EE795: Computer Vision and Intelligent Systems

Spring 2012
TTh 17:30-18:45 FDH 204

Lecture 12
130228

http://www.ee.unlv.edu/~b1morris/ecg795/

Outline

- Review
  - Panoramas, Mosaics, Stitching
- Two View Geometry Basics
- Epipolar Geometry
- Planar Homography
- Stereo

Image Mosaics

-  +  + … +  =

Goal: Stitch together several images into a seamless composite

Image Stitching

1. Align the images over each other
   - camera pan ↔ translation on cylinder
2. Blend the images together

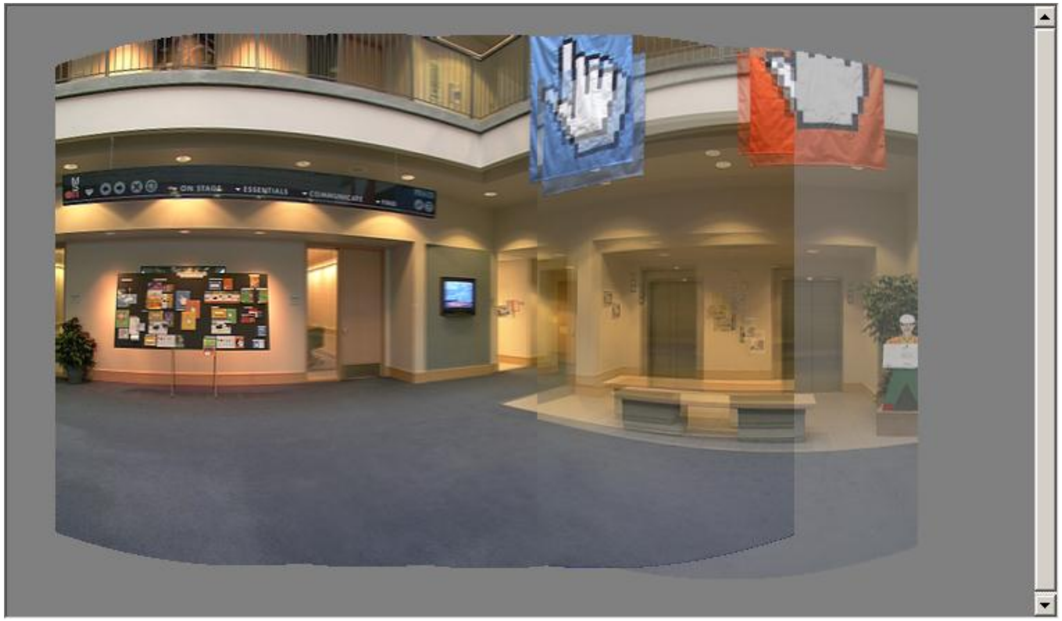
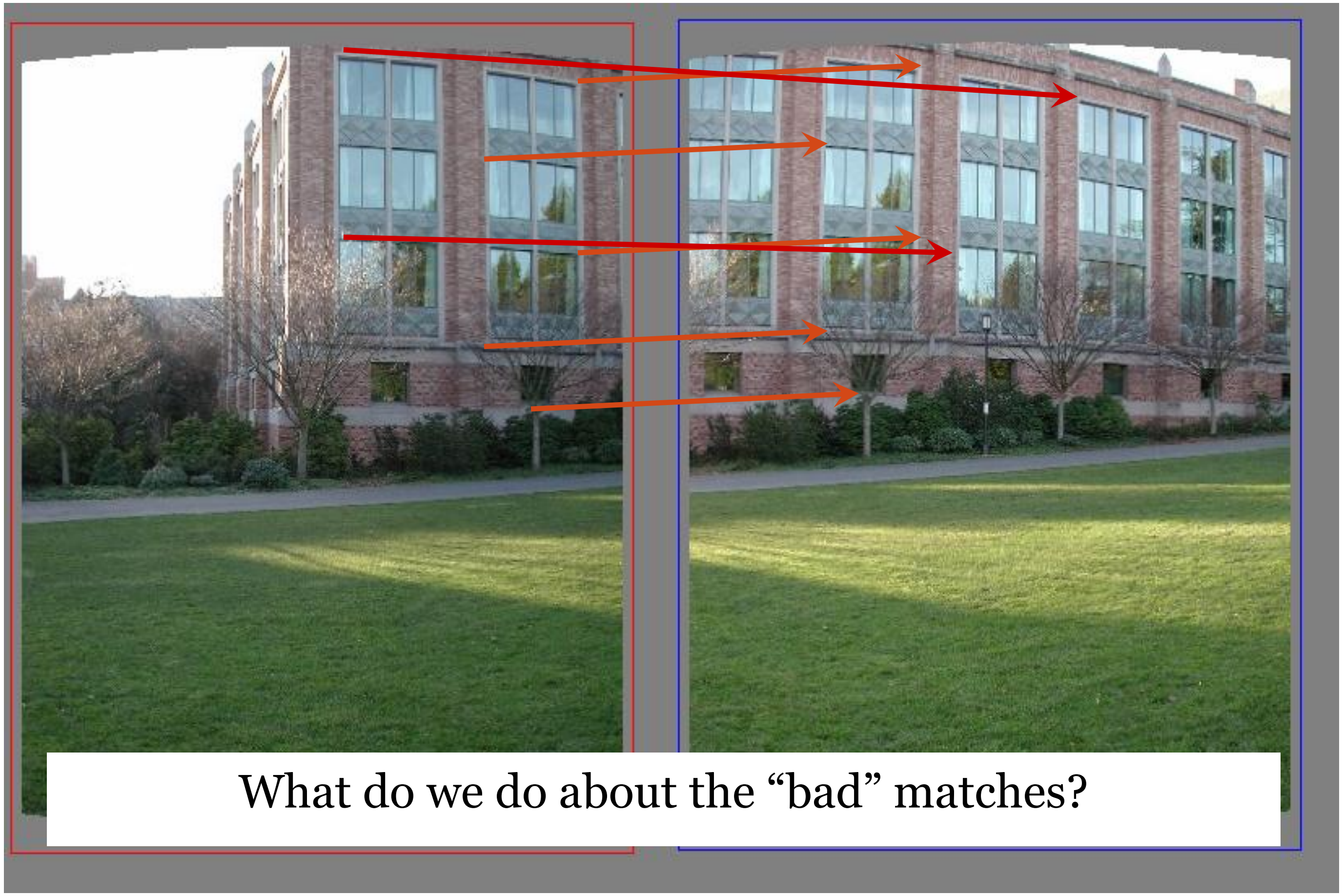


Image stitching steps

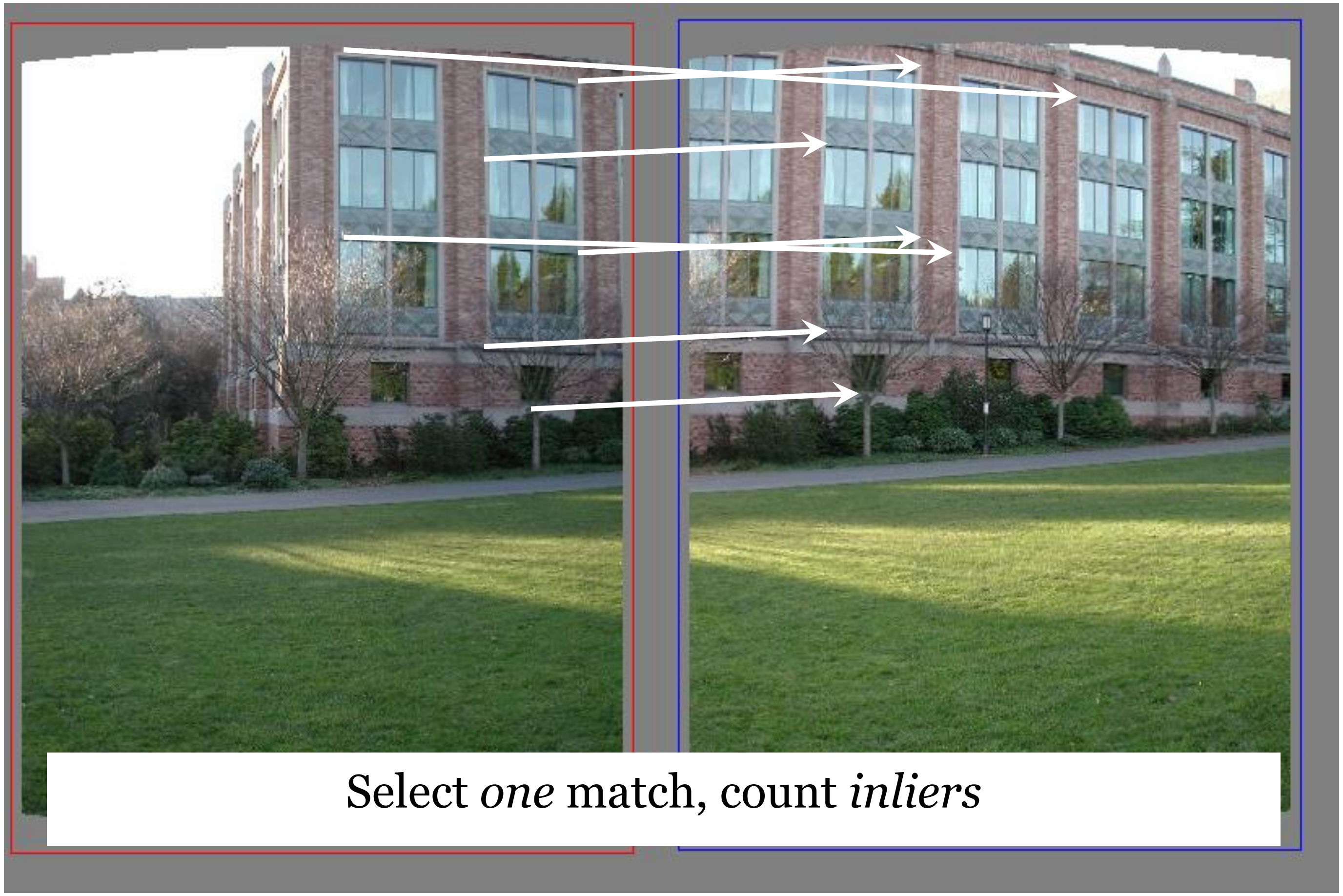
1. Take pictures on a tripod (or handheld)
2. Warp images to spherical coordinates
3. Extract features
4. Align neighboring pairs using RANSAC
5. Write out list of neighboring translations
6. Correct for drift
7. Read in warped images and blend them
8. Crop the result and import into a viewer

Matching features

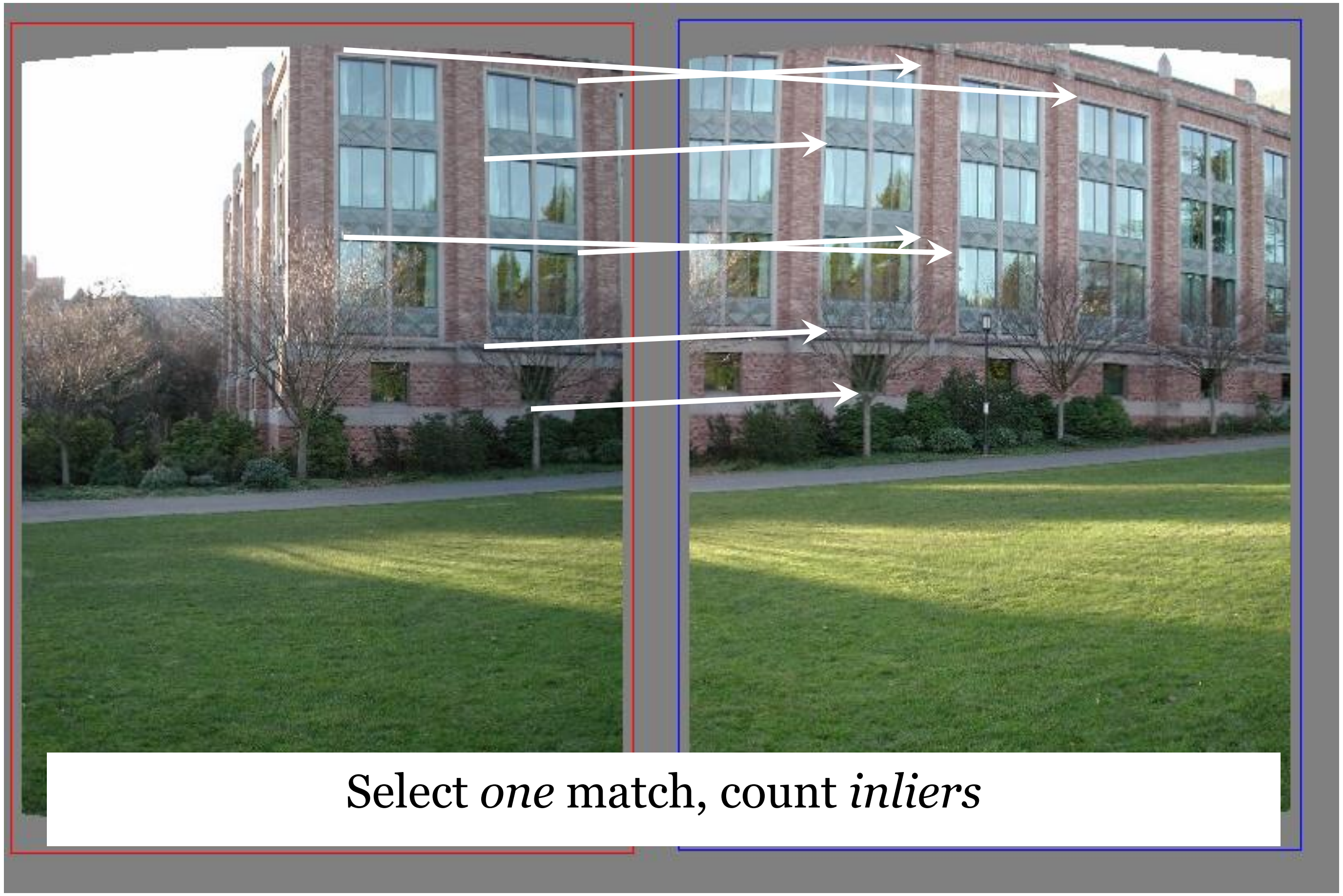


What do we do about the “bad” matches?

RAndom SAmple Consensus

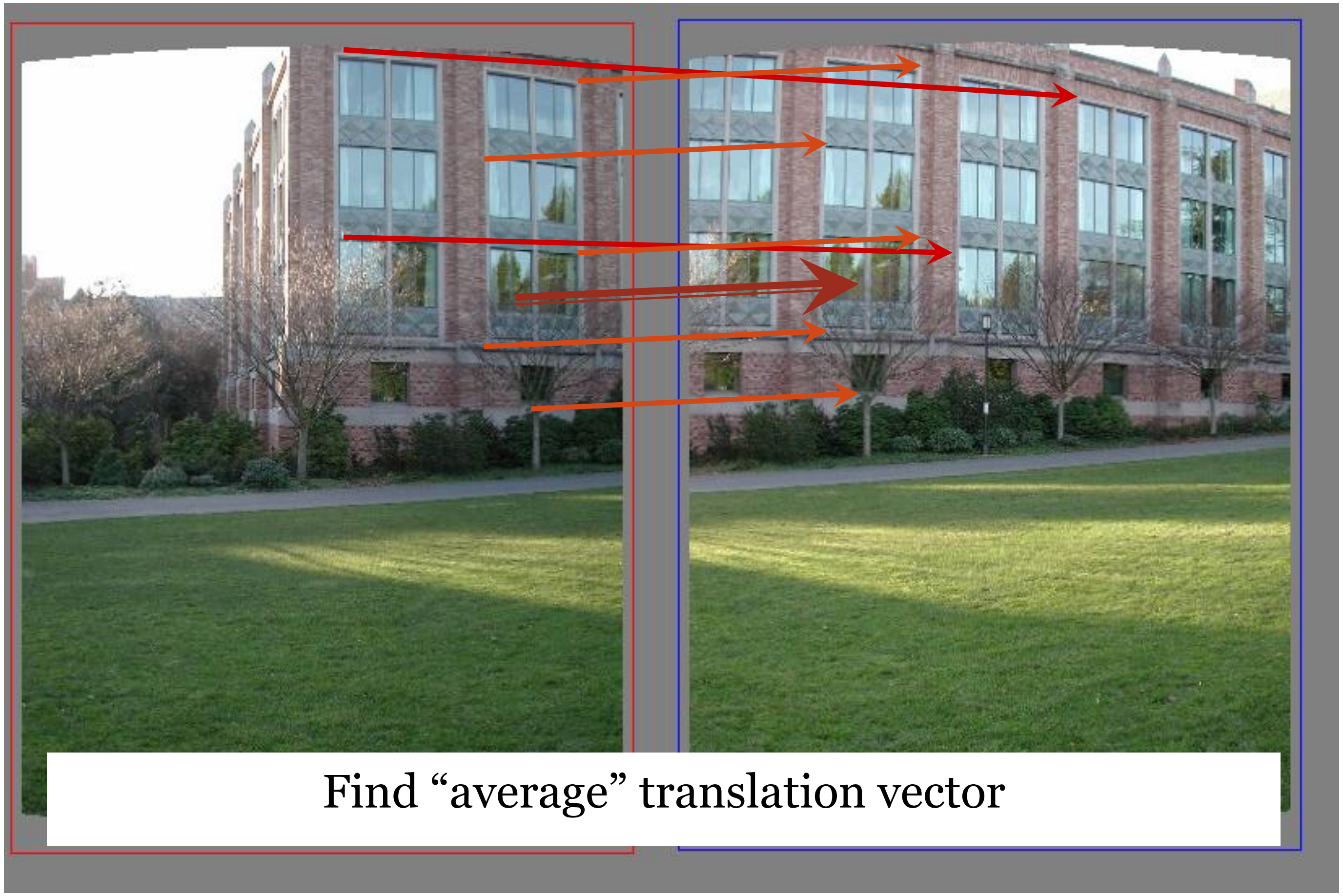


RAndom SAmple Consensus



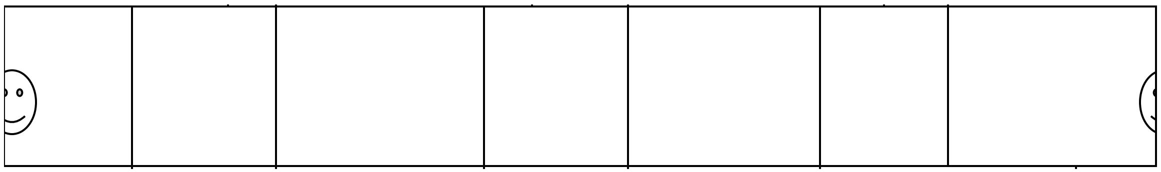
Select *one* match, count *inliers*

Least squares fit



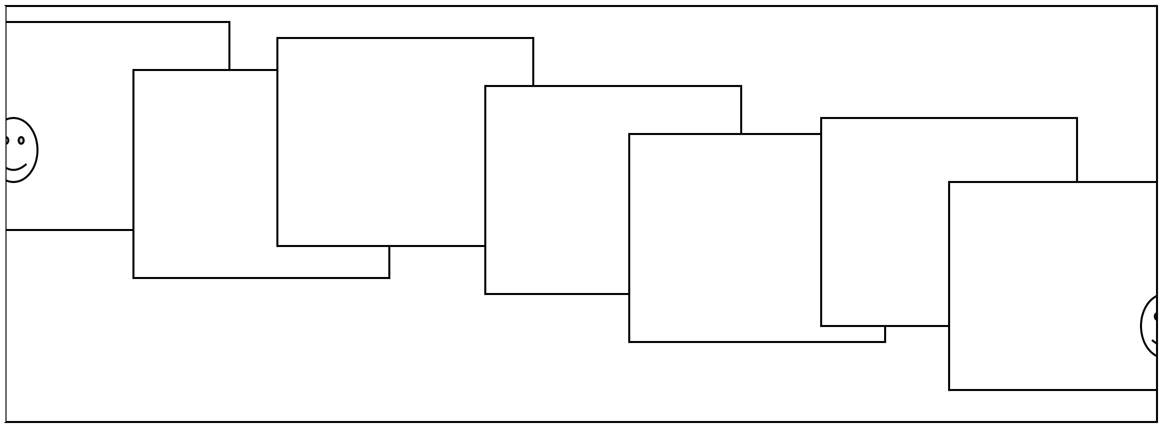
Find “average” translation vector

Assembling the panorama



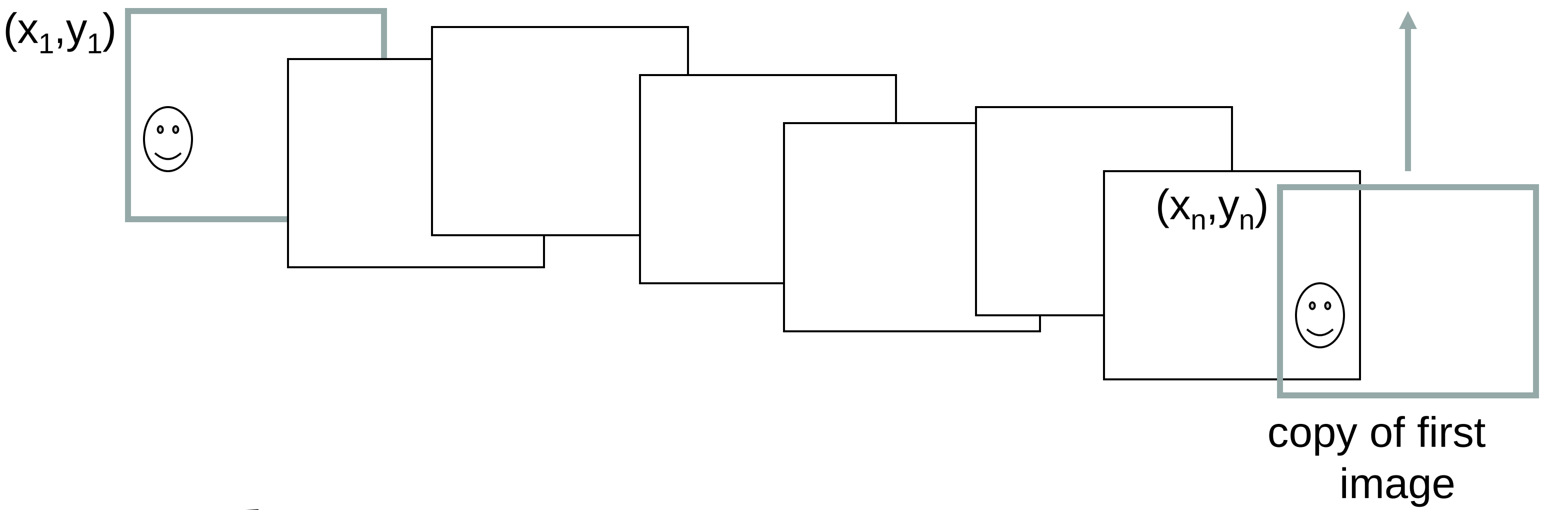
- Stitch pairs together, blend, then crop

Problem: Drift



- Error accumulation
  - small (vertical) errors accumulate over time
  - apply correction so that sum = 0 (for 360° pan.)

Problem: Drift



- Solution
  - add another copy of first image at the end
  - this gives a constraint: $y_n = y_1$
  - 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”

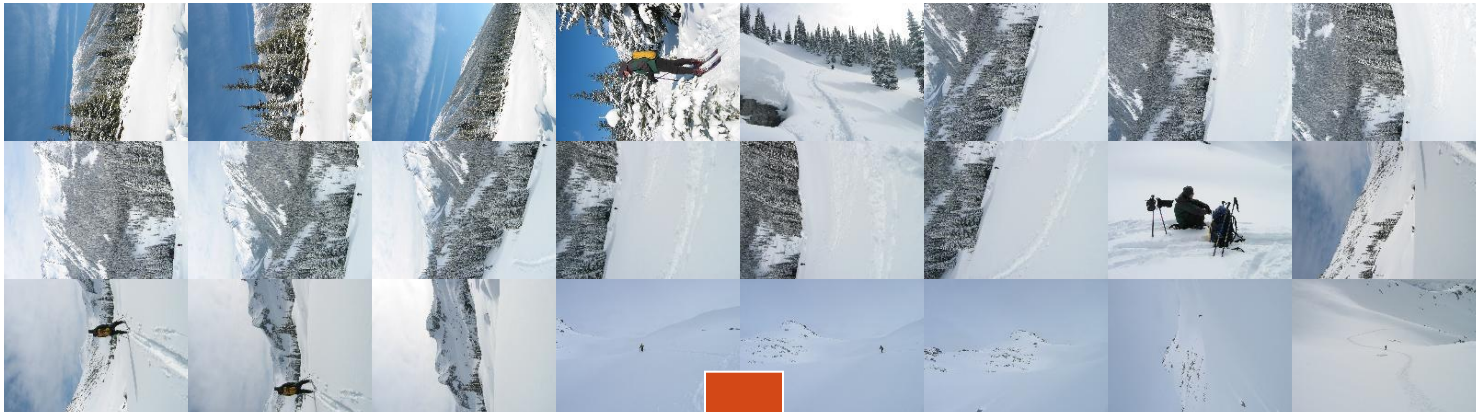
Full-view Panorama

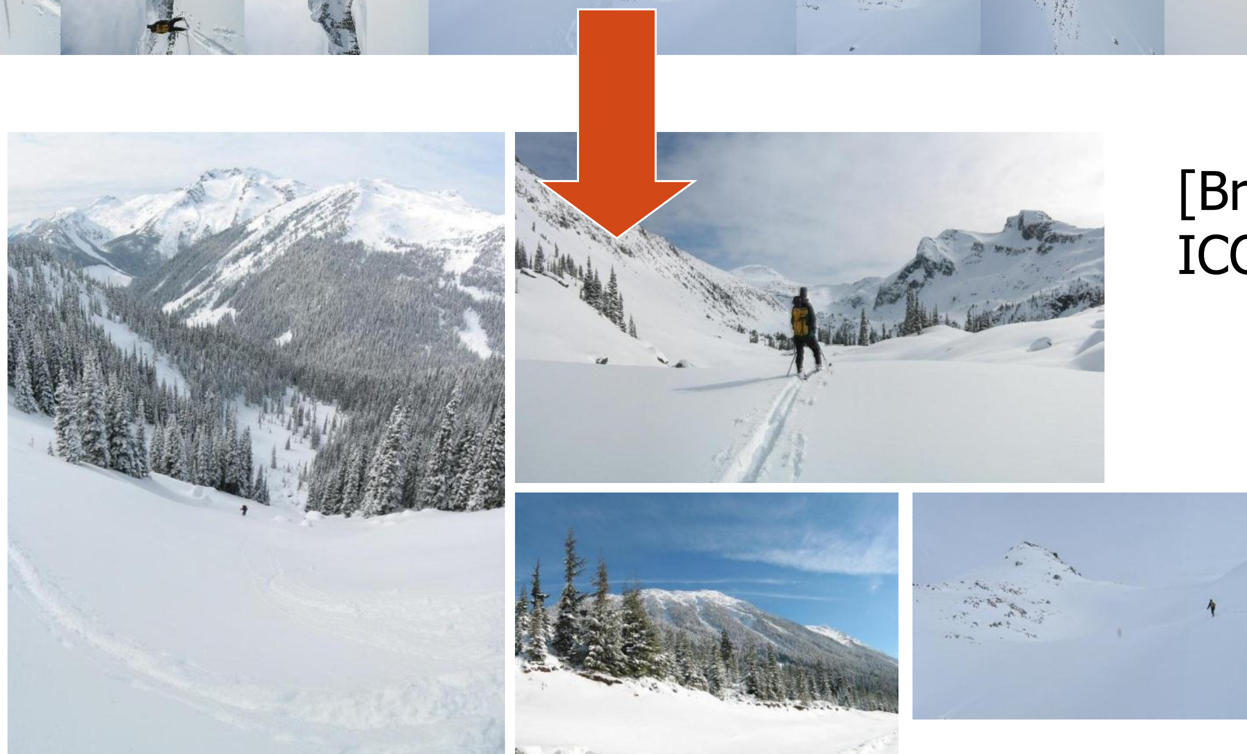
Texture Mapped Model

Global alignment

- Register *all* pairwise overlapping images
- Use a 3D rotation model (one R per image)
- Use direct alignment (patch centers) or feature based
- *Infer* overlaps based on previous matches (incremental)
- Optionally *discover* which images overlap other images using feature selection (RANSAC)

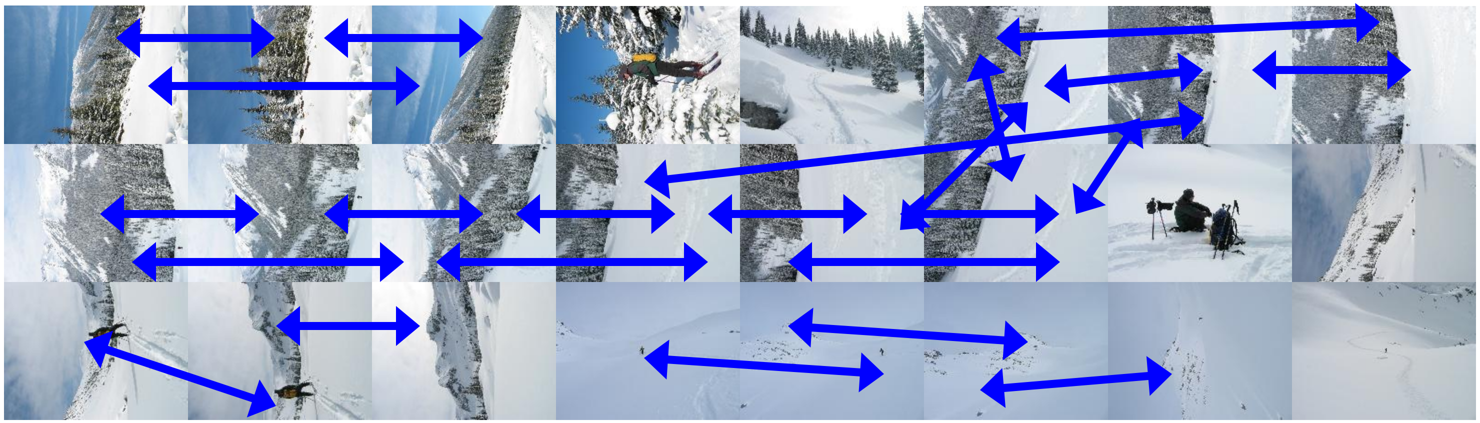
Recognizing Panoramas



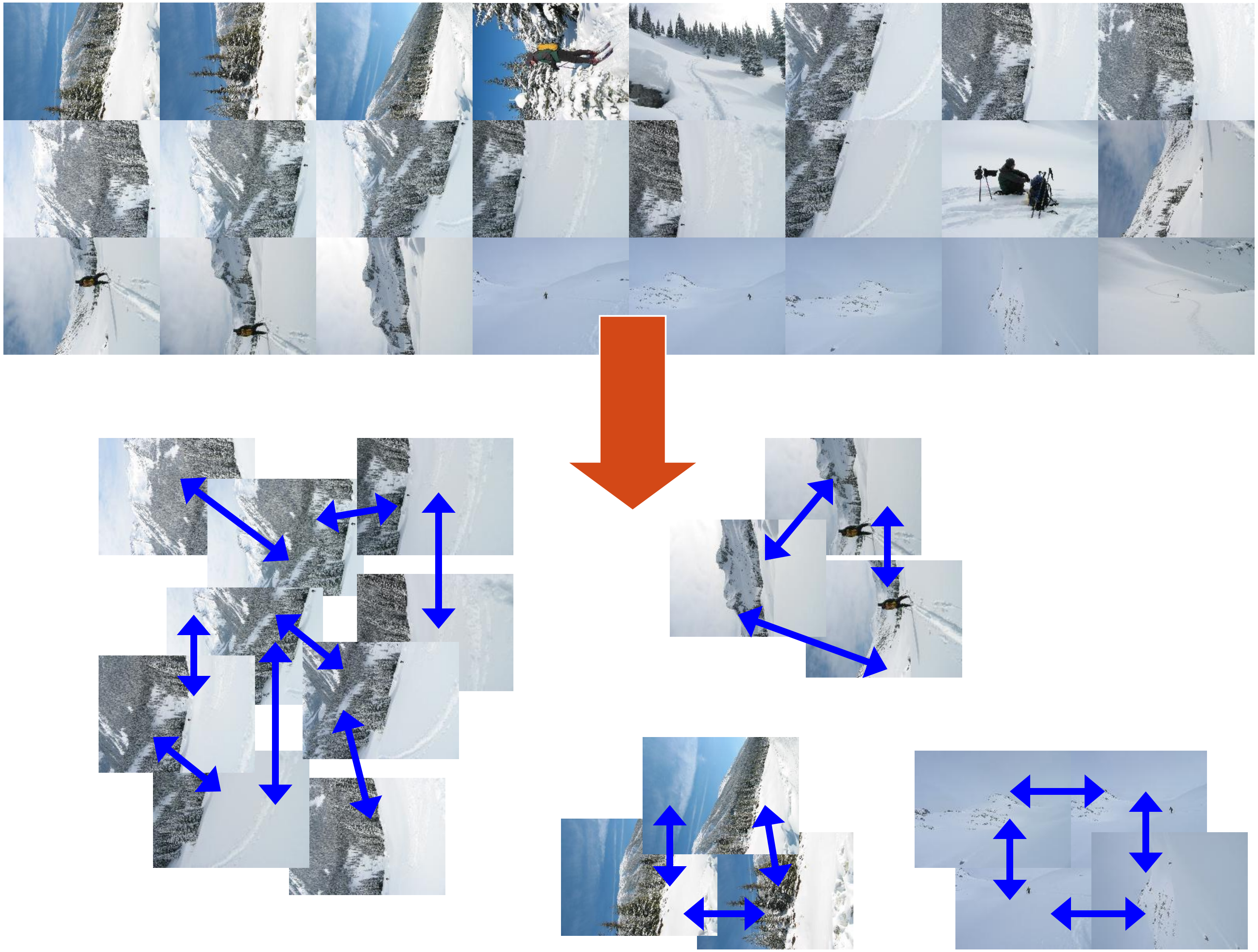


[Brown & Lowe, ICCV’03]

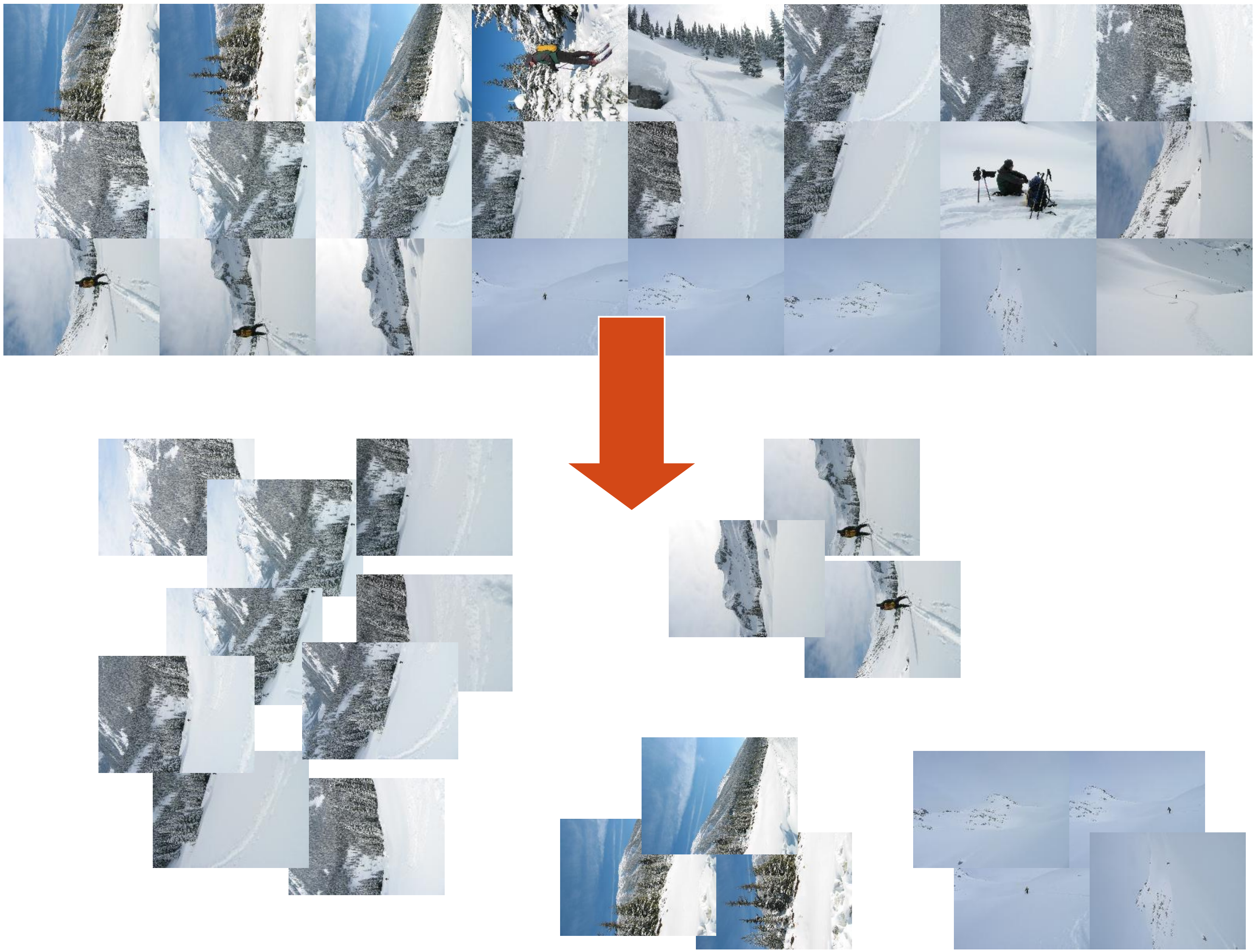
Finding the panoramas



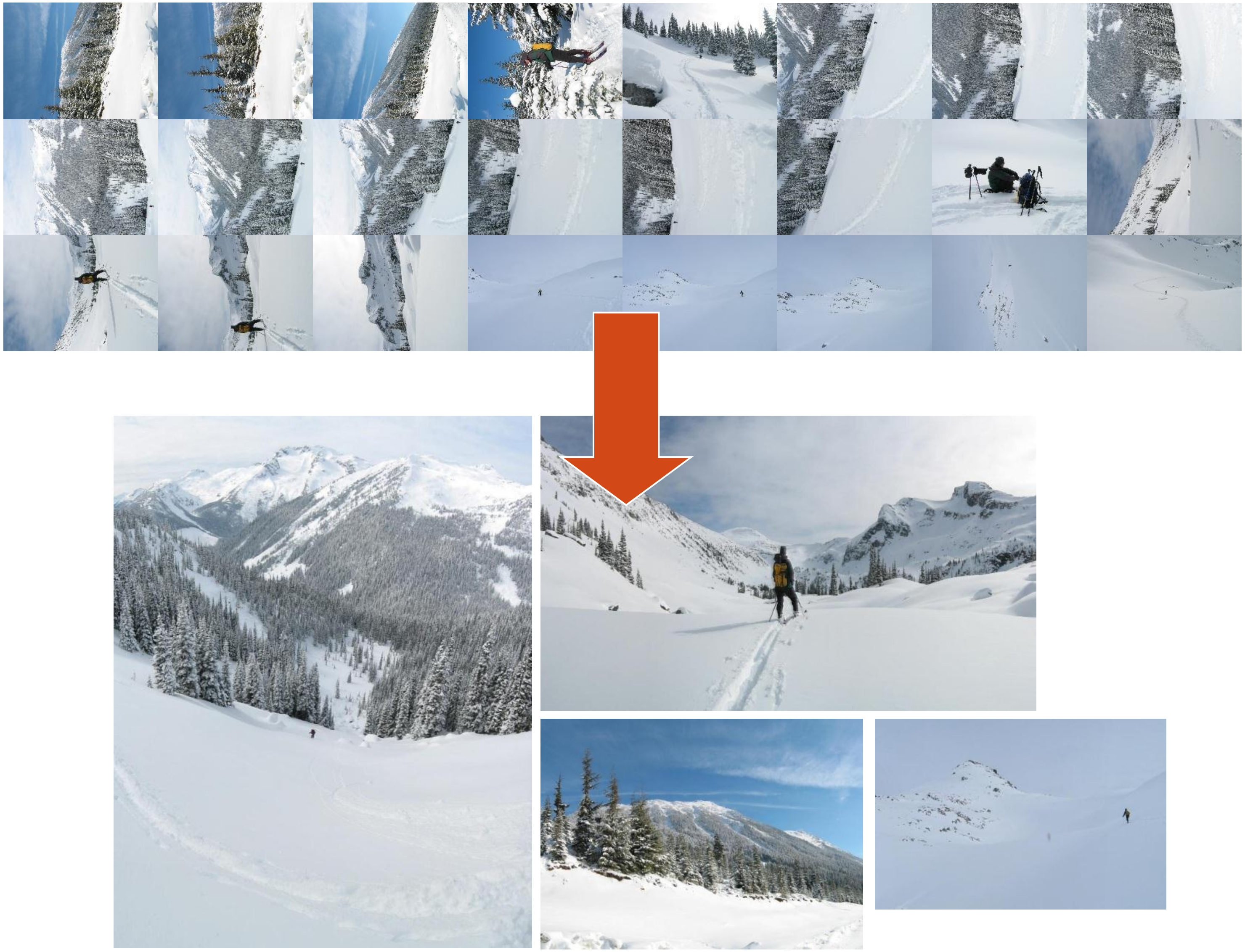
Finding the panoramas



Finding the panoramas



Finding the panoramas



Fully automated 2D stitching

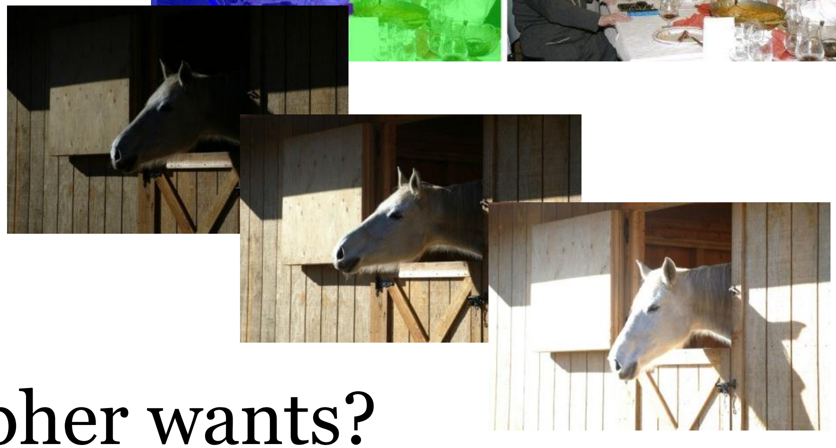
- Free copy from Microsoft Essentials
  - http://windows.microsoft.com/en-us/windows-live/photo-gallery-get-started

Final thought: What is a "panorama"?

- Tracking a subject
  - Panorama

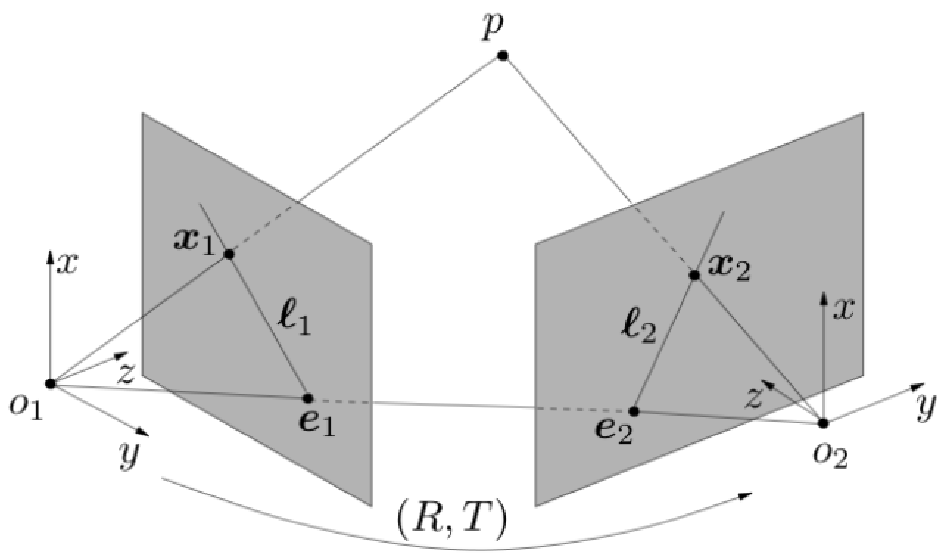
- Repeated (best) shots
  - Photo Fuse

- Multiple exposures
  - Photo Fuse



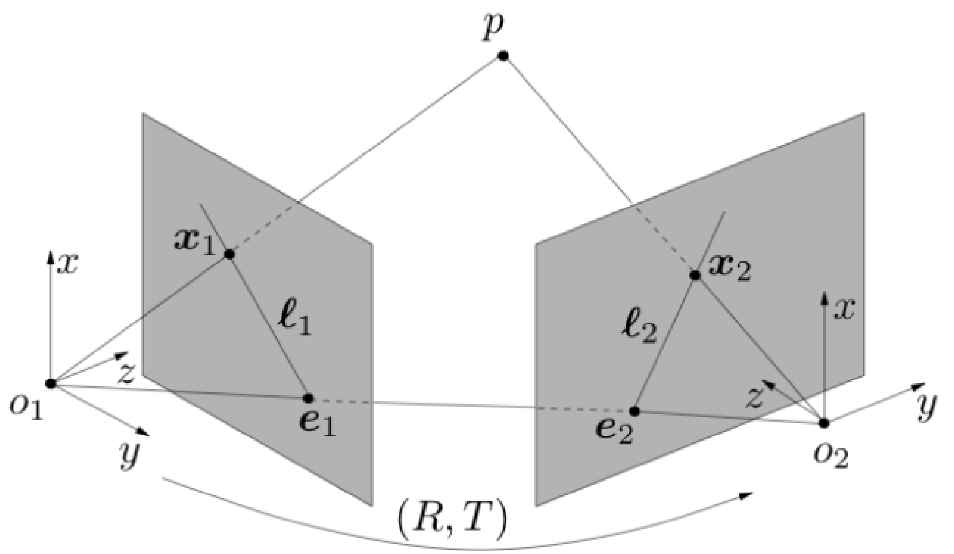
- "Infer" what photographer wants?

Two View Geometry



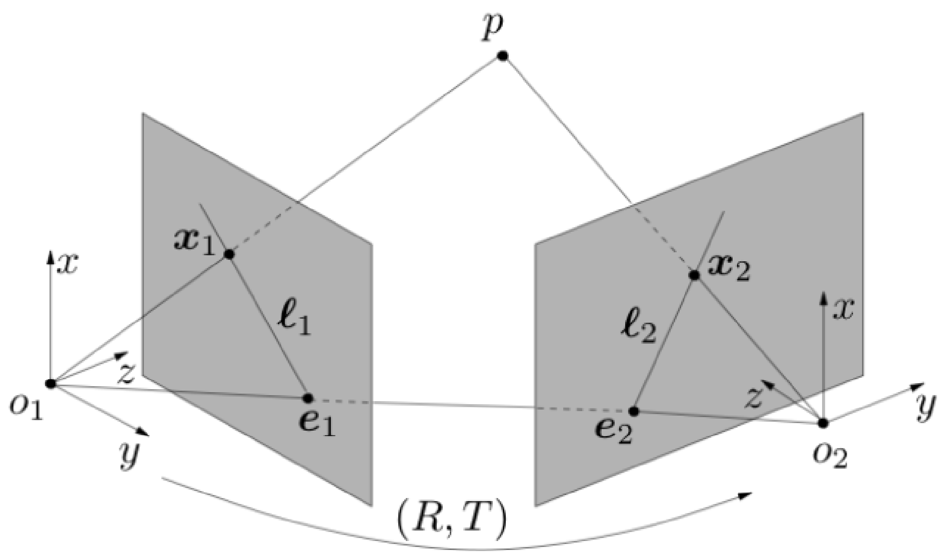
- Basic geometry to relate images of points and their 3d position
  - $X_2 = RX_1 + T$
  - Use triangulation
- Will assume cameras are calibrated
  - Cameras “match”

Epipolar Geometry



- Cameras centered at o_1 and o_2
- Homogeneous vectors e_1, e_2 are known as epipoles
  - Points where baseline pierces the image plane
  - Projection of the other camera's optical center onto the other image plane
  - Translation vector T between cameras

Epipolar Lines



- An point in one image maps to a line in another image

- A plane is spanned by the epipoles, o_1, o_2, and a 3d point p
- l_1, l_2 are epipolar lines
  - Intersection of the p-plane with the image plane
  - Image point x_1 can map anywhere along line l_2
    - Depends on the depth of the point

Rectified Stereo

- Simplest case of two-view geometry
- Camera views are rectified so that there is only a translation between images

- Epipolar lines are horizontal
  - Points in one image map to a horizontal scan line with the same y coordinate in the other image plane

- Simple search and match techniques
  - Can give us disparity (depth) image

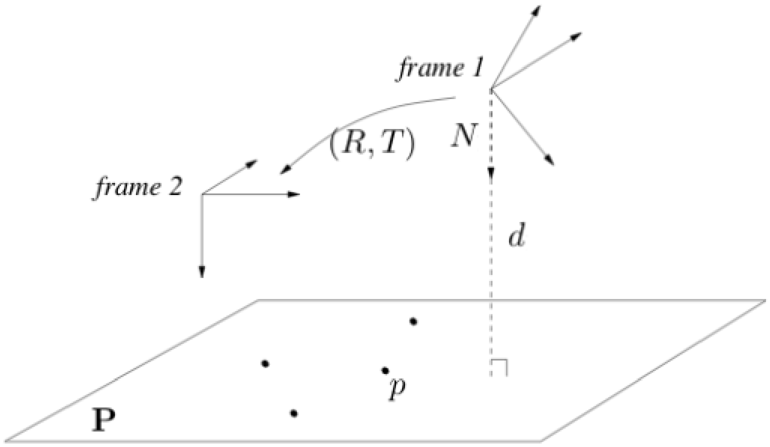
Essential Matrix

- Relate 3d world coordinates and image coordinates with epipolar constraints
  - Given 3d relationship
    - $X_2 = RX_1 + T$
  - Image coordinate relationship
    - $X_1 = \lambda_1 x_1$ and $X_2 = \lambda_2 x_2$
- Map line to a point between images
  - $x_2^T E x_1 = 0$
    - E – 3×3 Essential matrix compactly encodes (R, T)
- Can find the Essential matrix using the 8-point algorithm
  - This gives the camera rotation and translation

Homography

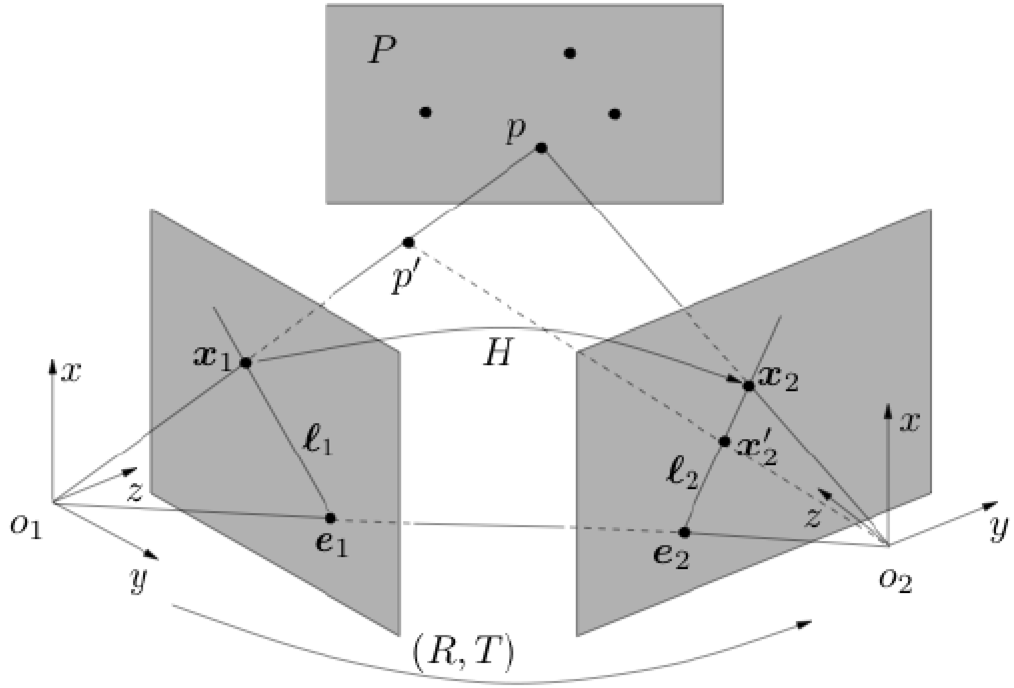
- 8-point algorithm will fail with coplanar points

- Planar relationship
  - $N^T X_1 = n_1 X + n_2 Y + n_3 Z = d$
  - $\frac{1}{d} N^T X_1 = 1$
- 3d relationship
  - $X_2 = R X_1 + T$
  - $X_2 = R X_1 + T \frac{1}{d} N^T X_1$
  - $X_2 = H X_1$
- Homography matrix
  - $H = R + T \frac{1}{d} N^T$
  - 3×3 linear transformation from 3d points



- 2D homography coordinates
  - $X_1 = \lambda_1 x_1$ and $X_2 = \lambda_2 x_2$
  - $\lambda_2 x_2 = H \lambda_1 x_1$
- Homography is a direct mapping between points in the image planes
  - Equality up to a scale factor (universal scale ambiguity)

Induced Homography



- Plane P induces the homography
  - $x_2 \sim H x_1$
- A point *p'* not actually on the plane *P* will get mapped in image 2 as if it were pushed onto P along the ray $\overline{o_1 p'}$

Computing Homography

- Process known as the 4-point algorithm or the Direct Linear Transform (DLT)
- Start from homography constraint
  - $x_2 \sim H x_1$

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\begin{aligned} x_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y_i'(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{aligned} x_i' &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y_i' &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Adapted from R. Szeliski

Solving for Homographies

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

A	**h**	**0**
2n × 9	**9**	**2n**

Defines a least squares problem: $\text{minimize } \|\mathbf{A}\mathbf{h} - \mathbf{0}\|^2$

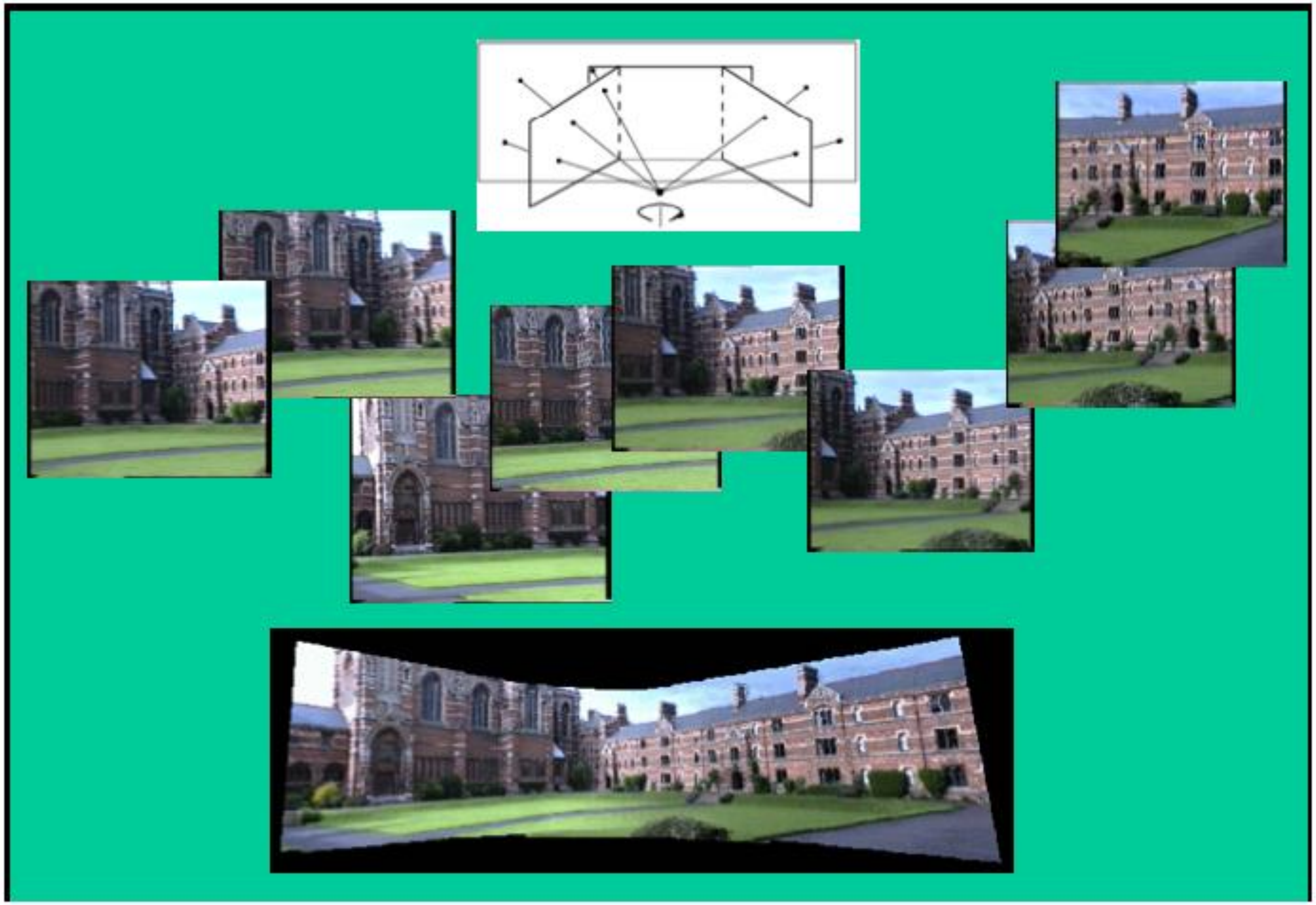
- Since **h** is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

Adapted from R. Szeliski

Purely Rotating Camera

- No translation between cameras
  - $X_2 = RX_1$
  - $H = R$
- Camera rotating about optical center captures images of a 3d scene as if the scene were on a plane infinitely far away from the camera
- Planar panoramas can be constructed from rotation
  - Select image to be a reference
  - Find corresponding points between overlapping images → derive pairwise homography
  - Blend images together

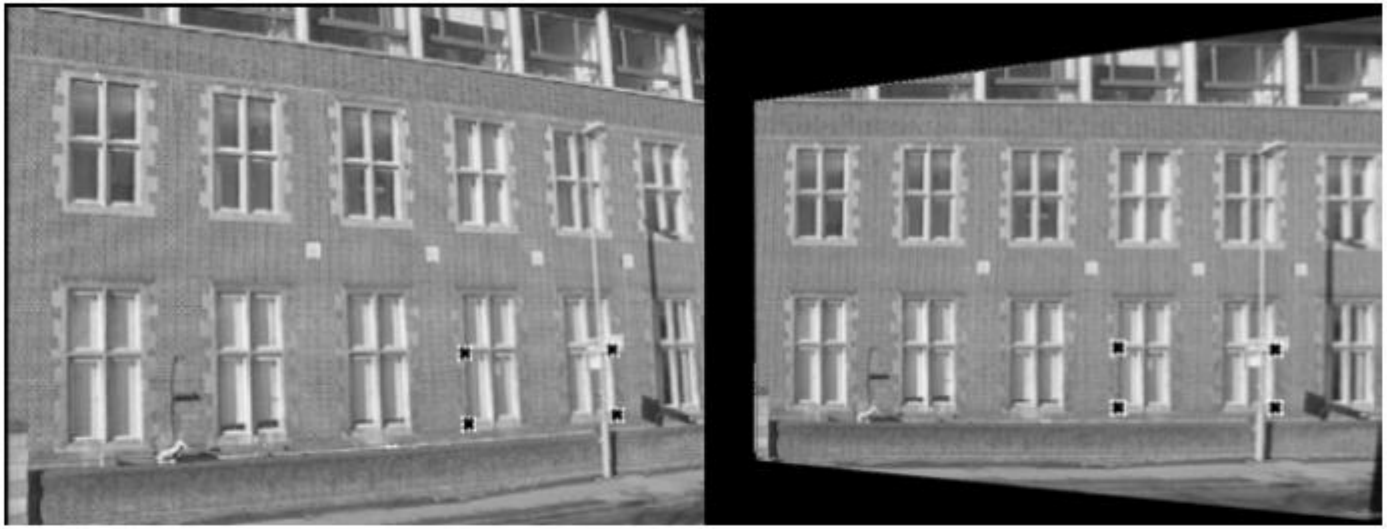
Planar Panorama



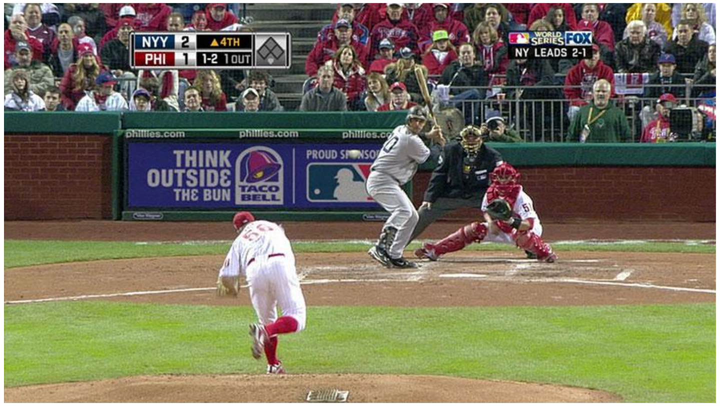
- Notice the bow-tie effect as images further away from the reference are warped outward to fit the homography

Planar Image Rectification

- Given a single perspective image, unwarp it so that a plane has parallel lines
  - Select 4 corners of rectangle, define the output coordinates, compute homography, and warp image



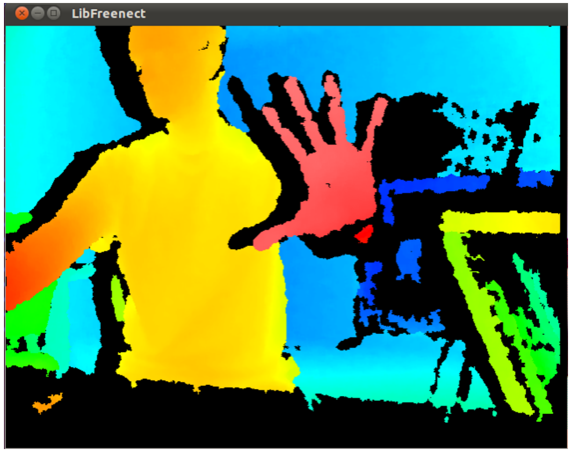
More Fun with Homographies



- http://youtu.be/UirmvNktkBc

Stereo Matching

- Given two more images of the same scene or object, compute a representation of its shape
- Common application is generating disparity or depth map
  - Popularized for games recently by Kinect
- What are applications?



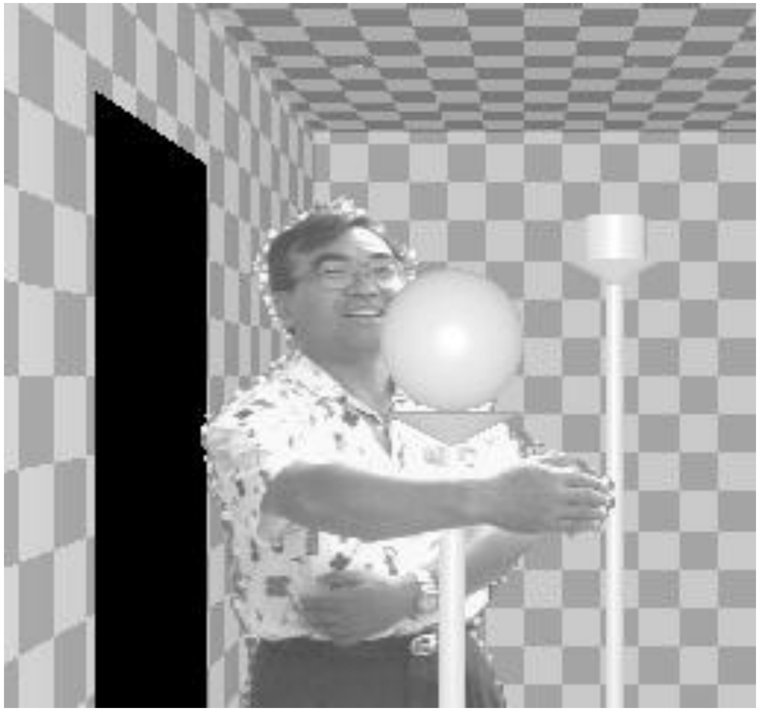
Face modeling

- From one stereo pair to a 3D head model

[Frederic Deverney, INRIA]

Z-keying: mix live and synthetic

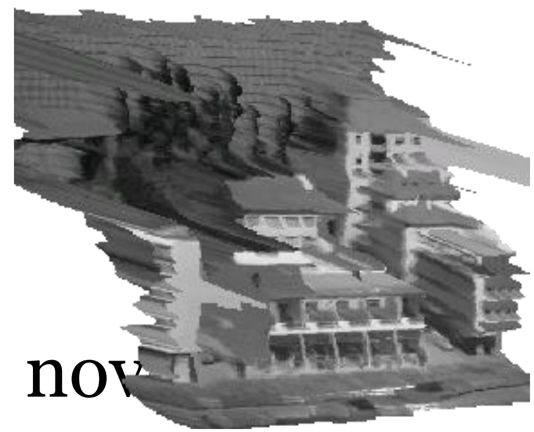
- Takeo Kanade, CMU (Stereo Machine)



View Interpolation

- Given two images with correspondences, *morph* (warp and cross-dissolve) between them [Chen & Williams, SIGGRAPH'93]

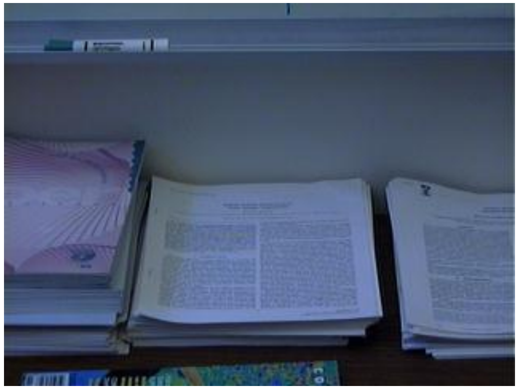
nov



[Matthies,Szeliski,Kanade'88]

More view interpolation

- Spline-based depth map



input

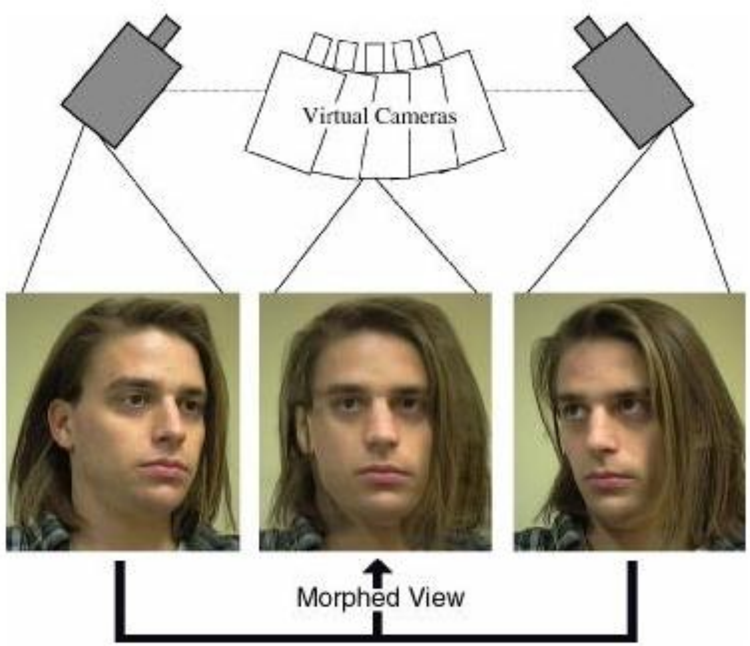
depth image

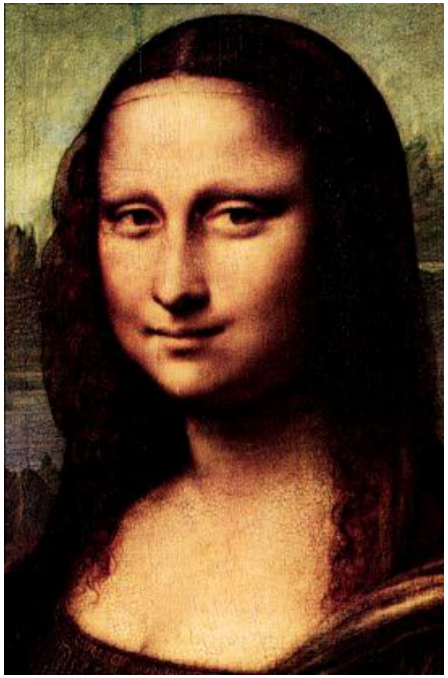
novel view

- [Szeliski & Kang '95]

View Morphing

- Morph between pair of images using epipolar geometry [Seitz & Dyer, SIGGRAPH'96]





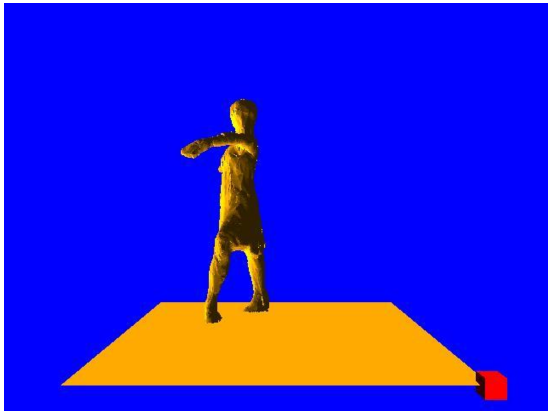
Video view interpolation

[Zitnick et. al 2004]

- http://research.microsoft.com/en-us/um/people/larryz/MassiveArabesque.wmv
- http://www.youtube.com/watch?v=uqKTbyNoaxE

Virtualized Reality™

- [Takeo Kanade *et al.*, CMU]
  - collect video from 50+ stream
  - reconstruct 3D model sequences



  - steerable version used for SuperBowl XXV "eye vision"

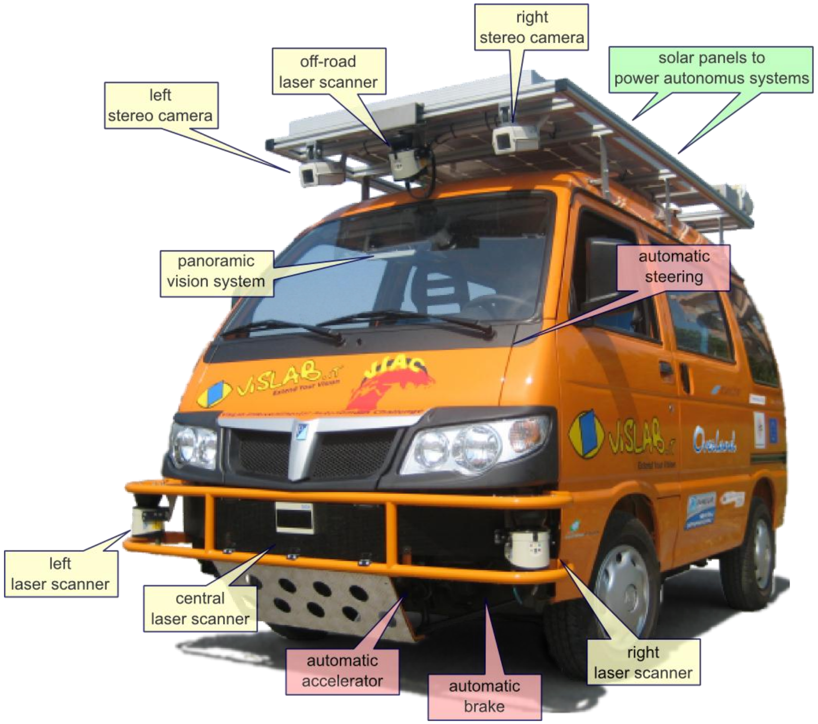
Real-time stereo

Nomad robot searches for meteorites in Antartica
http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

- Used for robot navigation (and other tasks)
  - Software-based real-time stereo techniques

Driver Assistance Systems

- Environment sensing and autonomous control
- http://youtu.be/_imrrzn8NDk?t=12s
- http://youtu.be/-MWVbfia3Dk



http://vislab.it/automotive/
Autonomous drive from Parma, Italy to Shanghai, China

Additional applications

- Real-time people tracking (systems from Pt. Gray Research and SRI)
- “Gaze” correction for video conferencing [Ott,Lewis,Cox InterChi’93]
- Other ideas?

Stereo Matching

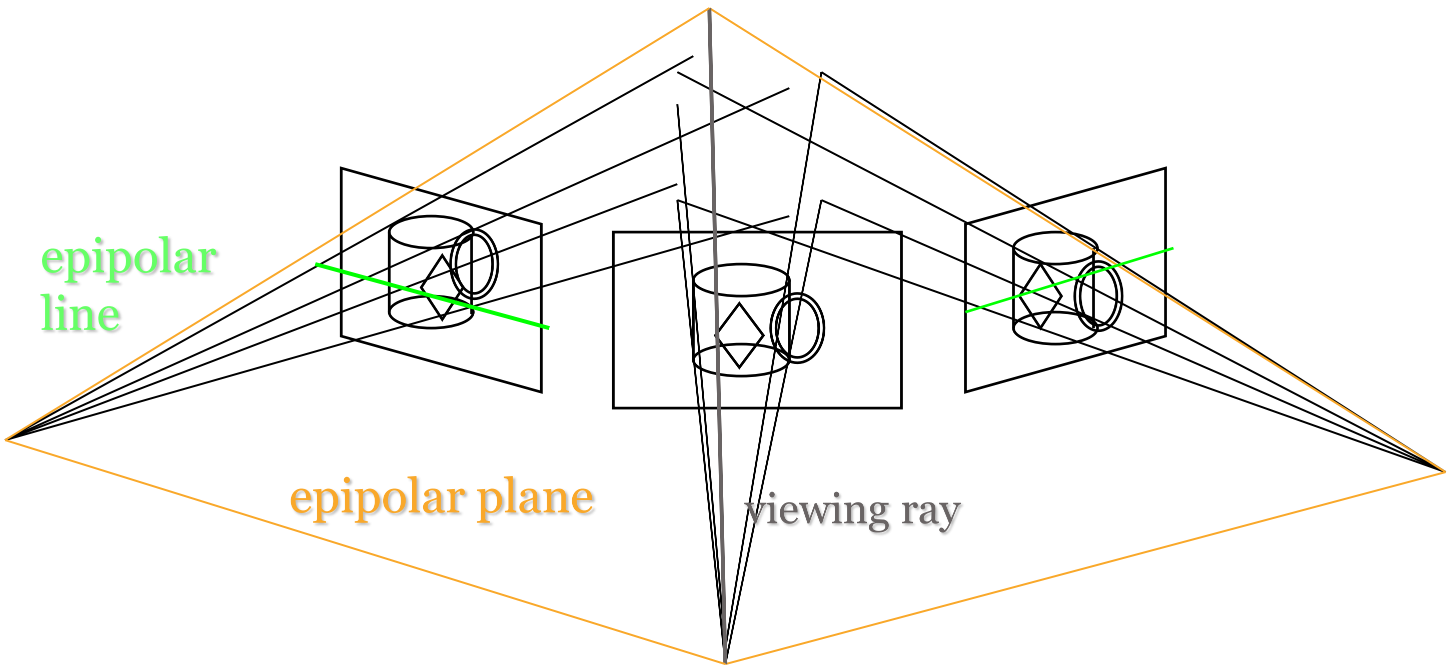
- Given two or more images of the same scene or object, compute a representation of its shape
- What are some possible representations?
  - depth maps
  - volumetric models
  - 3D surface models
  - planar (or offset) layers

Stereo Matching

- What are some possible algorithms?
  - match “features” and interpolate
  - match edges and interpolate
  - match all pixels with windows (coarse-fine)
  - use optimization:
    - iterative updating
    - dynamic programming
    - energy minimization (regularization, stochastic)
    - graph algorithms

Stereo: epipolar geometry

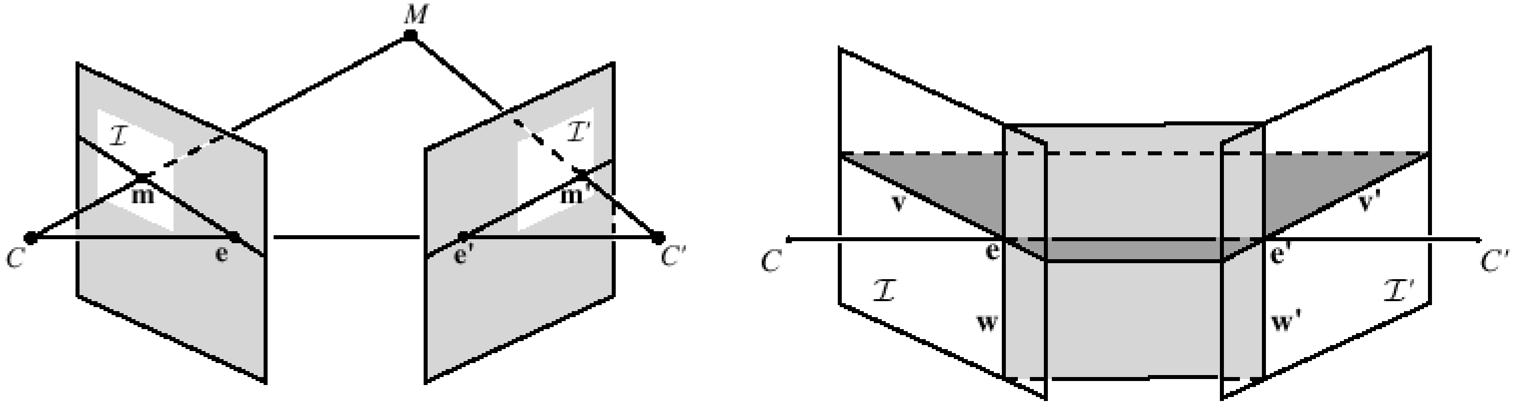
- Match features along epipolar lines



- **Rectification:** warping the input images (perspective transformation) so that epipolar lines are horizontal

Rectification

- Project each image onto same plane, which is parallel to the epipole
- Resample lines (and shear/stretch) to place lines in correspondence, and minimize distortion
-



- [Loop and Zhang, CVPR'99]

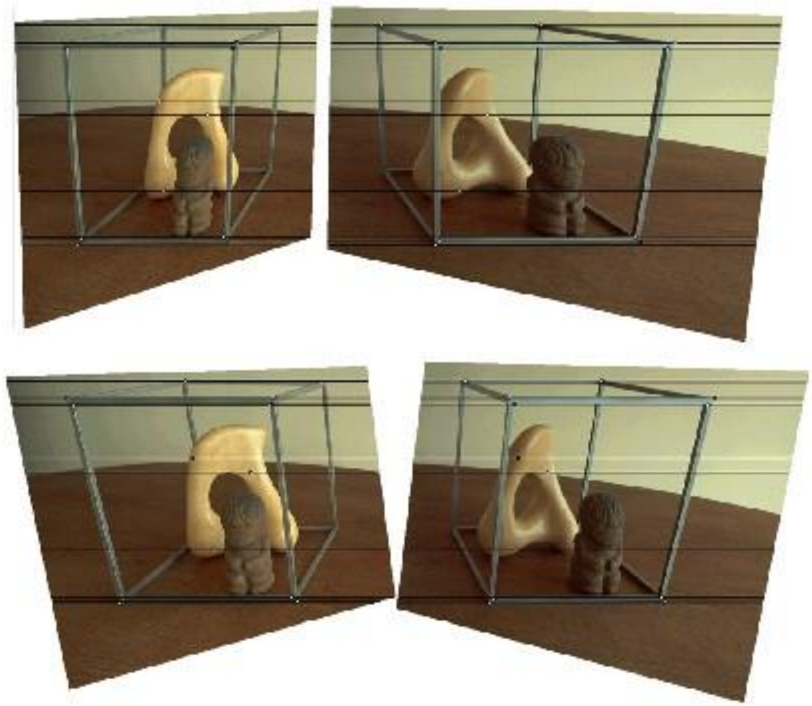
Rectification

(a) Original image pair overlayed with several epipolar lines.

(b) Image pair transformed by the specialized projective mapping $\mathbf{H}_p$ and $\mathbf{H}'_p$. Note that the epipolar lines are now parallel to each other in each image.

BAD!

Rectification



(c) Image pair transformed by the similarity $\mathbf{H}_r$ and $\mathbf{H}'_r$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).

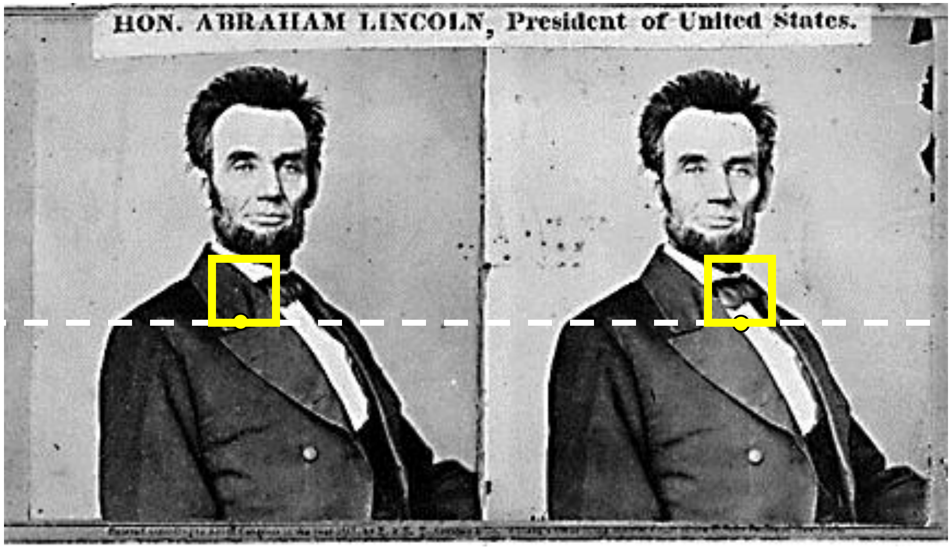
(d) Final image rectification after shearing transform $\mathbf{H}_s$ and $\mathbf{H}'_s$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

GOOD!

Finding correspondences

- apply feature matching criterion (e.g., correlation or Lucas-Kanade) at *all* pixels simultaneously
- search only over epipolar lines (many fewer candidate positions)

Your basic stereo algorithm



For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

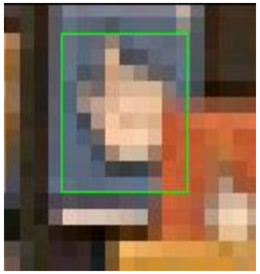
- This should look familar...

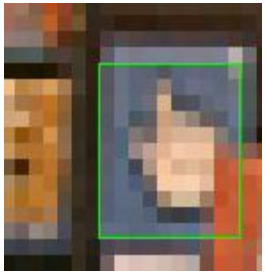
Image registration (revisited)

- How do we determine correspondences?
  - *block matching* or *SSD* (sum squared differences)

$$E(x,y;d) = \sum_{(x',y')\in N(x,y)} [I_L(x'+d,y') - I_R(x',y')]^2$$

d is the *disparity* (horizontal motion)

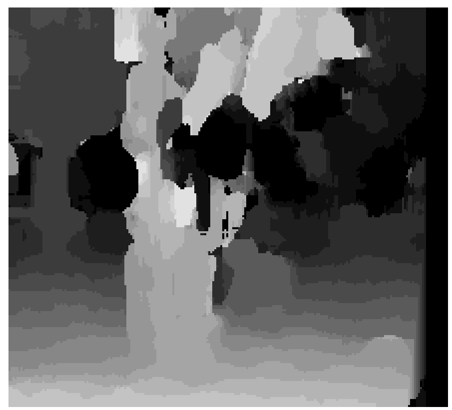


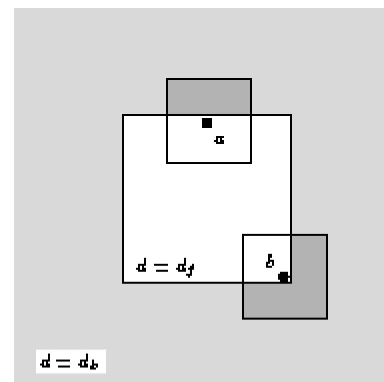


- How big should the neighborhood be?

Neighborhood size

- Smaller neighborhood: more details
- Larger neighborhood: fewer isolated mistakes





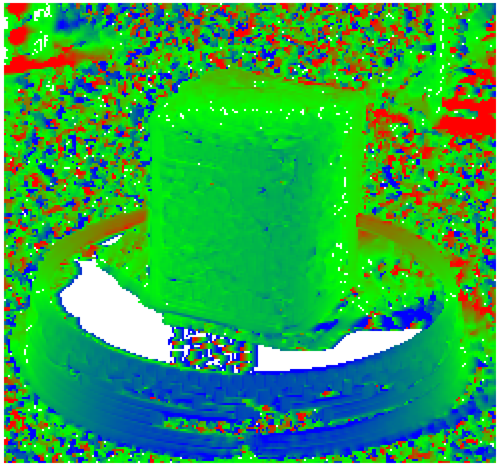
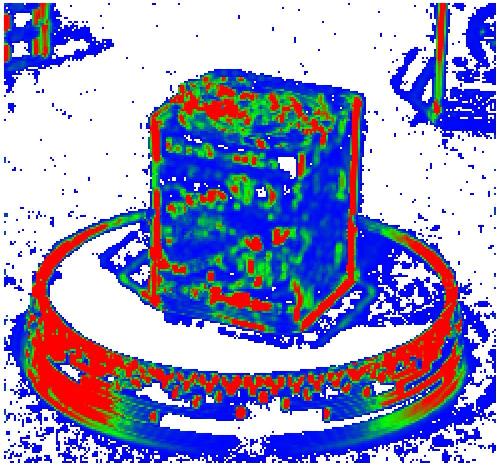
- w = 3 w = 20

Matching criteria

- Raw pixel values (correlation)
- Band-pass filtered images [Jones & Malik 92]
- “Corner” like features [Zhang, …]
- Edges [many people…]
- Gradients [Seitz 89; Scharstein 94]
- Rank statistics [Zabih & Woodfill 94]

Stereo: certainty modeling

- Compute certainty map from correlations

- input depth map certainty map

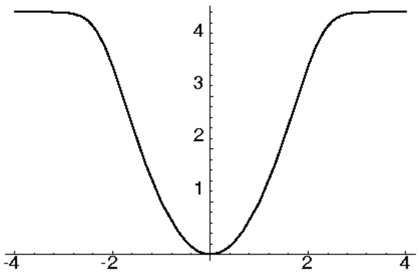
Stereo matching framework

1. For every disparity, compute *raw* matching costs

$$E_0(x, y; d) = \rho(I_L(x' + d, y') - I_R(x', y'))$$

Why use a robust function?

- occlusions, other outliers

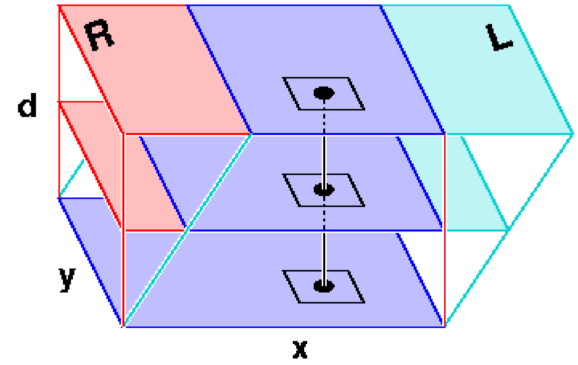


- Can also use alternative match criteria

Stereo matching framework

2. Aggregate costs spatially

$$E(x, y; d) = \sum_{(x',y') \in N(x,y)} E_0(x', y', d)$$



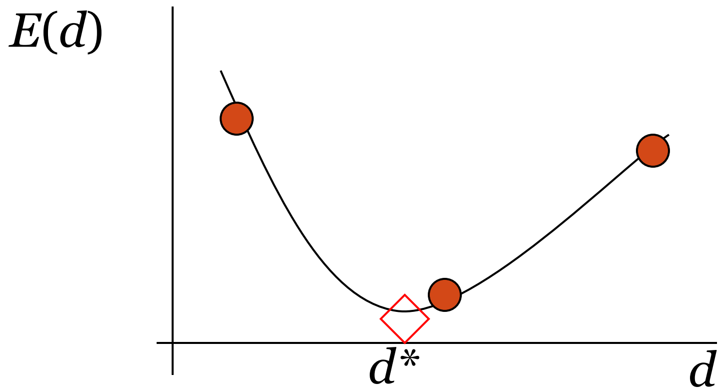
- Here, we are using a *box filter* (efficient moving average implementation)
- Can also use weighted average, [non-linear] diffusion…

Stereo matching framework

3. Choose winning disparity at each pixel

$$d(x,y) = \arg\min_{d} E(x,y;d)$$

4. Interpolate to *sub-pixel* accuracy

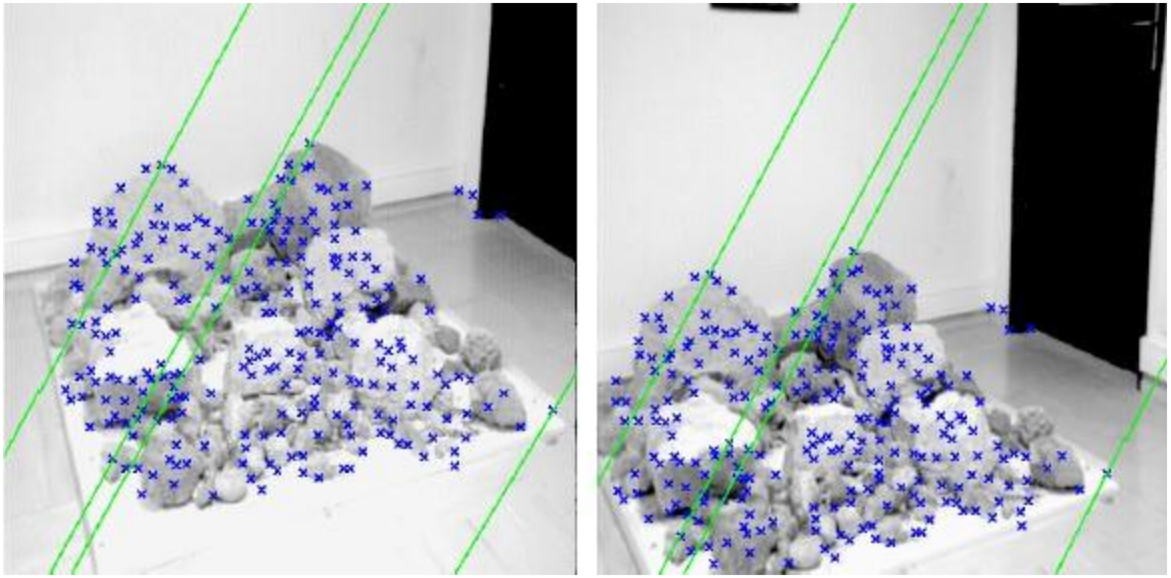


Traditional Stereo Matching

- Advantages:
  - gives detailed surface estimates
  - fast algorithms based on moving averages
  - sub-pixel disparity estimates and confidence
- Limitations:
  - narrow baseline $\Rightarrow$ noisy estimates
  - fails in textureless areas
  - gets confused near occlusion boundaries

Feature-based stereo

- Match “corner” (interest) points



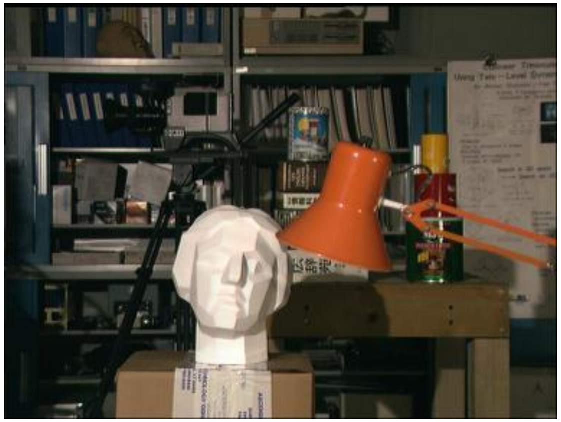
- Interpolate complete solution

Data interpolation

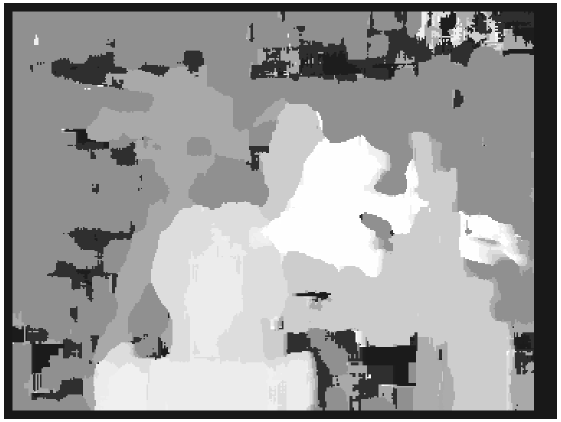
- Given a sparse set of 3D points, how do we *interpolate* to a full 3D surface?
- Scattered data interpolation [Nielson93]
- triangulate
- put onto a grid and fill (use pyramid?)
- place a *kernel function* over each data point
- minimize an energy function

- Lots of more advanced stereo matching options and algorithms exist

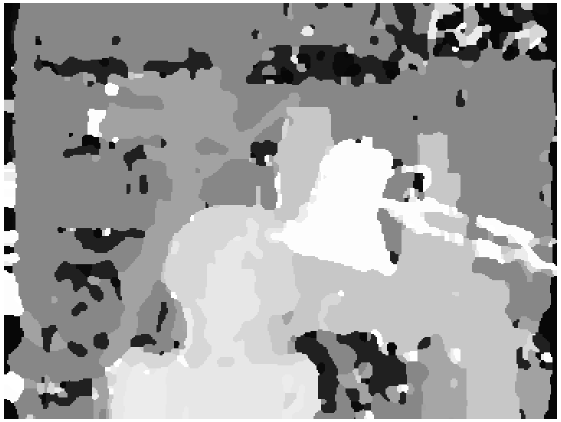
Depth Map Results



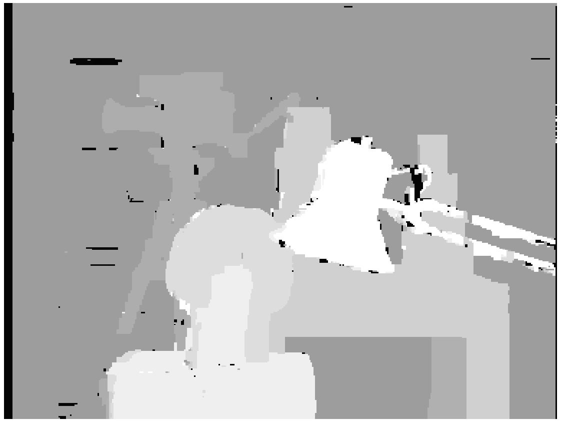
- Input image



Sum Abs Diff



- Mean field



Graph cuts