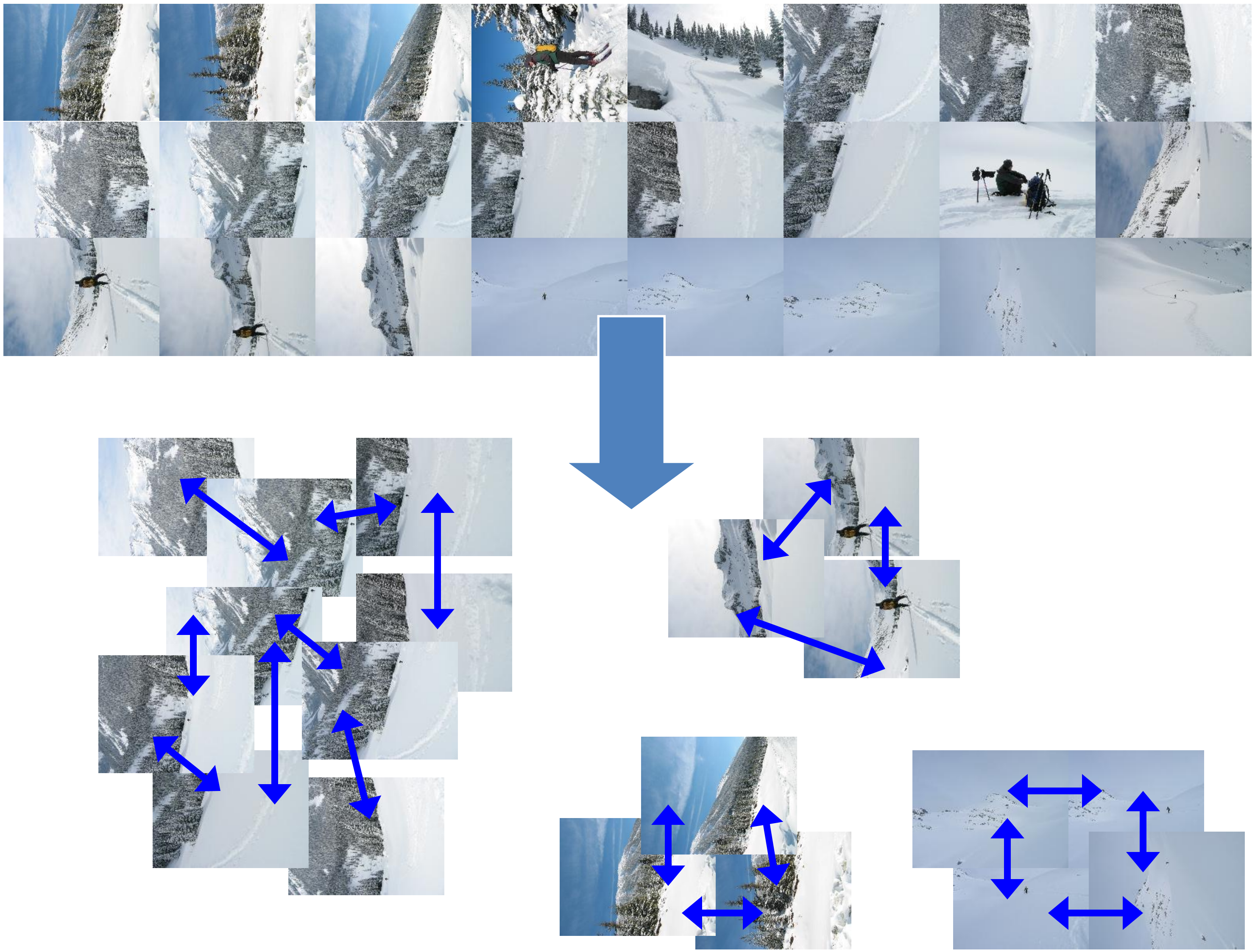
(now we have matched pairs of images)

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 (θ_1, θ_2, θ_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 **H = K' R' R^{-1} 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

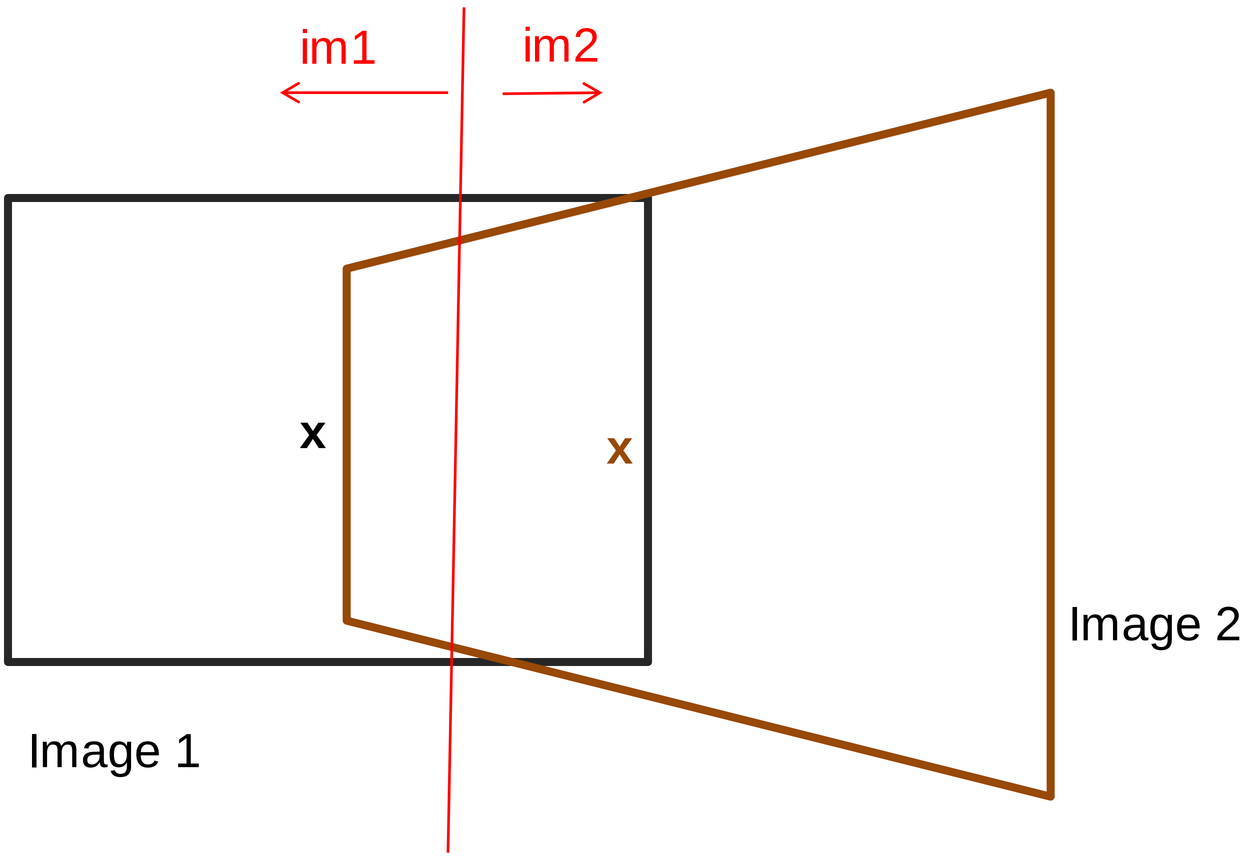
- New images initialised with rotation, focal length of best matching image

Details to make it look good

- Choosing seams
- Blending

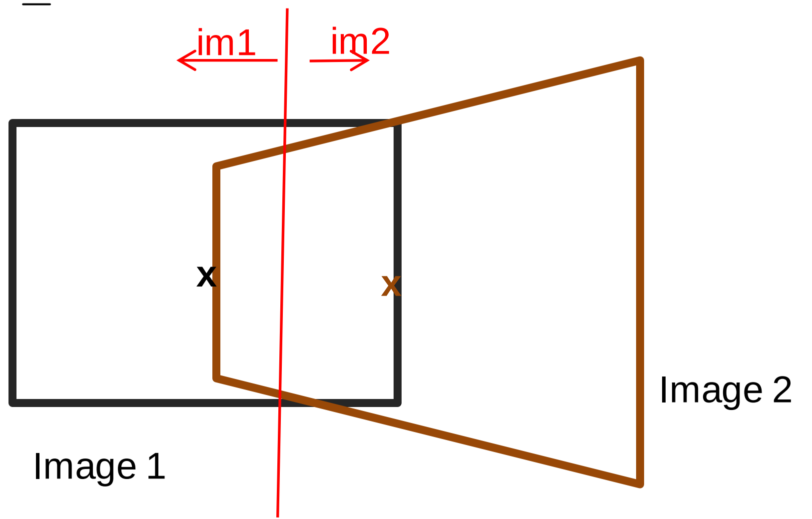
Choosing seams

- Easy method
  - Assign each pixel to image with nearest center



Choosing seams

- Easy method
  - Assign each pixel to image with nearest center
  - Create a mask:
    - `mask(y, x) = 1` iff pixel should come from im1
  - Smooth boundaries (called “feathering”):
    - `mask_sm = imfilter(mask, gausfil);`
  - Composite
    - `imblend = im1_c.*mask + im2_c.*(1-mask);`



Choosing seams

- Better method: dynamic program to find seam along well-matched regions

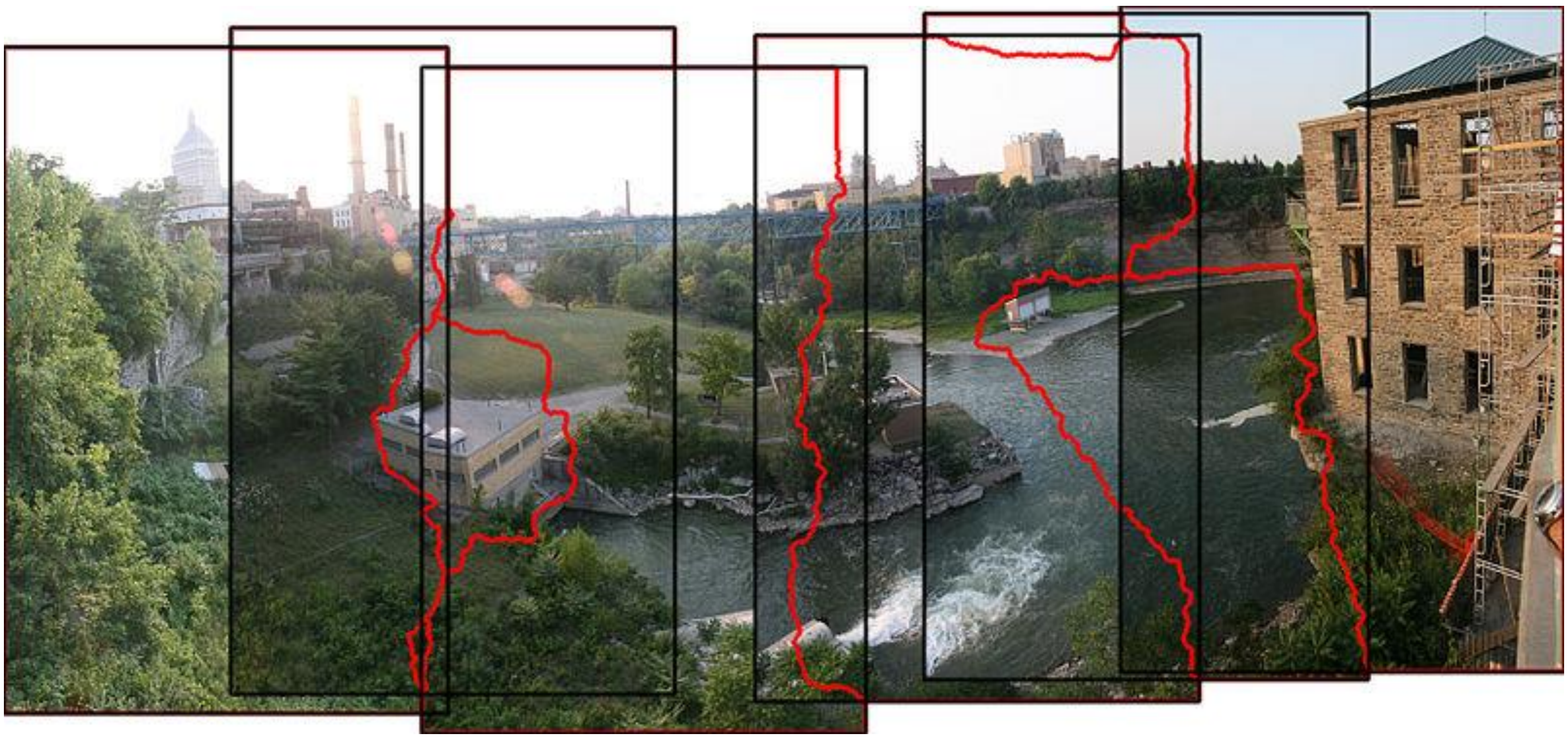
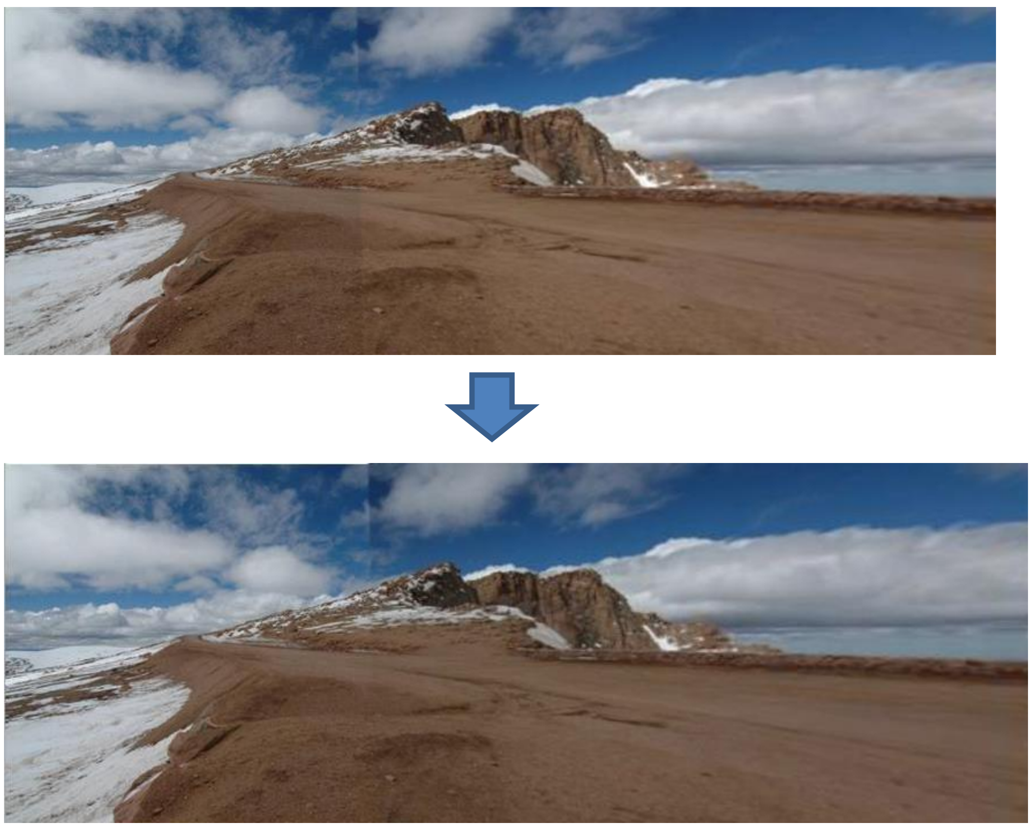
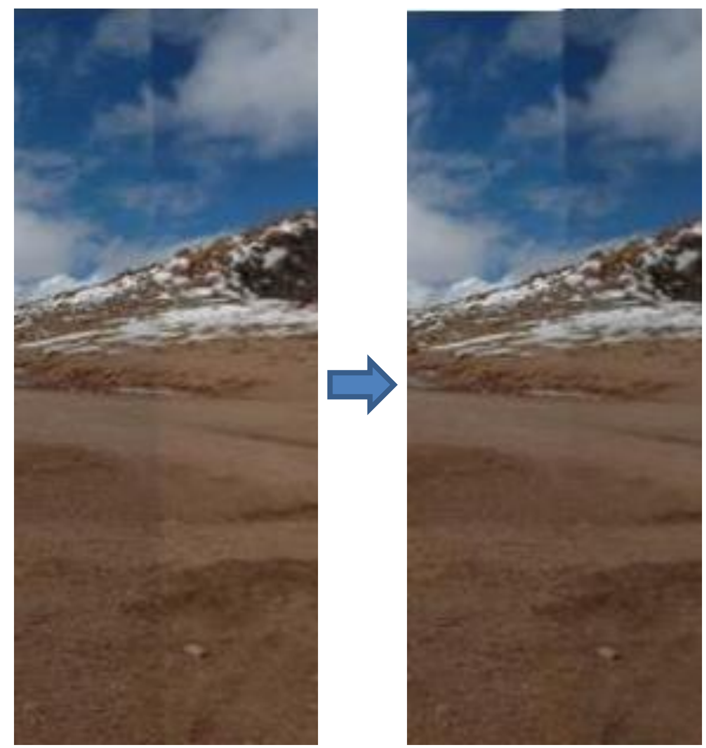


Illustration: http://en.wikipedia.org/wiki/File:Rochester_NY.jpg

Gain compensation

- Simple gain adjustment
  - Compute average RGB intensity of each image in overlapping region
  - Normalize intensities by ratio of averages



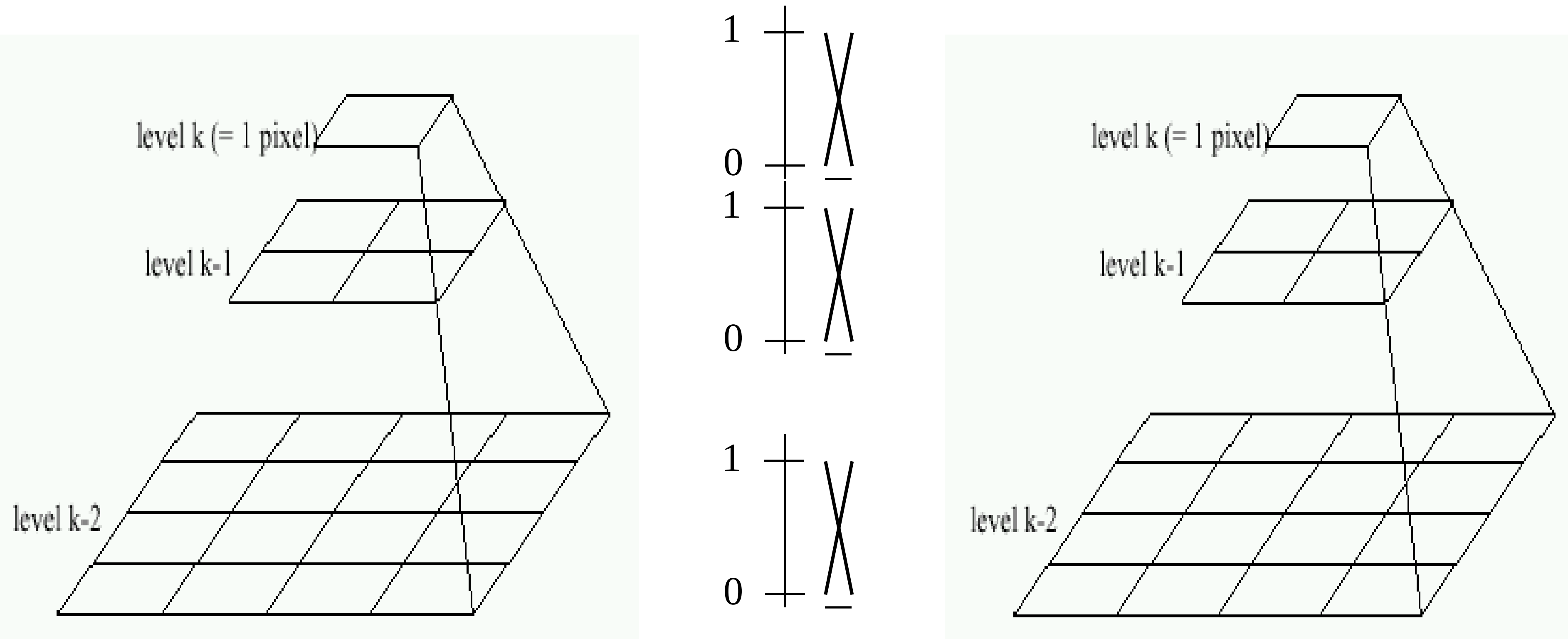


Multi-band Blending

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

Multiband Blending with Laplacian Pyramid

- At low frequencies, blend slowly
- At high frequencies, blend quickly

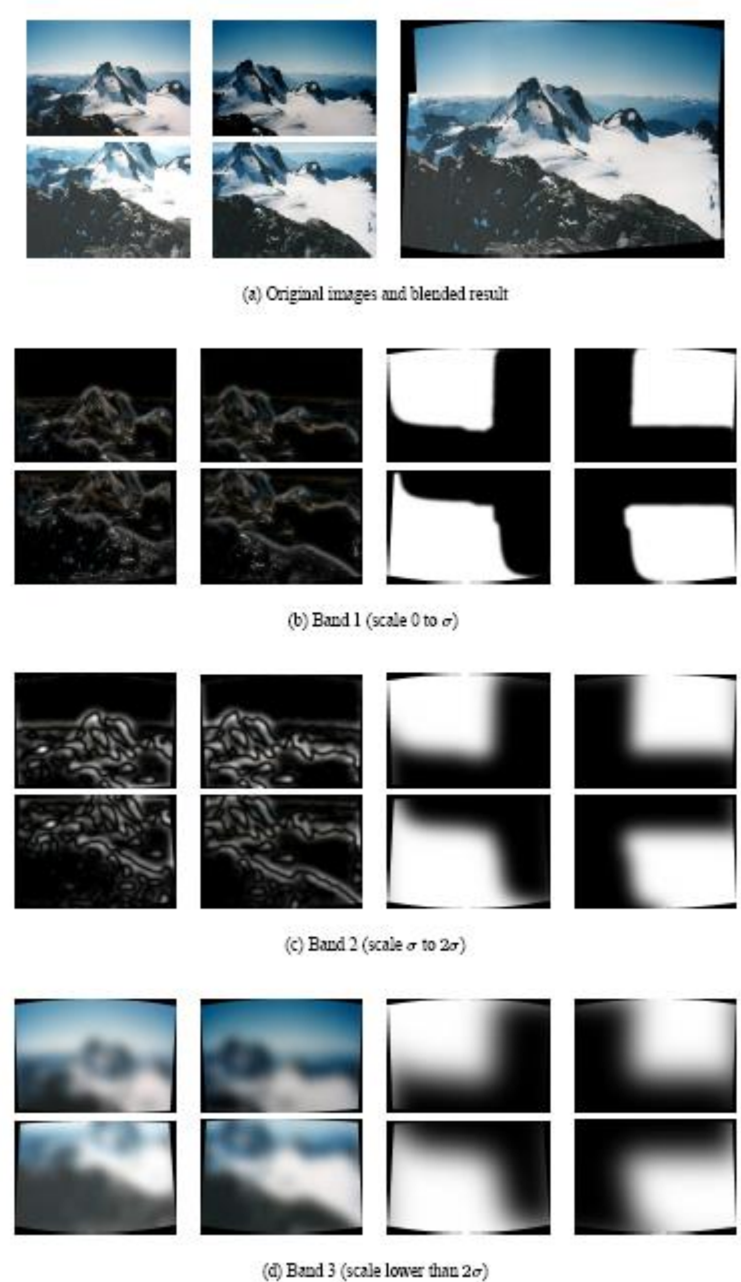


Left pyramid blend Right pyramid

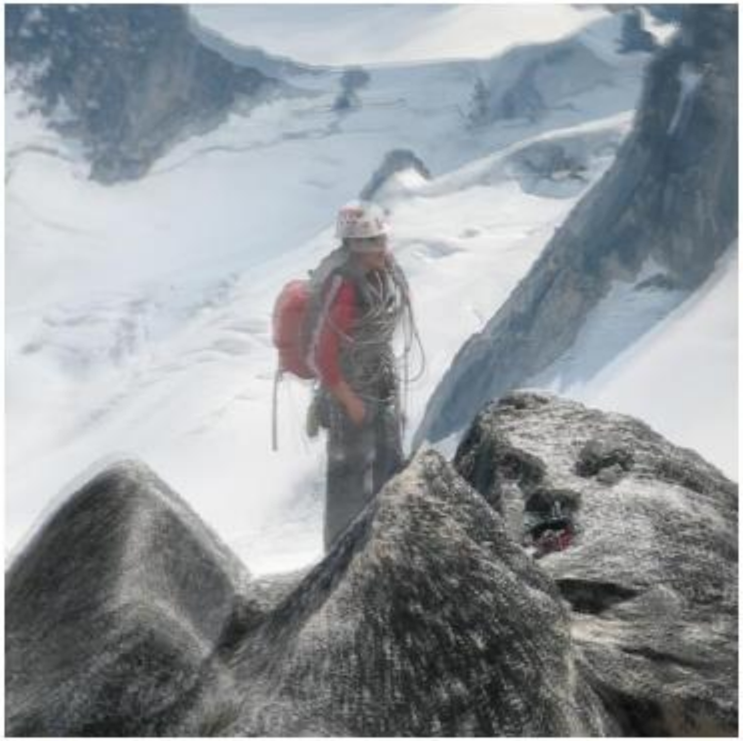
Multiband blending

1. Compute Laplacian pyramid of images and mask
2. Create blended image at each level of pyramid
3. Reconstruct complete image

Laplacian pyramids



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

- Rick Szeliski's alignment/stitching tutorial
- Recognising Panoramas: Brown and Lowe, IJCV 2007 (also bundle adjustment)

How does iphone panoramic stitching work?

- Capture images at 30 fps
- Stitch the central 1/8 of a selection of images
  - Select which images to stitch using the accelerometer and frame-to-frame matching
  - Faster and avoids radial distortion that often occurs towards corners of images
- Alignment
  - Initially, perform cross-correlation of small patches aided by accelerometer to find good regions for matching
  - Register by matching points (KLT tracking or RANSAC with FAST (similar to SIFT) points) or correlational matching
- Blending
  - Linear (or similar) blending, using a face detector to avoid blurring face regions and choose good face shots (not blinking, etc)

http://www.patentlyapple.com/patently-apple/2012/11/apples-cool-iphone-5-panorama-app-revealed-in-5-patents.html

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

- Stereo and epipolar geometry