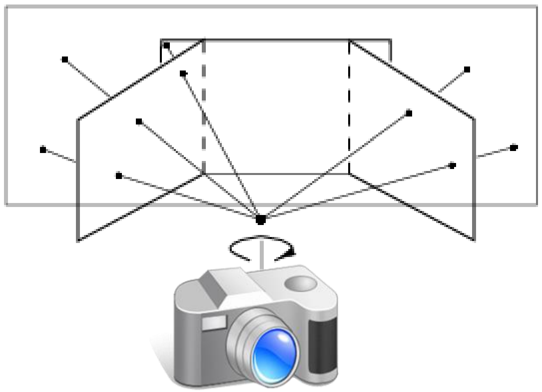
Lecture 6

黃正民
Cheng-Ming Huang

EE, NTUT

Image Stitching

- Obtain a mosaic image with wider angle view by combining multiple images.



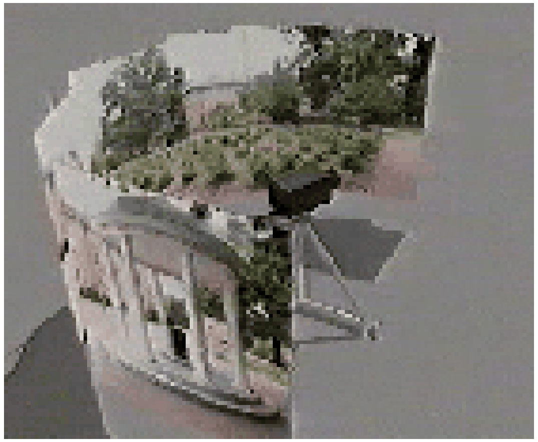


image from S. Seitz

Panoramas Mosaic : 360° field of view

Image Stitching

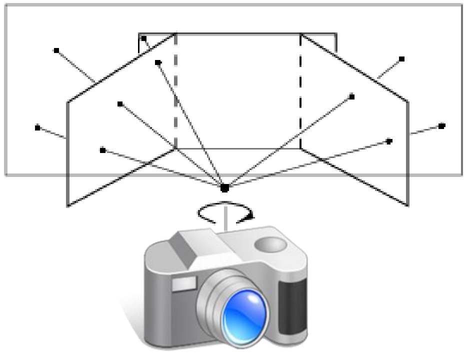
- Stitching = alignment + blending

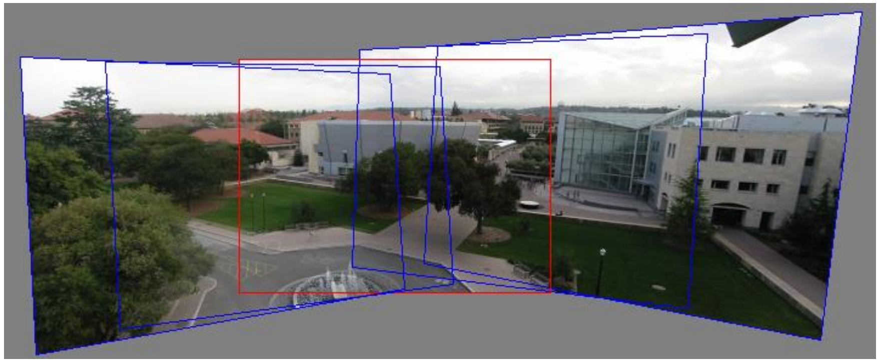
geometrical registration

photometric registration

Stitch Together (a Panorama)

- Basic Procedure
  - Take a sequence of images from the same position
    - Rotate the camera about its optical center
  - Compute transformation between second image and first (with cylindrical projection)
  - Transform the second image to overlap with the first
  - Blend the two together to create a mosaic
  - (If there are more images, repeat)





cylindrical projection

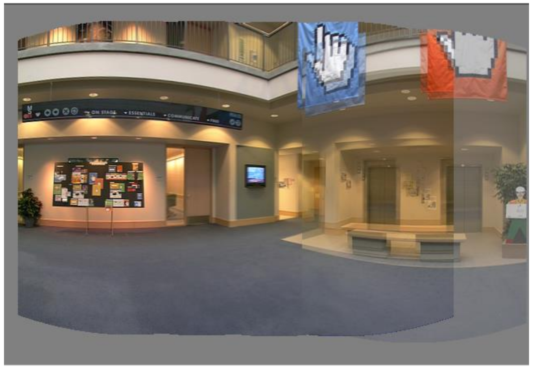


Image Warping with Homographies

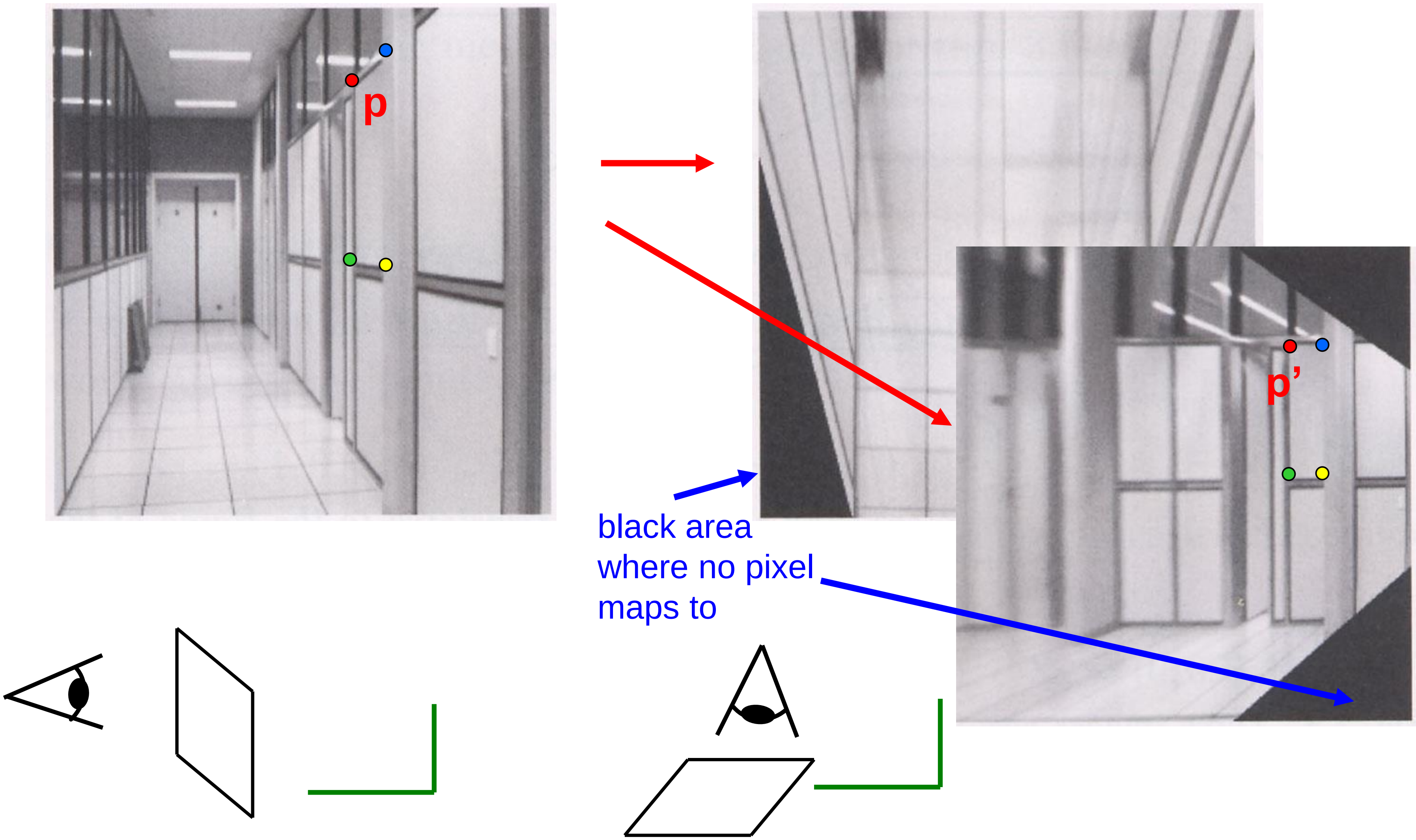
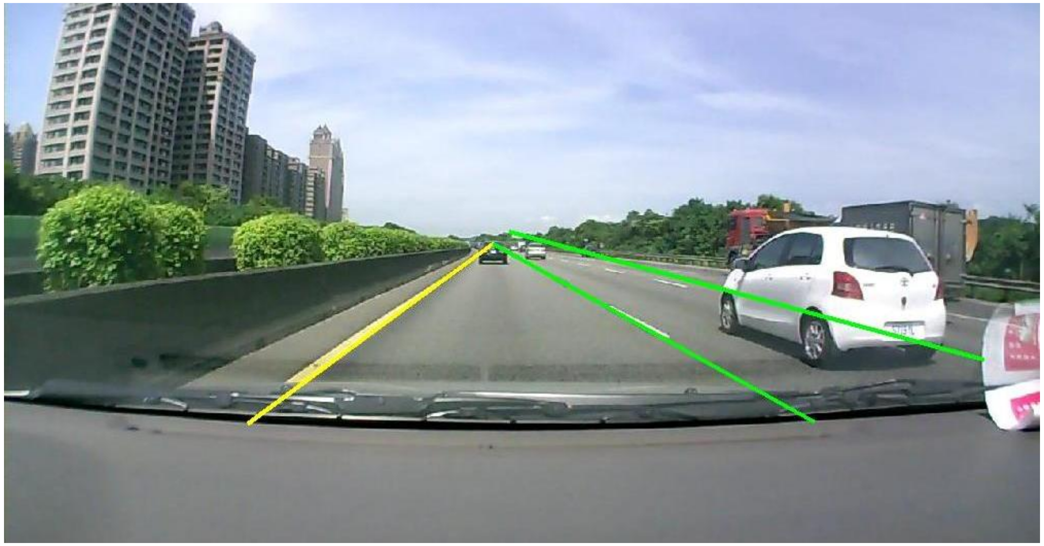


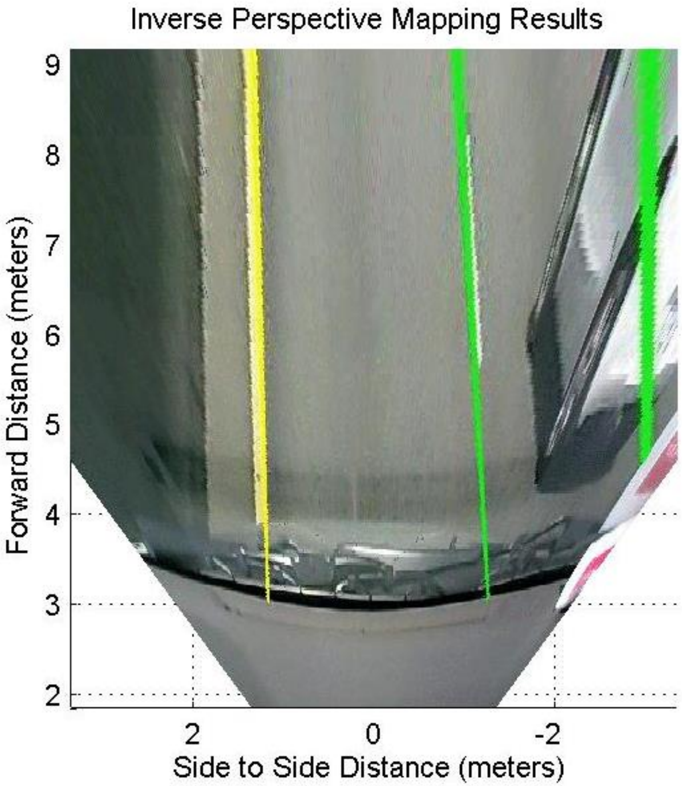
image plane in front

image plane below

Image Warping with Homographies

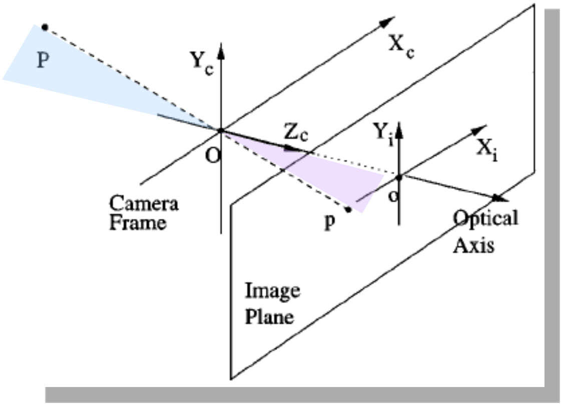
- Inverse perspective mapping (IPM)
  - Lane/vehicle detection and tracking

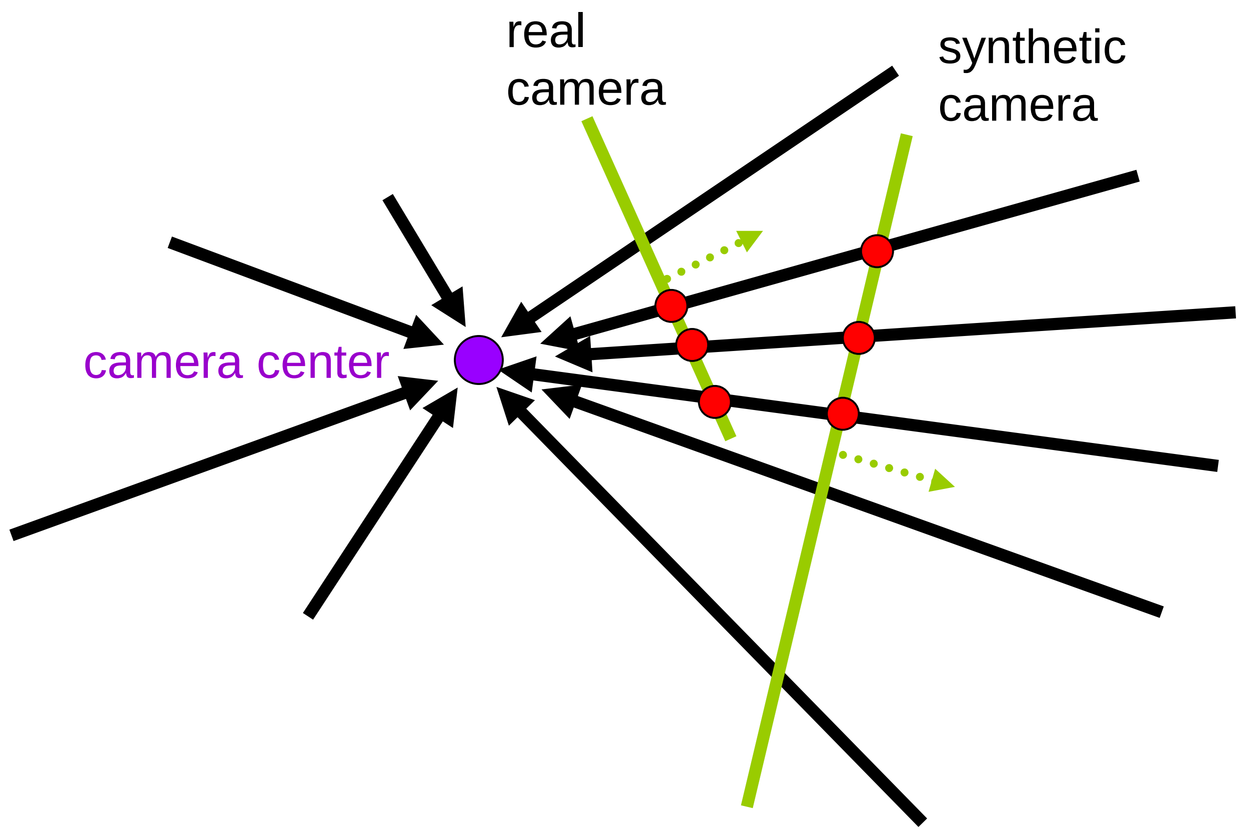




Stitch Together a Panorama

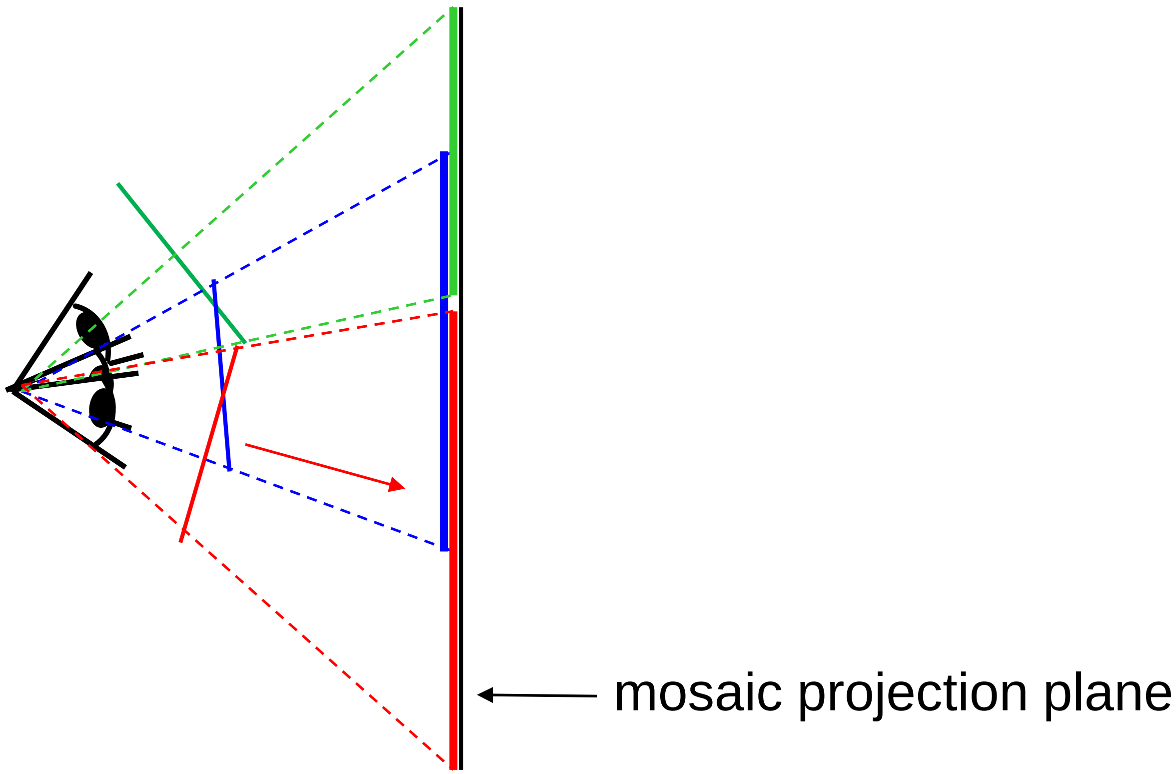
- A pencil of rays contains all views
  - Can generate any synthetic camera view as long as it has the same center of projection



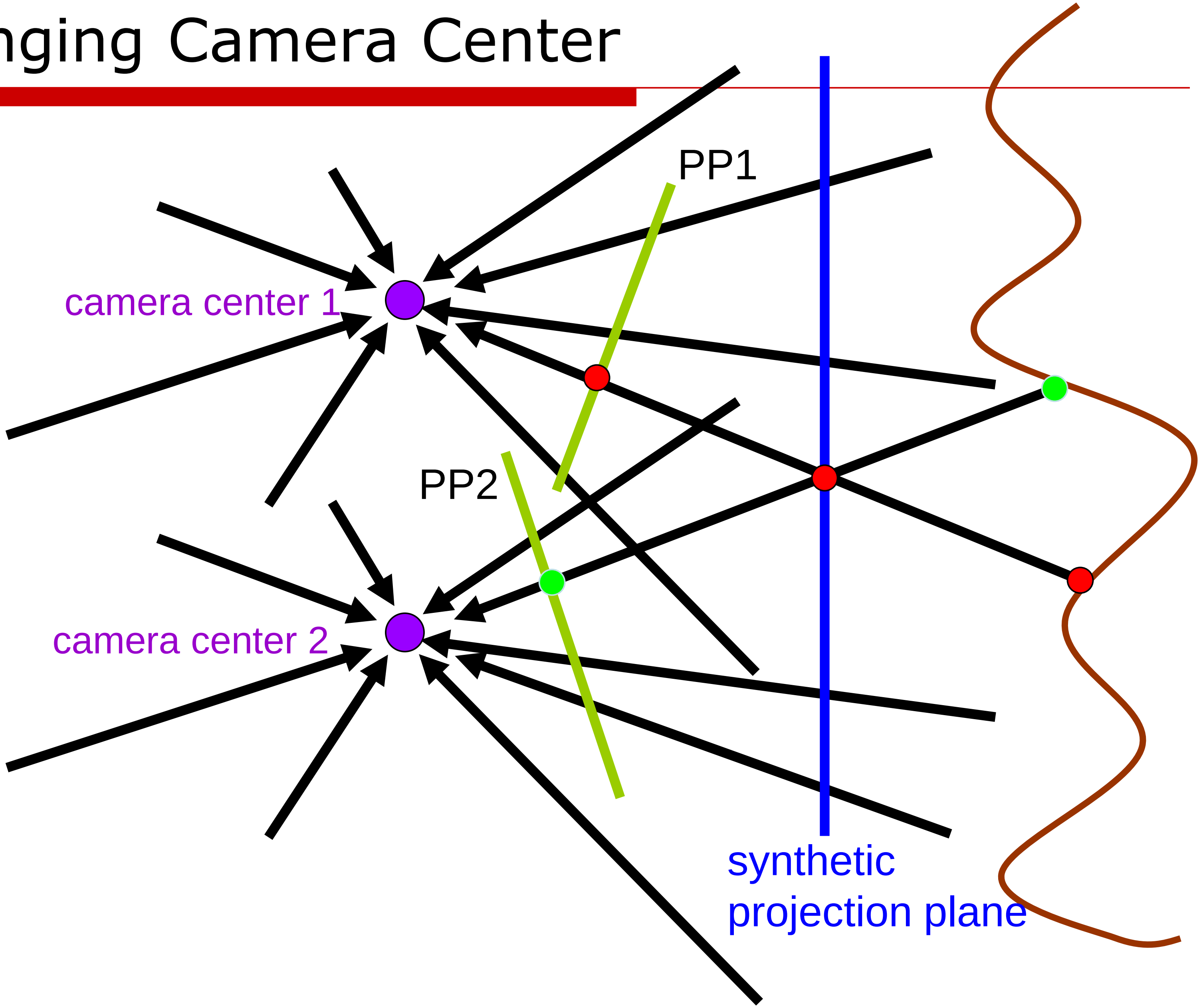


Stitch Together a Panorama

- 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



Planar Scene (or Far Away)

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

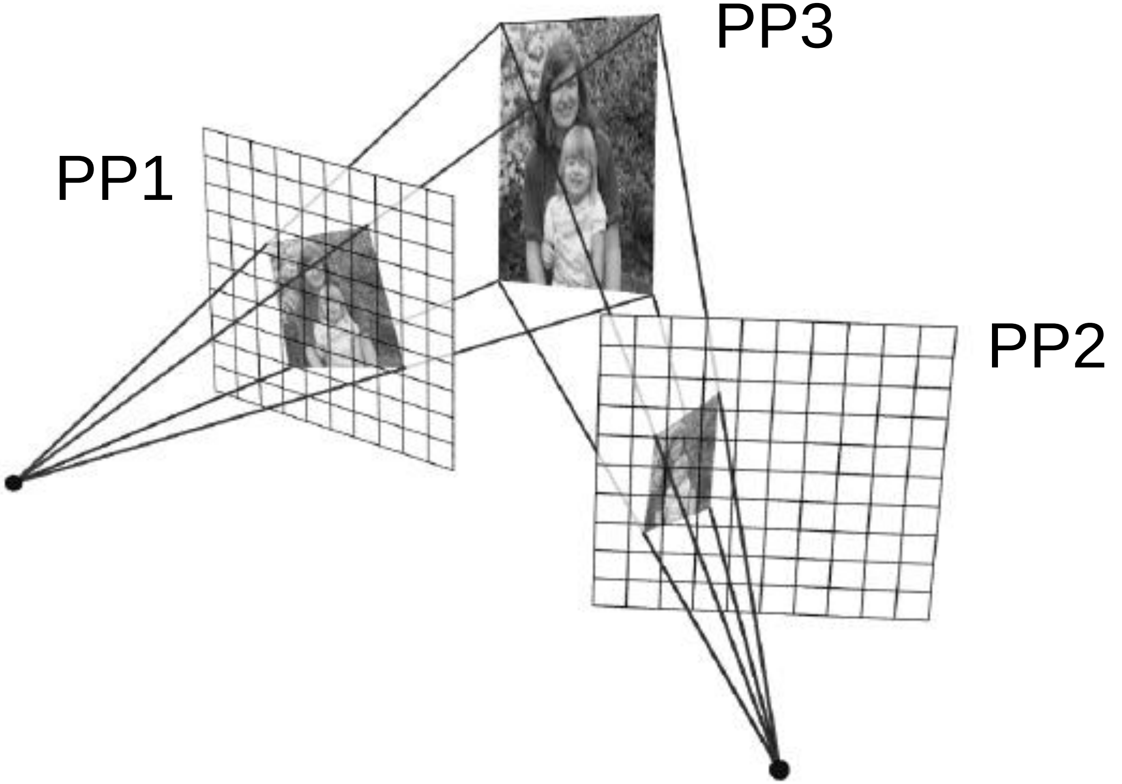


Image Alignment

- Two broad approaches:
  - Direct (pixel-based) alignment
    - Search for alignment where most pixels agree
  - Feature-based alignment
    - Search for alignment where *extracted features* agree
    - Can be verified using pixel-based alignment

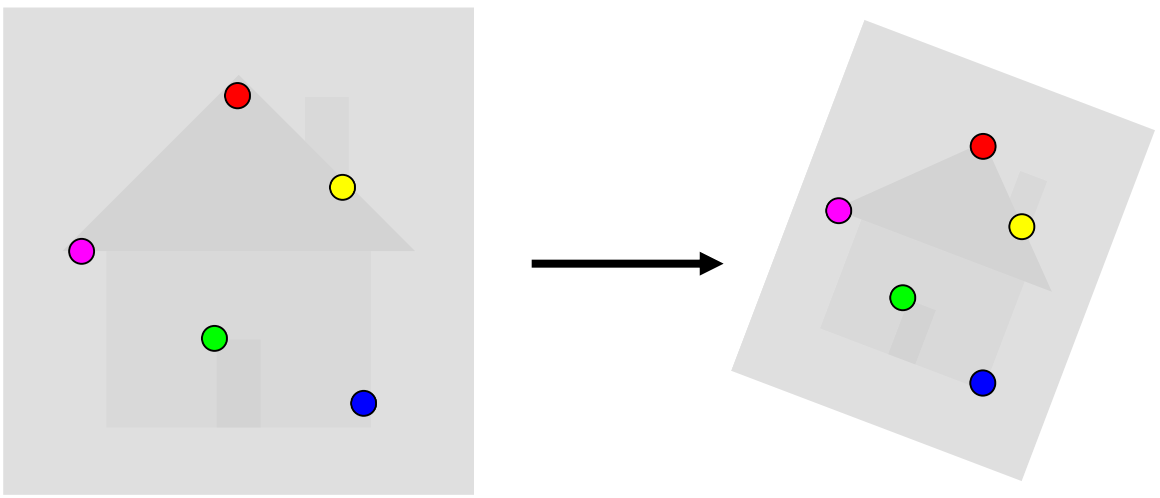
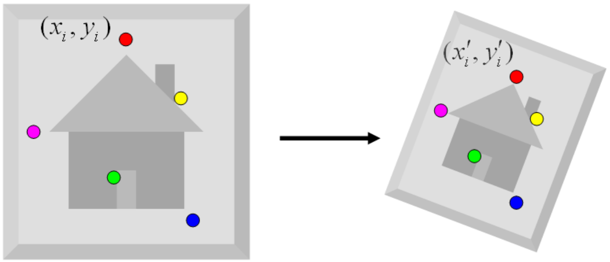


Image Alignment



- Fitting an affine transformation
  - Assuming we know the correspondences, how do we get the transformation?

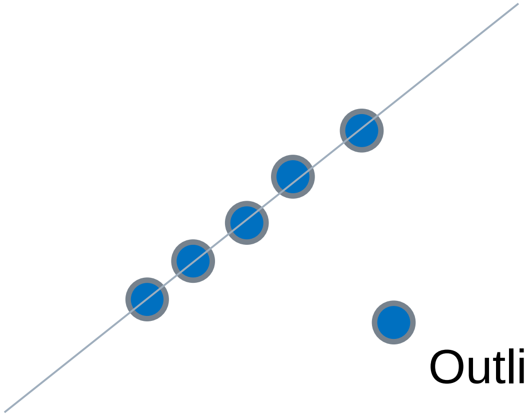
$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \Rightarrow \begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

A**x**=b

  - How many matches (correspondence pairs) do we need to solve for the transformation parameters?
    - (Translation+Rotation) 6 parameters ⇨ 3 points
  - Once we have solved for the parameters, we can compute the coordinates of the corresponding point for (x_{new}, y_{new})

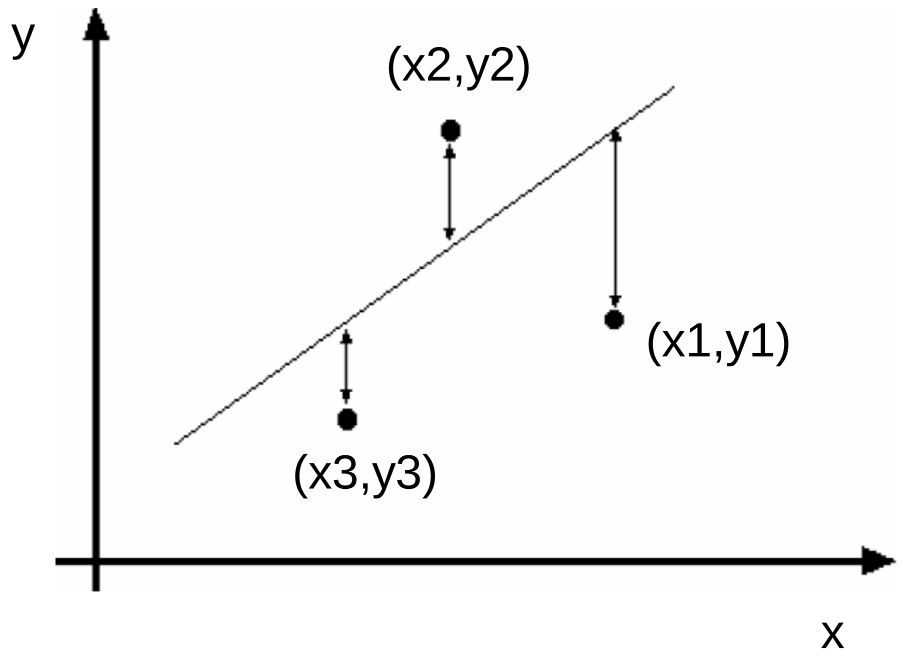
Outliers

- Outliers can hurt the quality of the parameter estimates, e.g.,
  - an erroneous pair of matching points from two images ⇨ wrong transformation matrix
  - an edge point that is noise, or doesn't belong to the line we are fitting.



Example: least squares line fitting

- Assuming all the points that belong to a particular line are known



$$f(a, b) = a + b\,x,$$

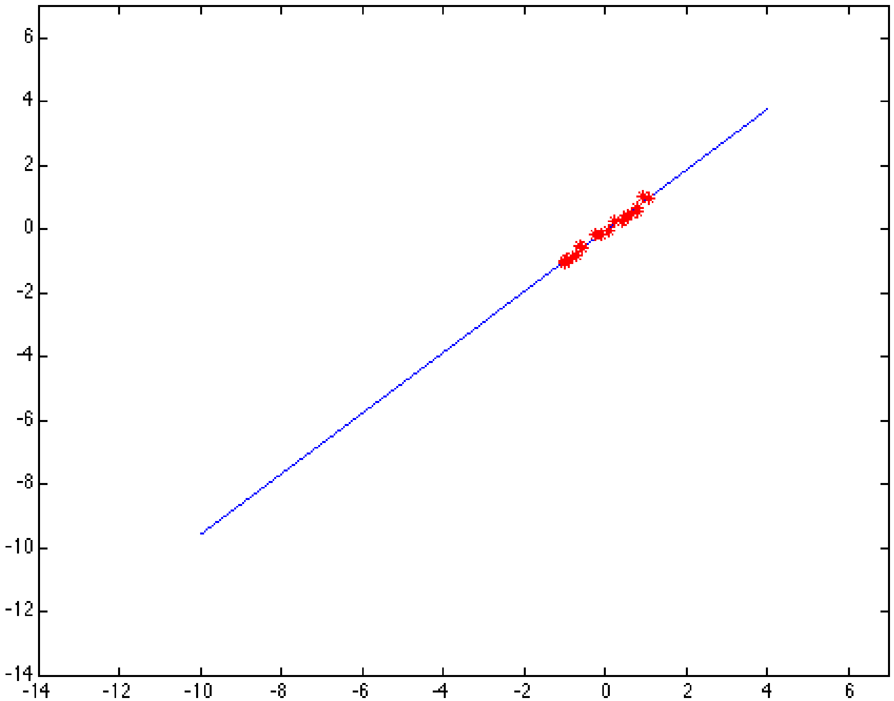
$$R^2(a, b) \equiv \sum_{i=1}^{n} [y_i - (a + b\,x_i)]^2$$

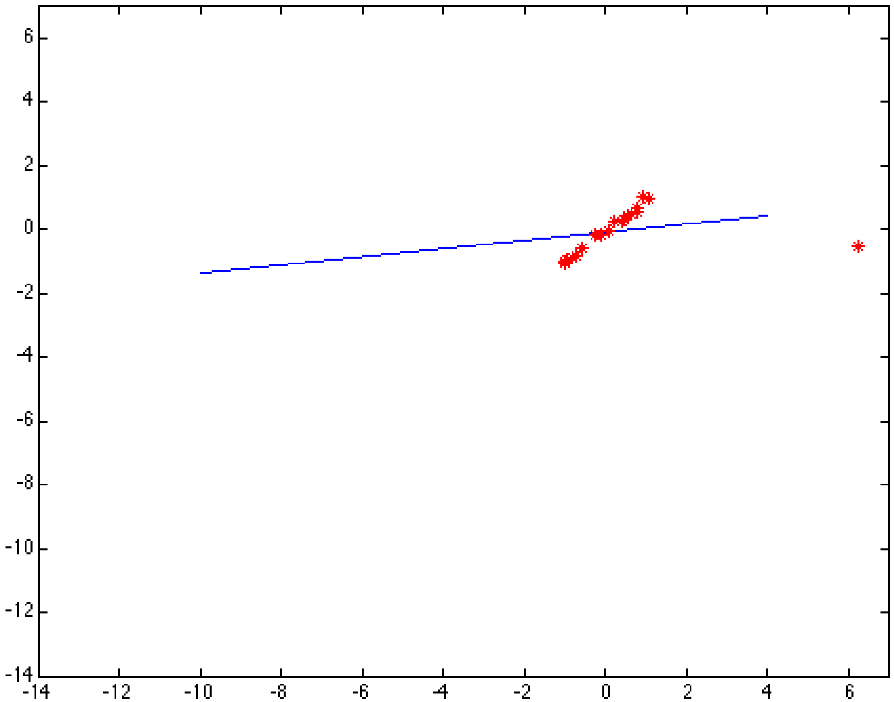
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix}$$

b = Ax

Outliers affect least squares fit

- 2D space
  - Given point data ⇨ obtain line parameters





RANSAC

- RANSAC : Random Sample Consensus
  - an algorithm for robust fitting of models in the presence of <u>many data outliers</u>
  - Given N data points x_i, assume that majority of them are generated from a model with parameters Θ, try to recover Θ.

- Algorithm

How many times?

Run k times, in each time:

How big?
Smaller is better

(1) draw n samples randomly

(2) fit parameters Θ with these n samples

(3) for each of other $(N - n)$ points,

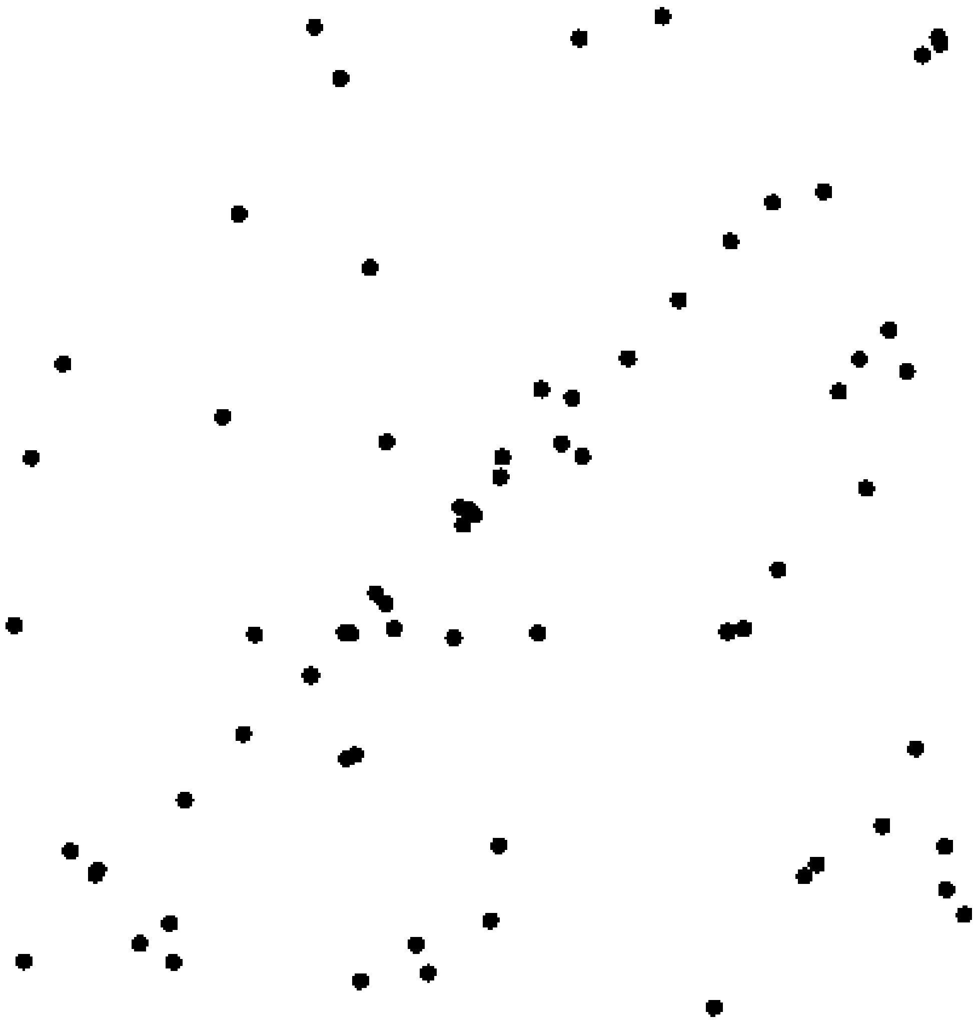
calculate its distance to the fitted model,

count the number of inlier points, c

How to define?
Depends on the problem.

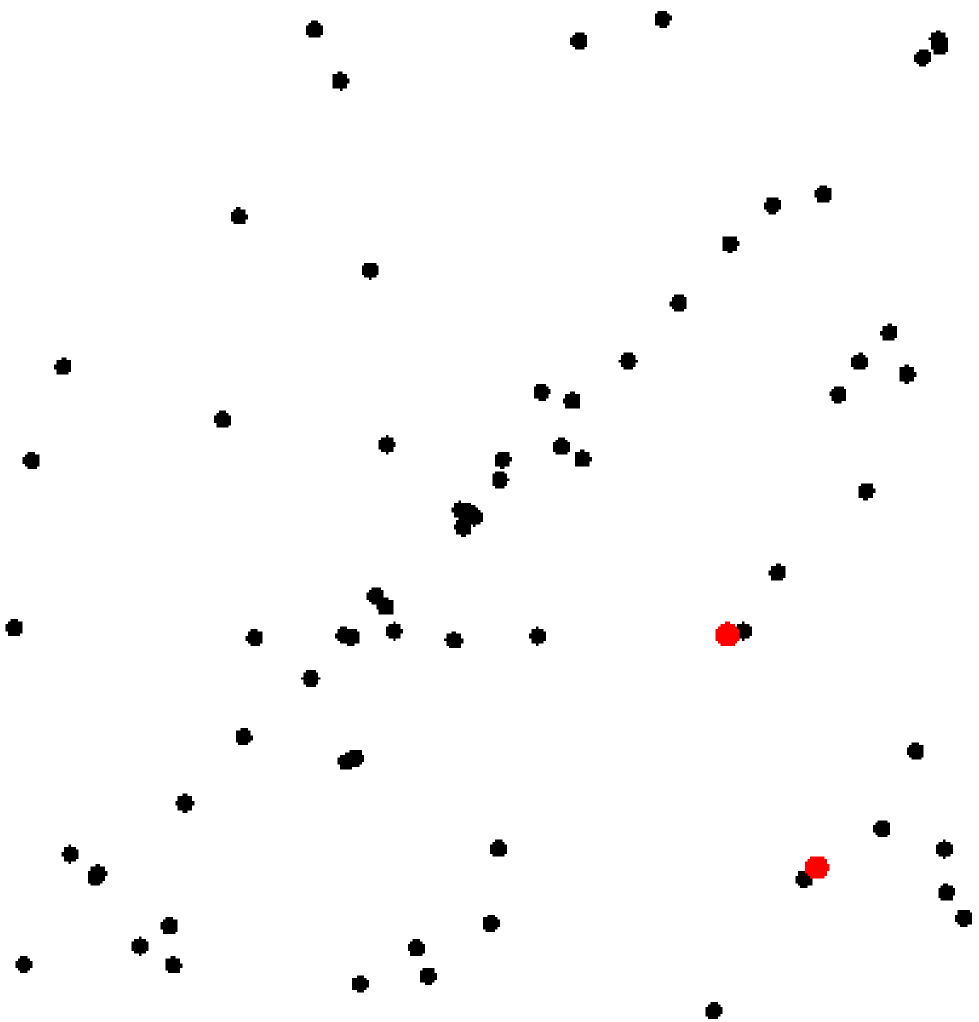
Output Θ with the largest c

RANSAC Example: Line Fitting



$b = Ax$

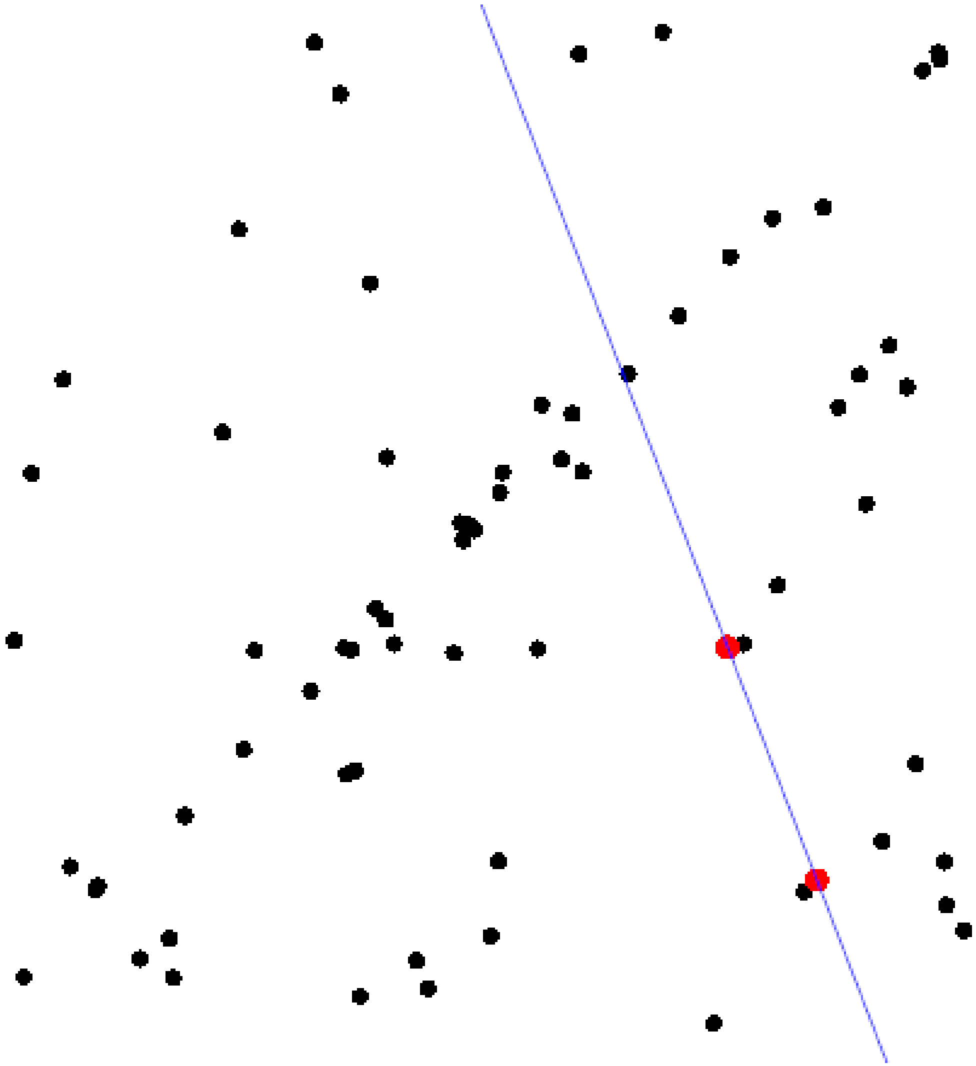
RANSAC Example: Line Fitting (1)



$n=2$

b = Ax

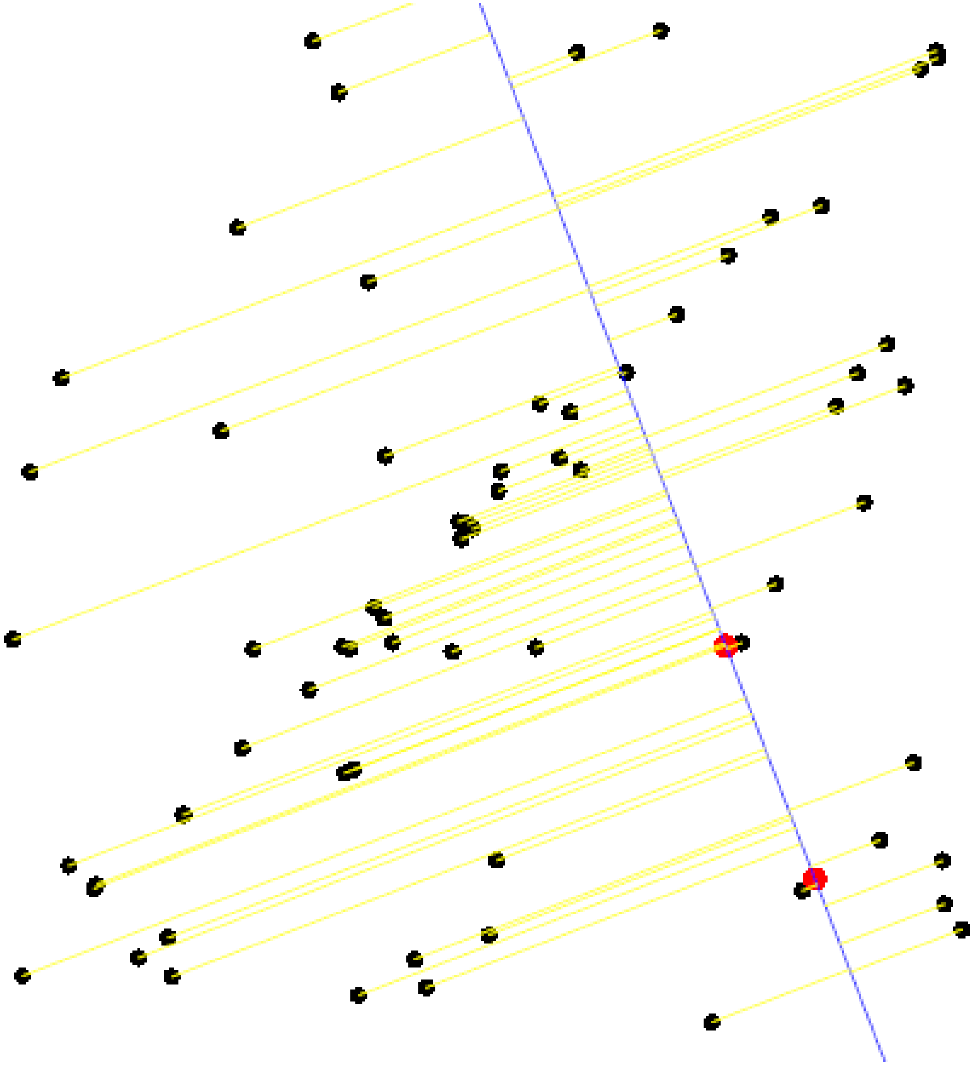
RANSAC Example: Line Fitting (2)



$b = Ax$

RANSAC Example: Line Fitting (3)

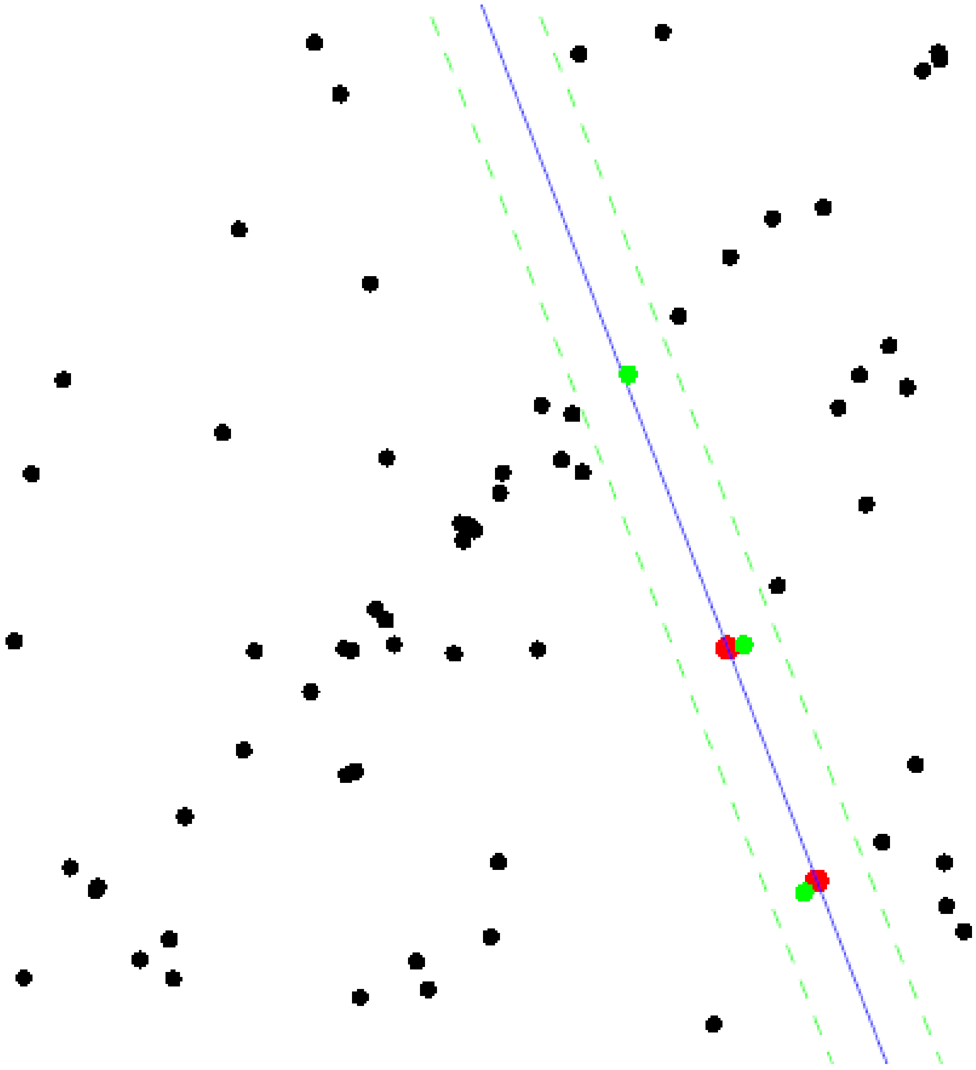
- Measure distances



$b = Ax$

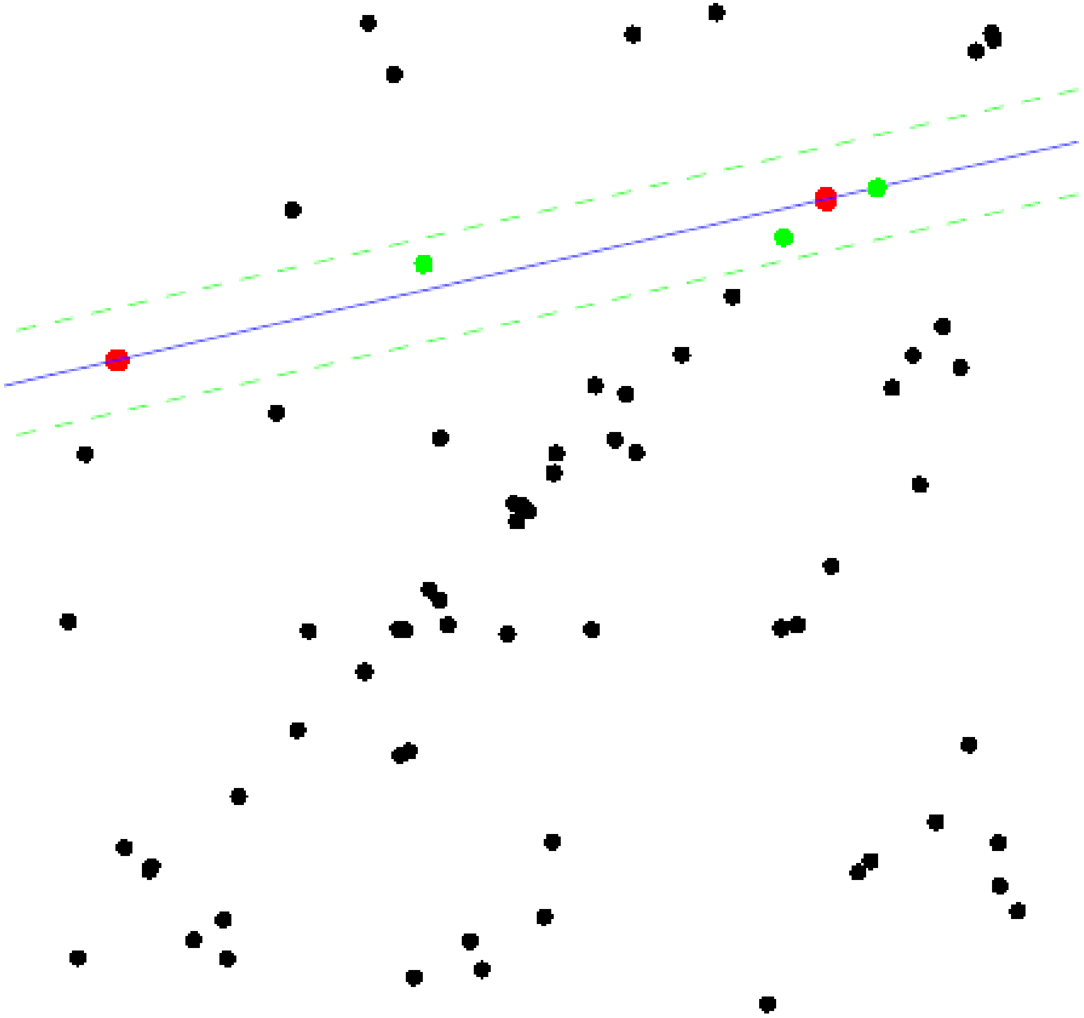
RANSAC Example: Line Fitting (4)

- Count inliers



RANSAC Example: Line Fitting (5)

- Another trial

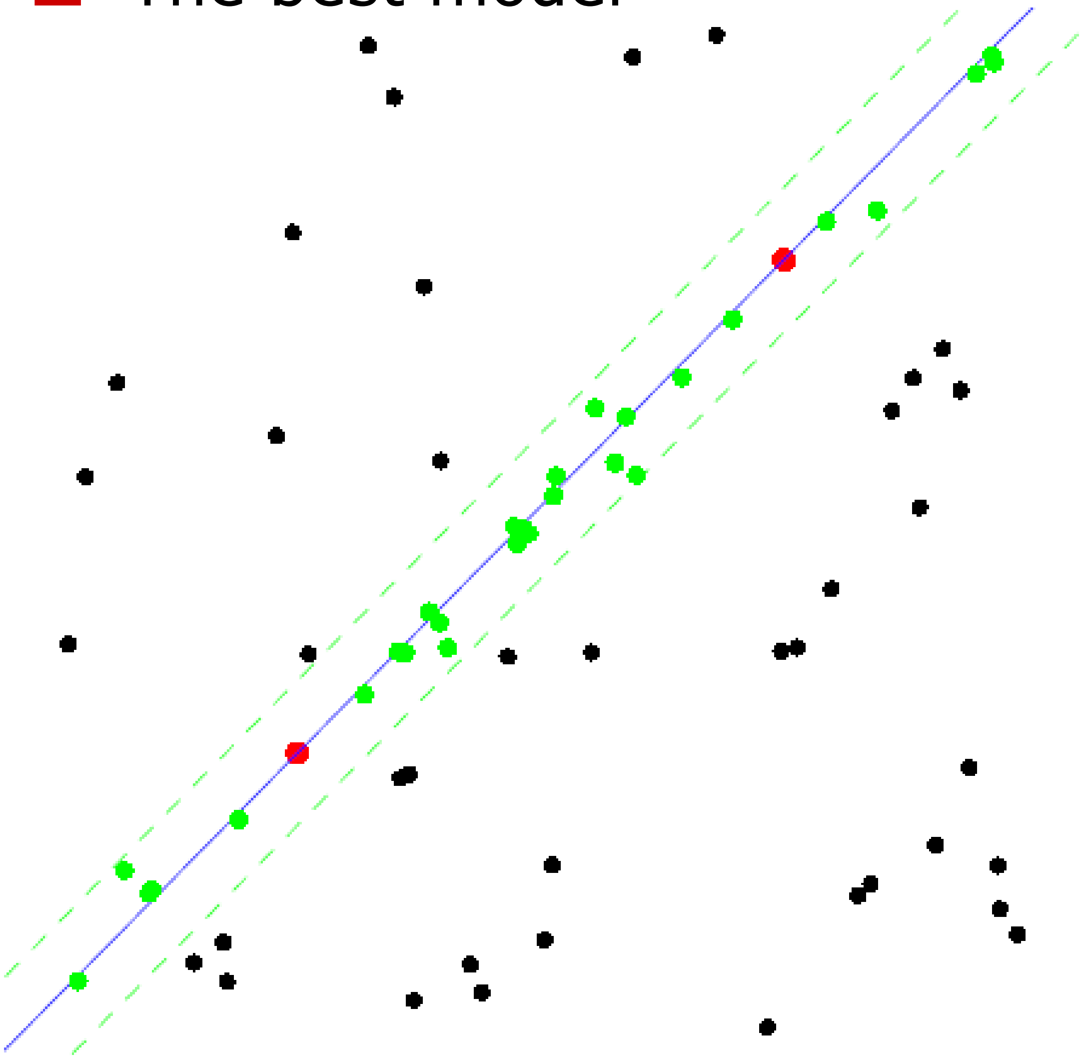


$c = 3$

b = Ax

RANSAC Example: Line Fitting (6)

- The best model



c=15

b = Ax

RANSAC

- How to determine k

  p: probability of real inliers

  P: probability of success after k trials

$$\underbrace{P = 1-\underbrace{(1-\underbrace{p^n}_{\text{n samples are all inliers}})}_{\text{a failure}}{}^k}_{\text{failure after k trials}}$$

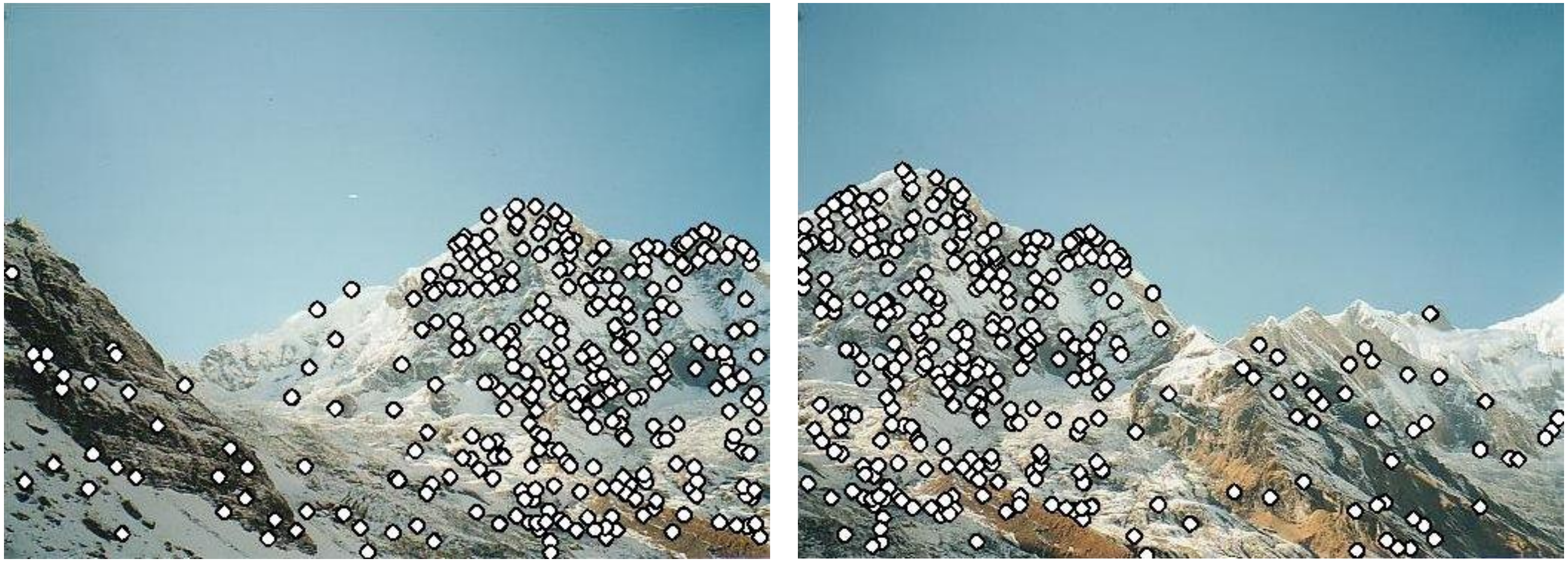
$$k = \frac{\log(1-P)}{\log(1-p^n)}$$

Define goal:
given n, p, P

for $P=0.99$

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

RANSAC for Homography

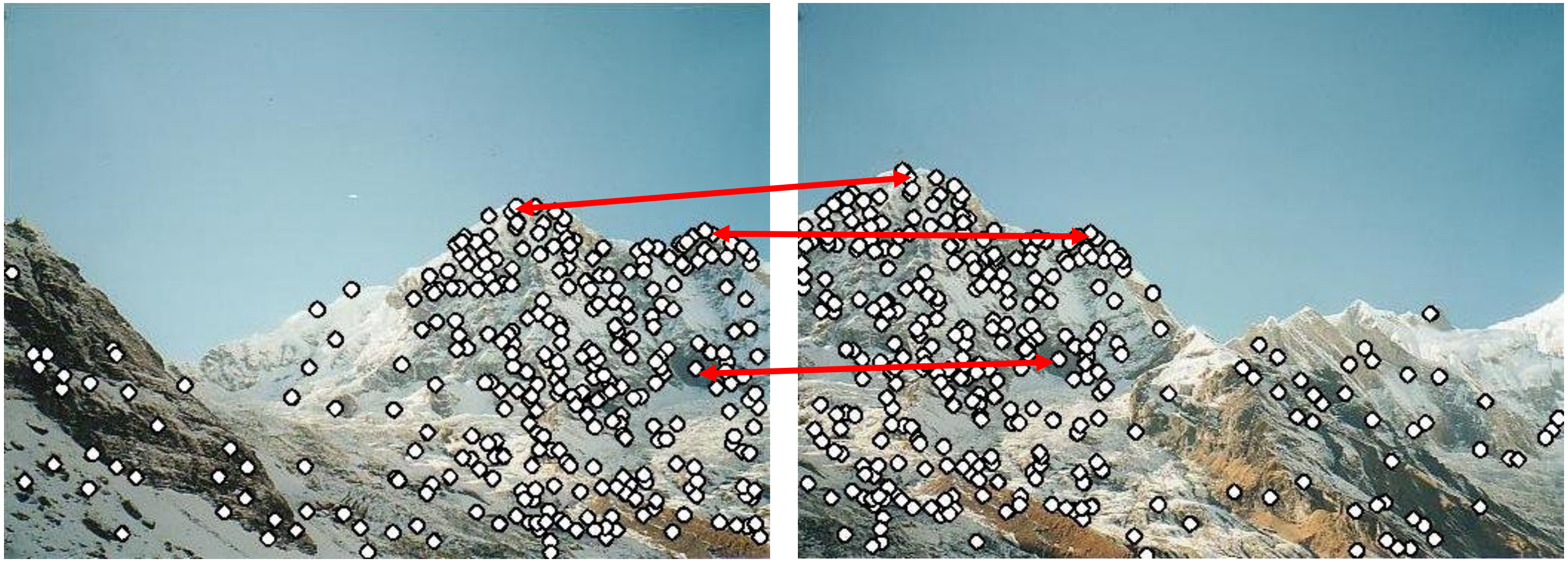


- Extract features

RANSAC for Homography

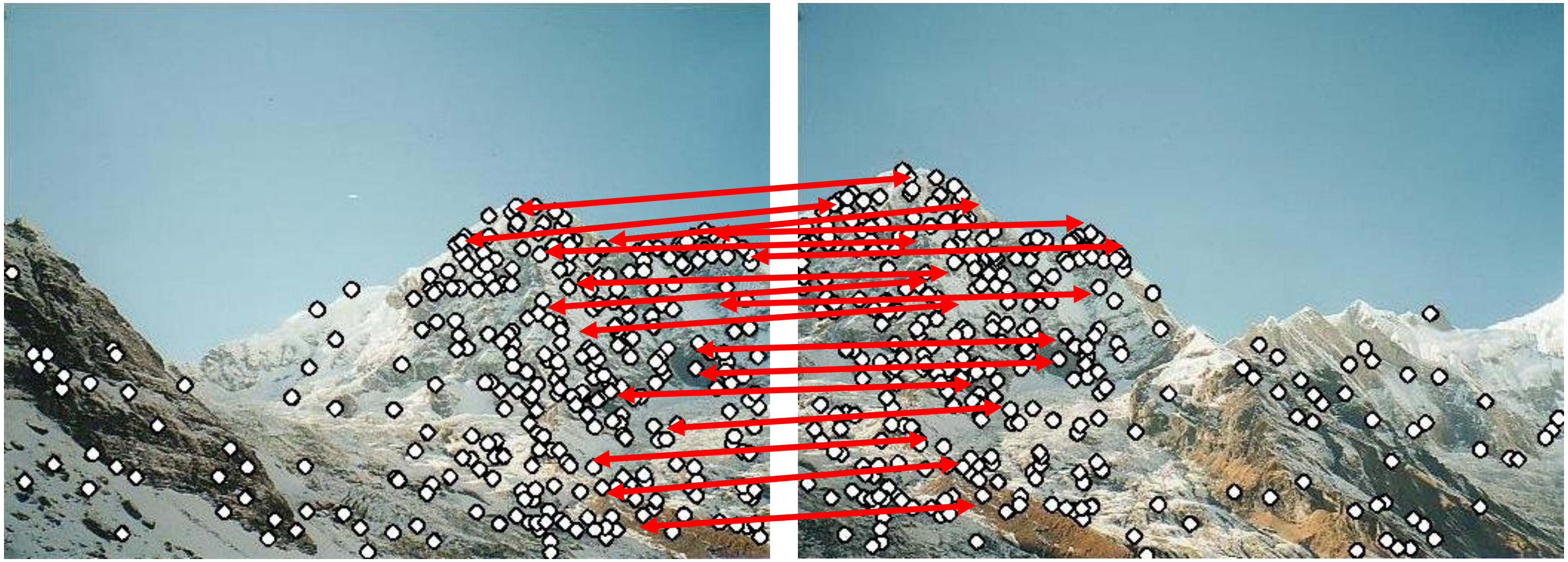
- Extract features
- Compute *putative matches*

RANSAC for Homography



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)

RANSAC for Homography



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation T (small group of putative matches that are related by T)
  - *Verify* transformation (search for other matches consistent with T)

RANSAC for Homography

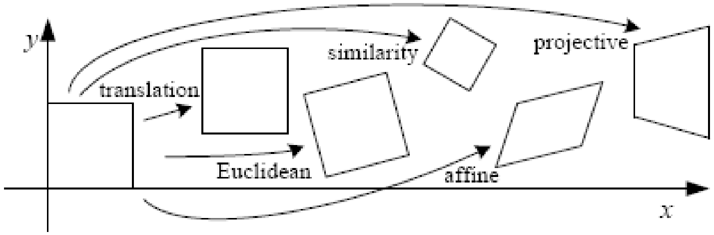
- Extract features
- Compute *putative matches*
- Loop:

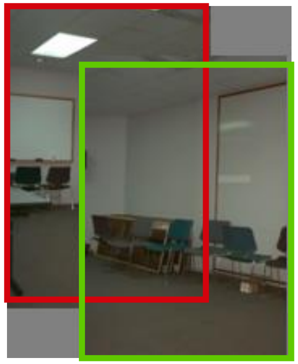
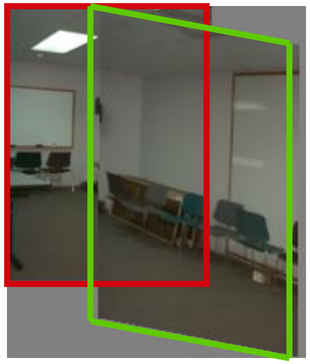
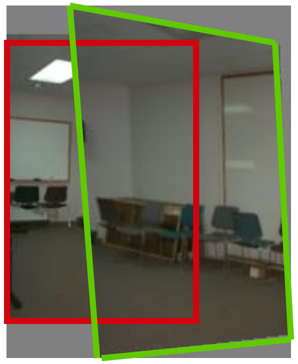
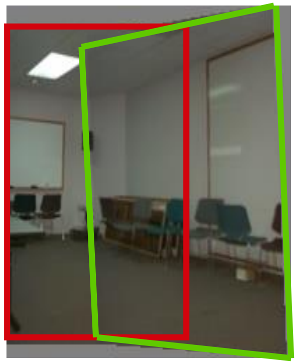
  b = Ax

  - *Hypothesize* transformation (small group of putative matches that are related by x)
  - *Verify* transformation (search for other matches consistent with x)

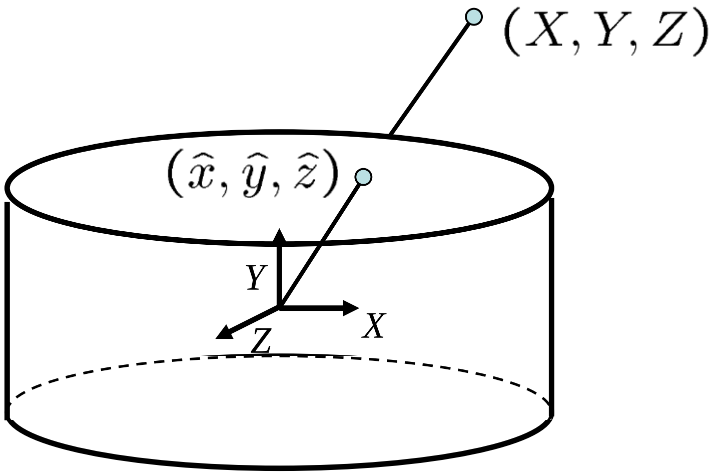
Large-Scale Mosaics

- Motion models

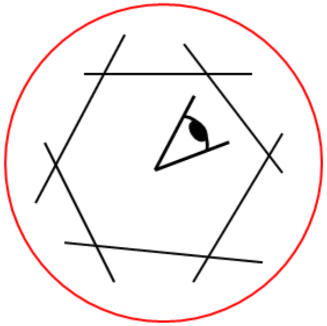


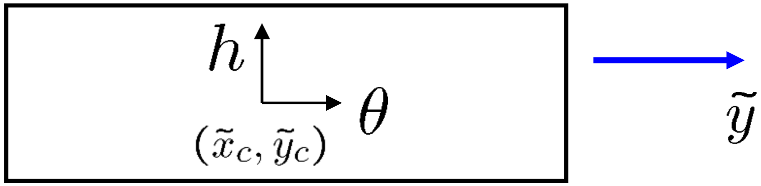
Translation	Affine	Perspective	3D rotation
			
2 unknowns	6 unknowns	8 unknowns	3 unknowns

Cylindrical Projection for Panorama Stitching



unit cylinder





unwrapped cylinder

- 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

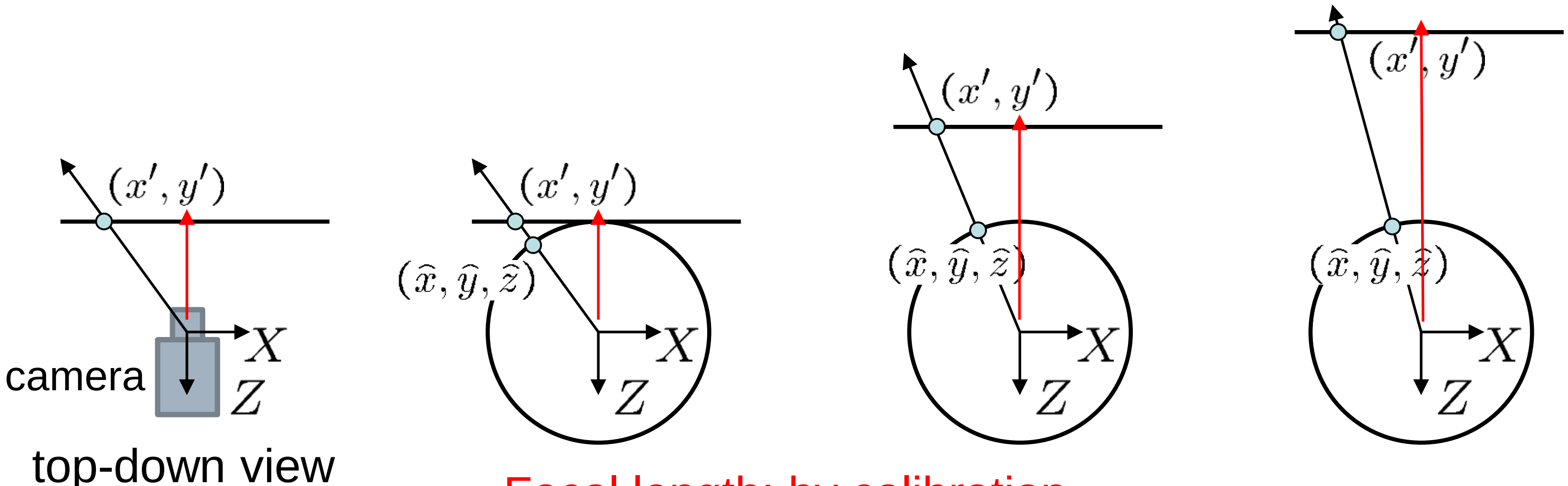
$$(sin\theta, h, cos\theta) = (\hat{x},\hat{y},\hat{z})$$

- Convert to cylindrical image coordinates

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

cylindrical image

Cylindrical Reprojection

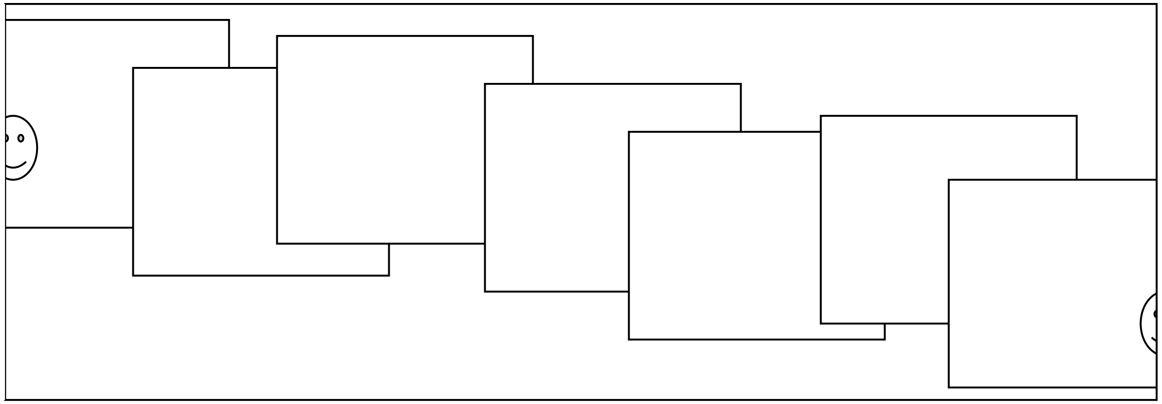


Focal length: by calibration

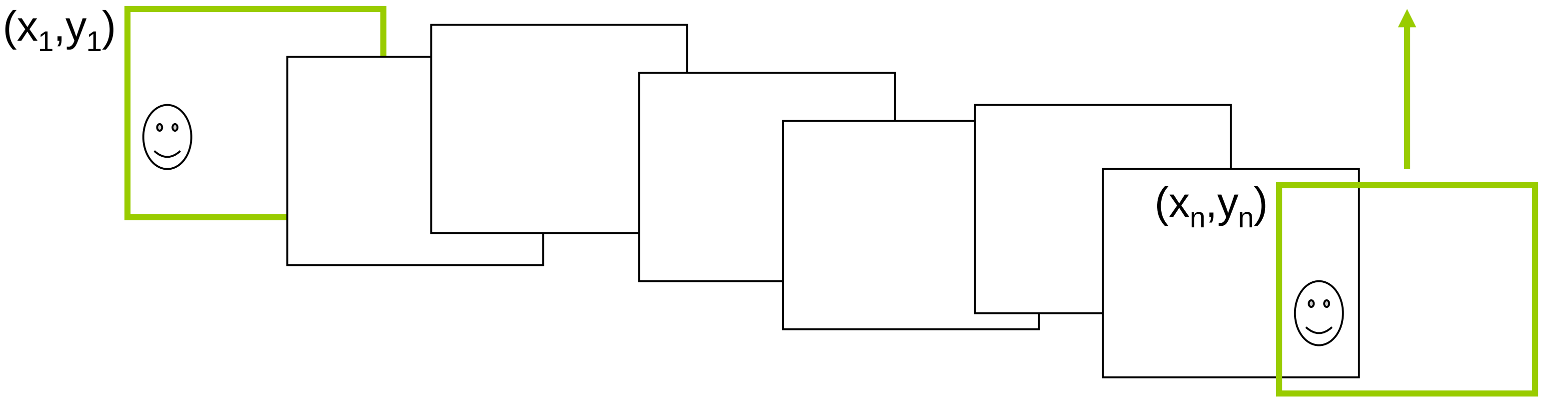
Image 384x300 f = 180 (pixels) f = 280 f = 380

Drift Problem

- Error accumulation
  - small errors accumulate over time



Drift Problem



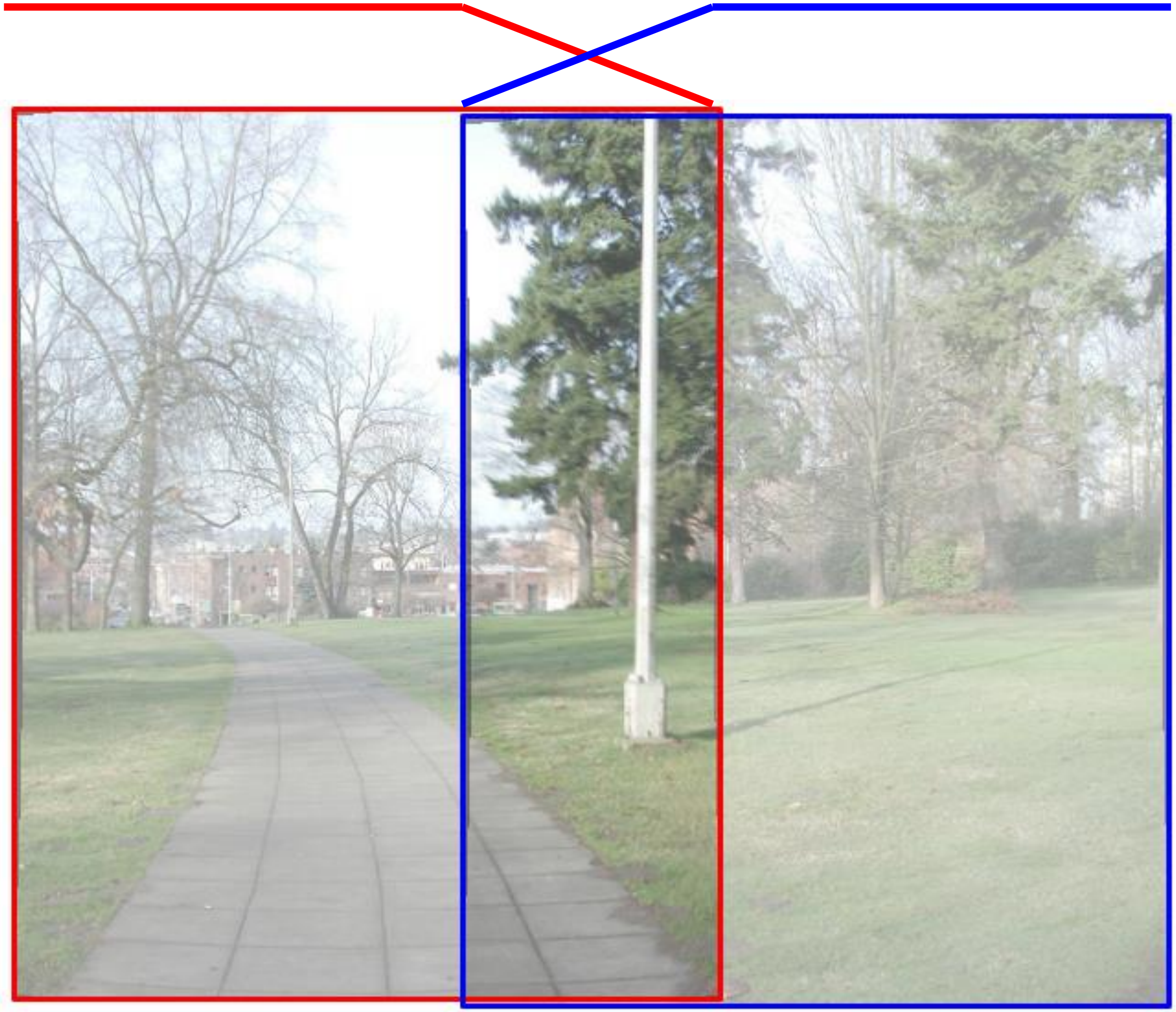
- copy of first image

- 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"

Blending

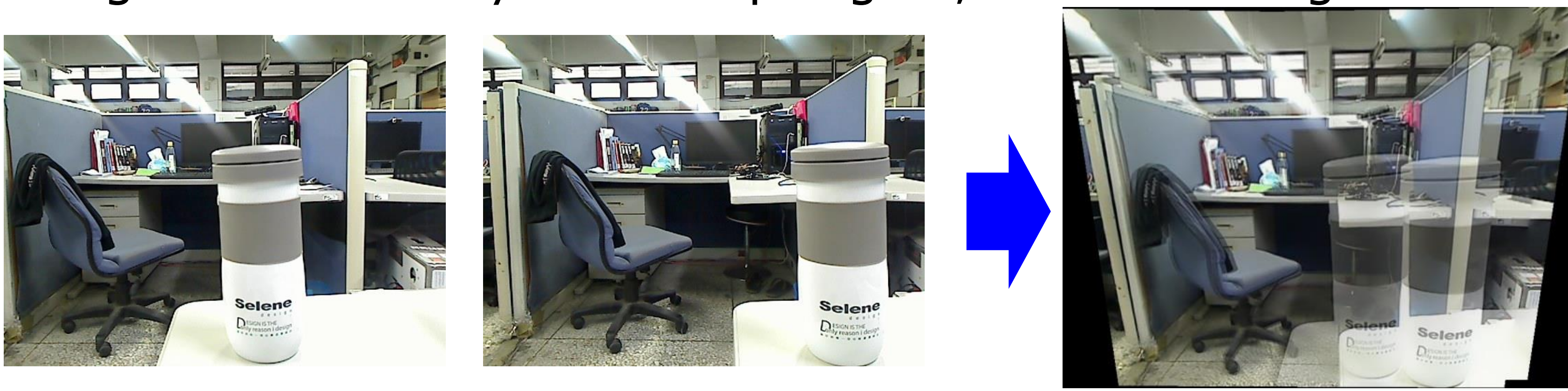
- Why blending: parallax, lens distortion, scene motion, exposure difference

Linear Blending



Motion(Visual) Parallax, Ghosting Effect

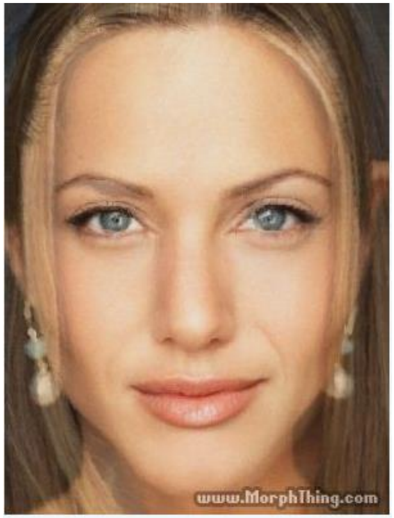
- Large inconsistency in overlap region, after blending …

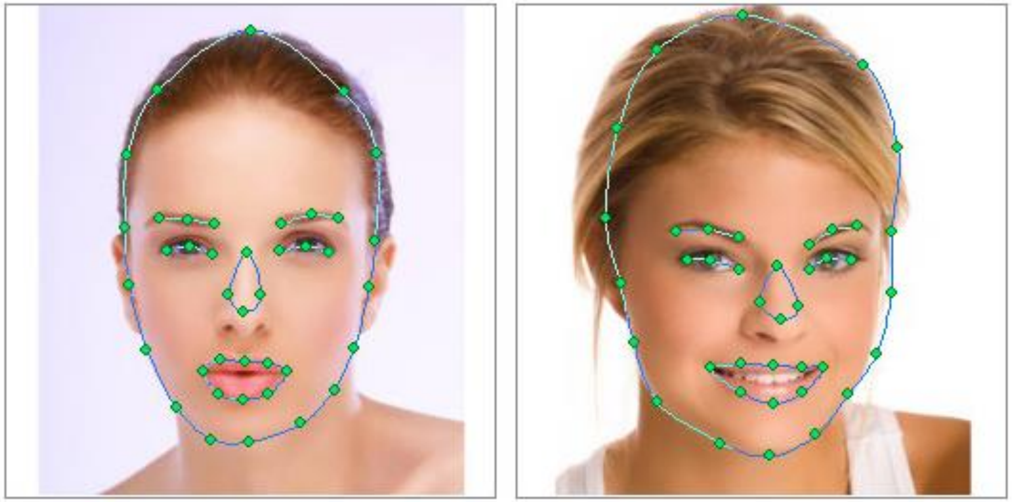


- Solution
  - Find a seam path avoid inconsistency in overlap region
  - Cut off the overlap region of each image

Face Morphing

- Face recognition
- Feature point detection
- Image warping
- Color blending





www. morphthing.com

www.facemorpher.com