MAEG 5720:
Computer Vision in Practice

Lecture 9:

Image Stitching

Dr. Terry Chang

2021-2022

Semester 1

香港中文大學
The Chinese University of Hong Kong

Department of Mechanical and Automation Engineering
機械與自動化工程學系

Content

- What is Image Stitching?
- Rational Stitching
- Feature-based Stitching
- Applications

Many slides have been borrowed from Kristen Grauman, Richard Szeliski, Steve Seitz and Y.Y Chuang@ cmu

What is Image Stitching?

- Stitching = alignment + blending

geometrical registration

photometric registration

Motivation

- To create a single image with wider Field of View and more detail.
- Blend together several overlapping images into one seamless mosaic

To recreate the background

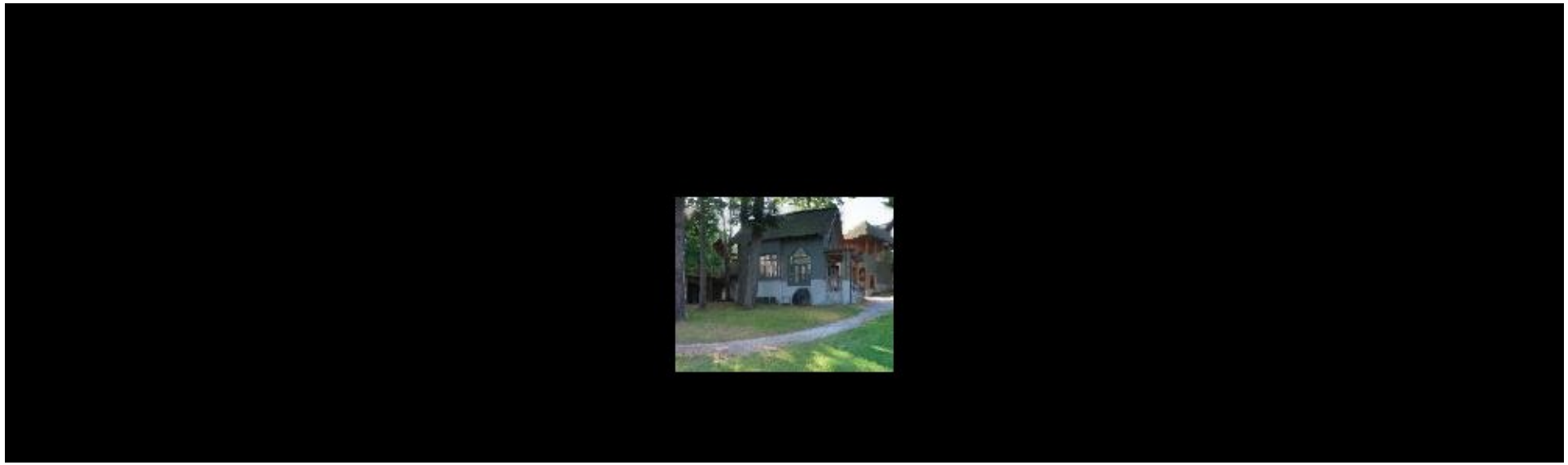
+

+

+ … +

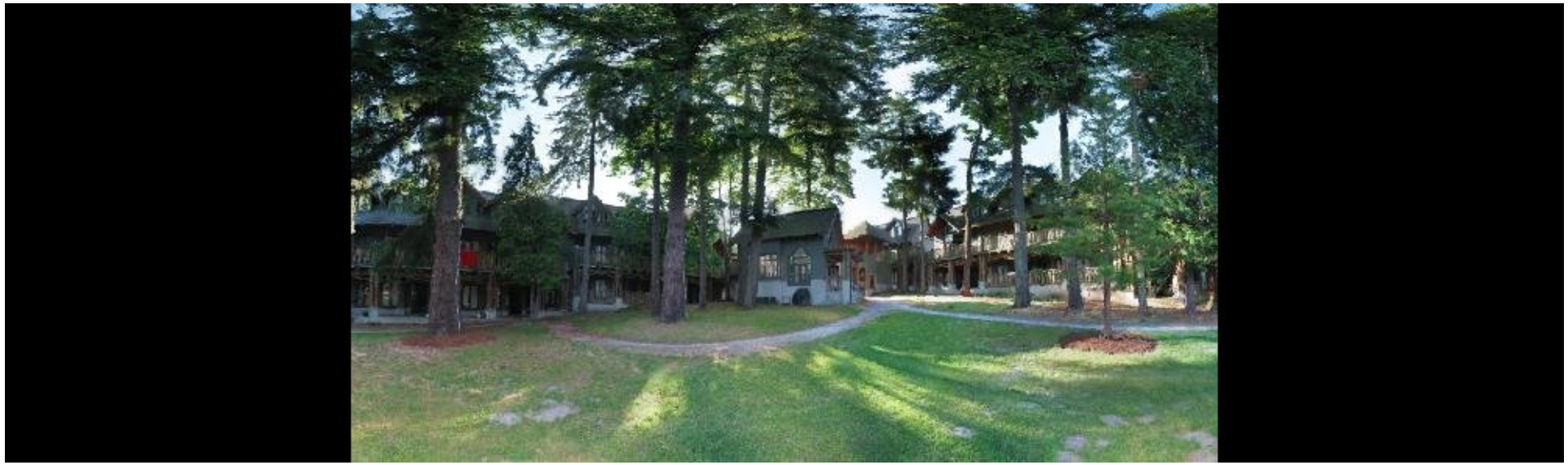
Why stitching?

- Are you getting the whole picture?
  - Compact Camera FOV



Why stitching?

- Are you getting the whole picture?
  - Compact Camera FOV
  - Human FOV



Why panorama?

- Are you getting the whole picture?
  - Compact Camera FOV
  - Human FOV
  - Panoramic Mosaic

Other stitching examples

- Similar to HDR, it is a topic of computational photography, seeking ways to build a better camera using either hardware or software.
- Most consumer cameras have a panorama mode

  http://www.360cities.net/
  http://maps.google.com.hk/

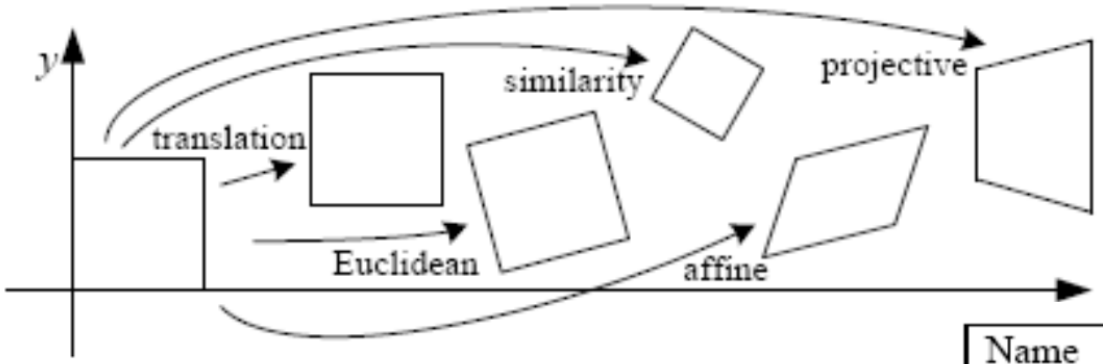
The Procedure for image stitching

- Basic Procedure
  1. Take a sequence of images from the same position.
     (Rotate the camera about its optical center (entrance pupil)).
  2. Robustly compute the homography transformation between second image and first.
  3. Transform (warp) the second image to overlap with first.
  4. Blend the two together to create a mosaic.
  5. If there are more images, repeat 2~4 for .

Motion Models

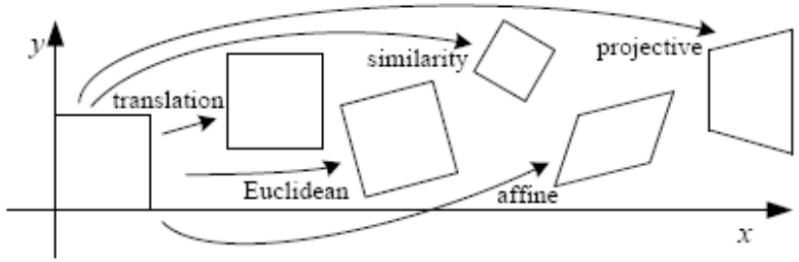
- Given: Two images taken with a camera
- Aim: We need to align them
- We consider a transformation/ warping between them
  - translation?
  - Rotation?
  - Scale?
  - Affine?
  - Perspective?

Motion Models

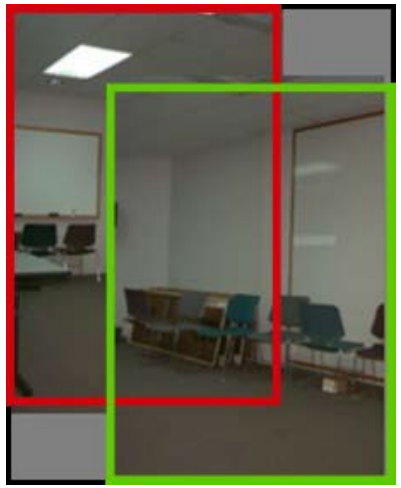


Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times3}$	2	orientation + ···	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times3}$	3	lengths + ···	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times3}$	4	angles + ···	
affine	$\begin{bmatrix}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

The motion

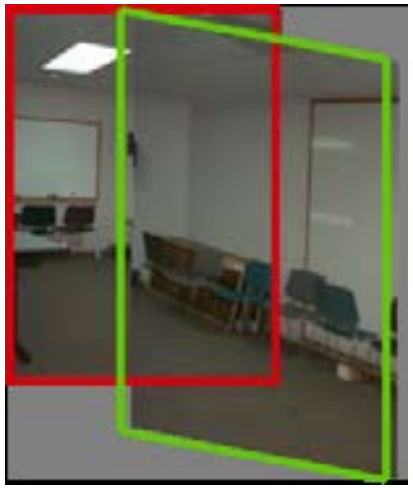


Translation



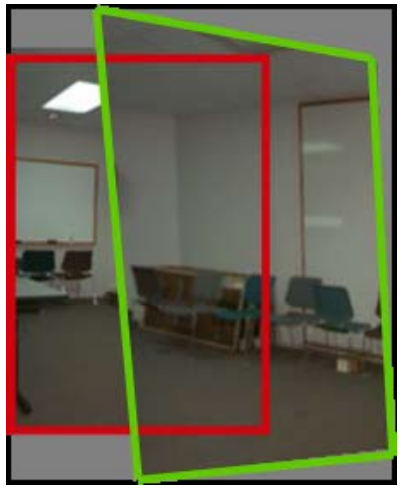
2 dof

Affine



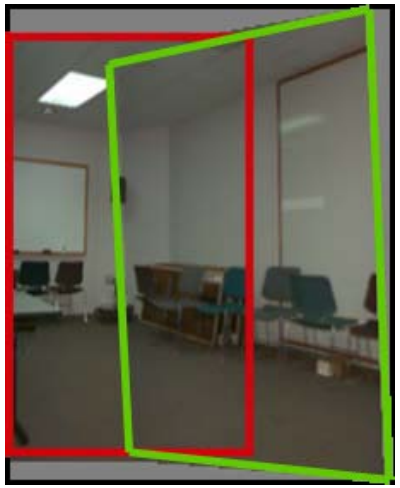
6 dof

Perspective



8 dof

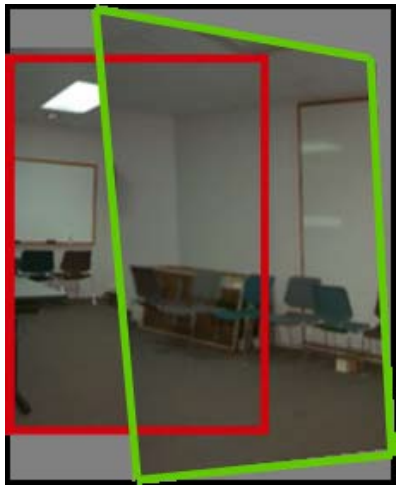
3D Rotation



3 dof

Plane perspective Mosaics

- 8-parameter generalization of affine motion
  - Work for pure rotation or planar surface
- Limitations:
  - Slow convergence
  - Difficult to control interactively



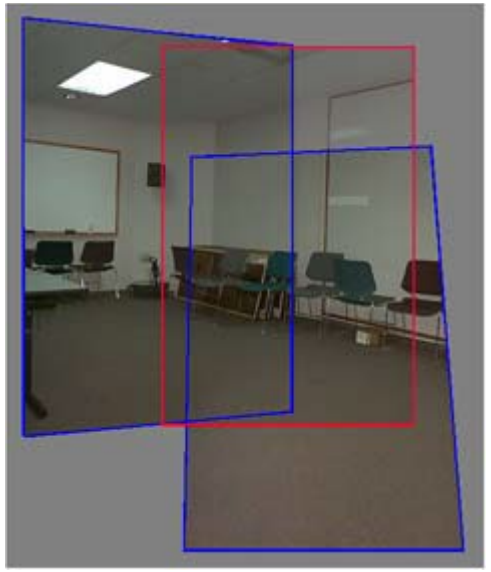
Method to establish correspondence

- Direct Method:
  - Use generalization of affine motion model
    [Szeliski & Shum ’97]
- Feature-based method
  - Compute feature-based correspondence
    [Lowe ICCV99, Brown & Lowe ICCV2003]
  - Compute ***R*** from correspondences

Rotational Stitching

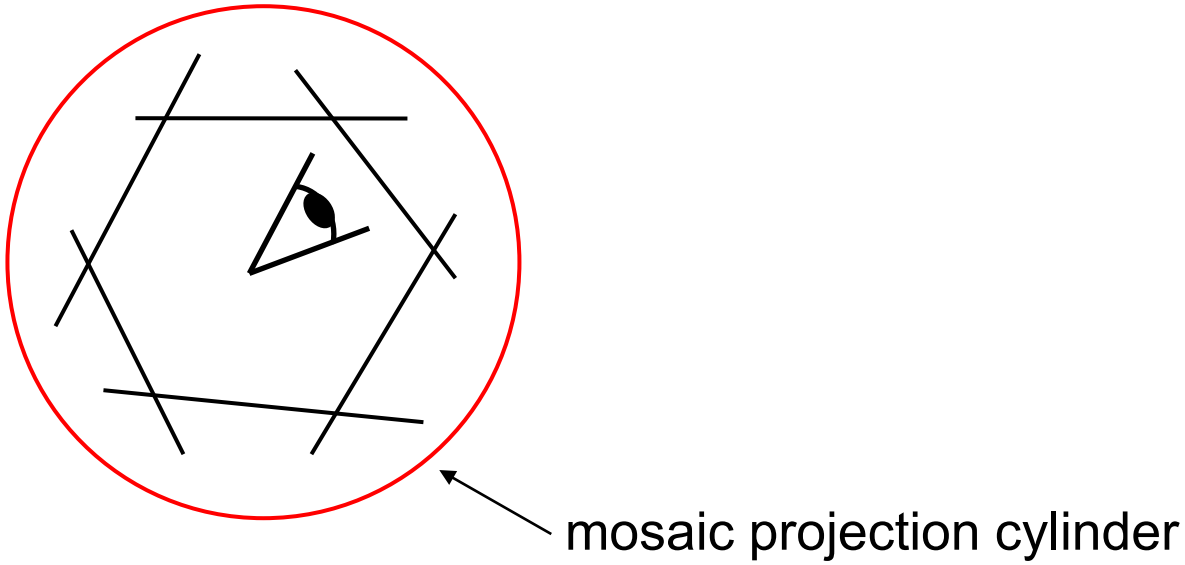
Rotational Stitching/Mosaics

- Directly optimize rotation and focal length
- Advantages:
  - Ability to build full-view panoramas
  - Easier to control interactively
  - More stable and accurate estimation

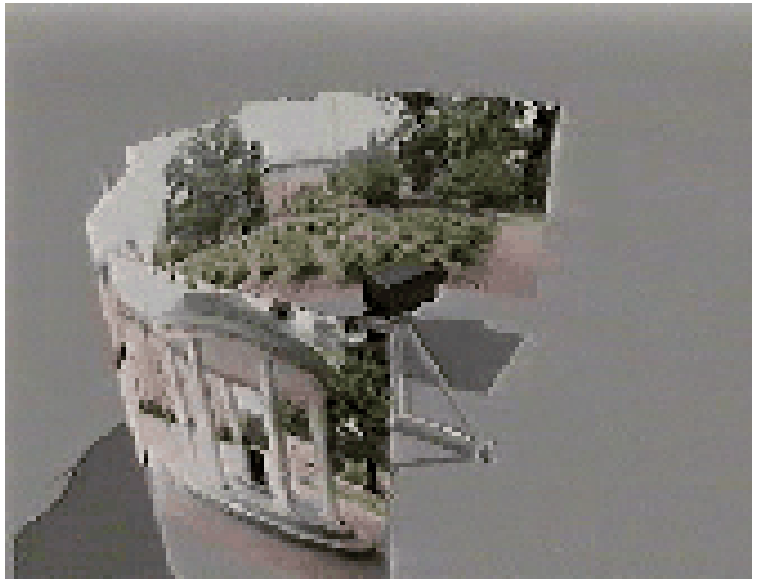


A case study: cylindrical panorama

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



Cylindrical Panoramas



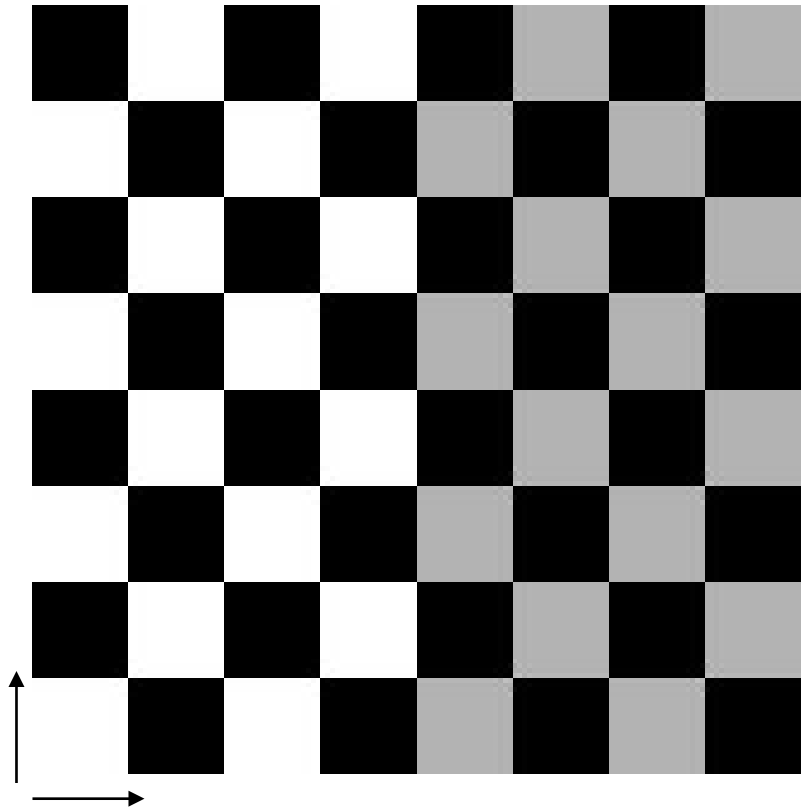
- Steps:
  - Take pictures by rotating the optical centre of the camera.
  - Project each image onto a cylinder (warping)
  - Estimate motion (a pure translation now)
  - Blend
  - Optional: project it back (unwarp)
  - Output the resulting mosaic

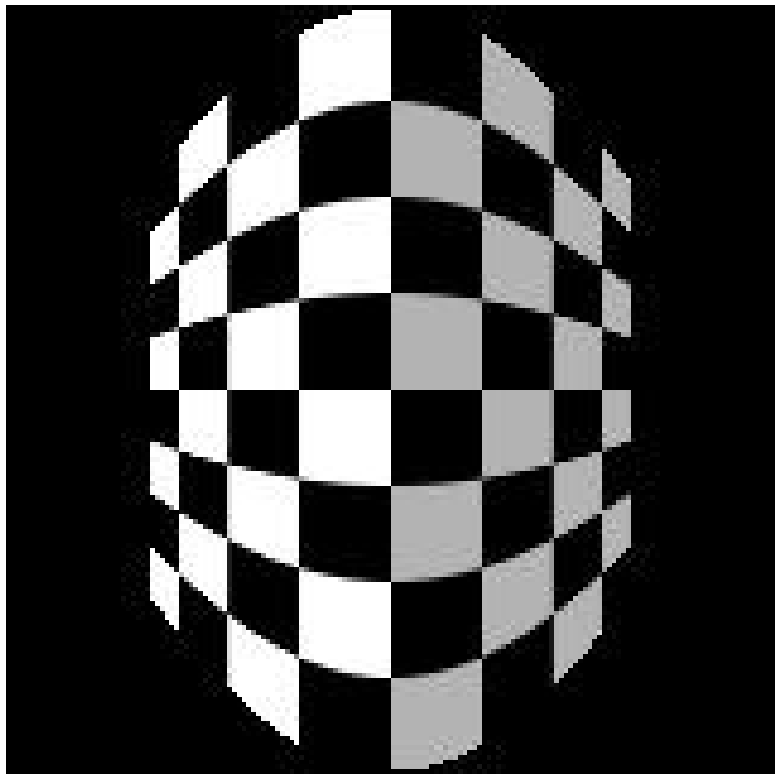
Taking pictures




Kaidan panoramic tripod head

Cylindrical Projection





The image acquired by a rotating camera

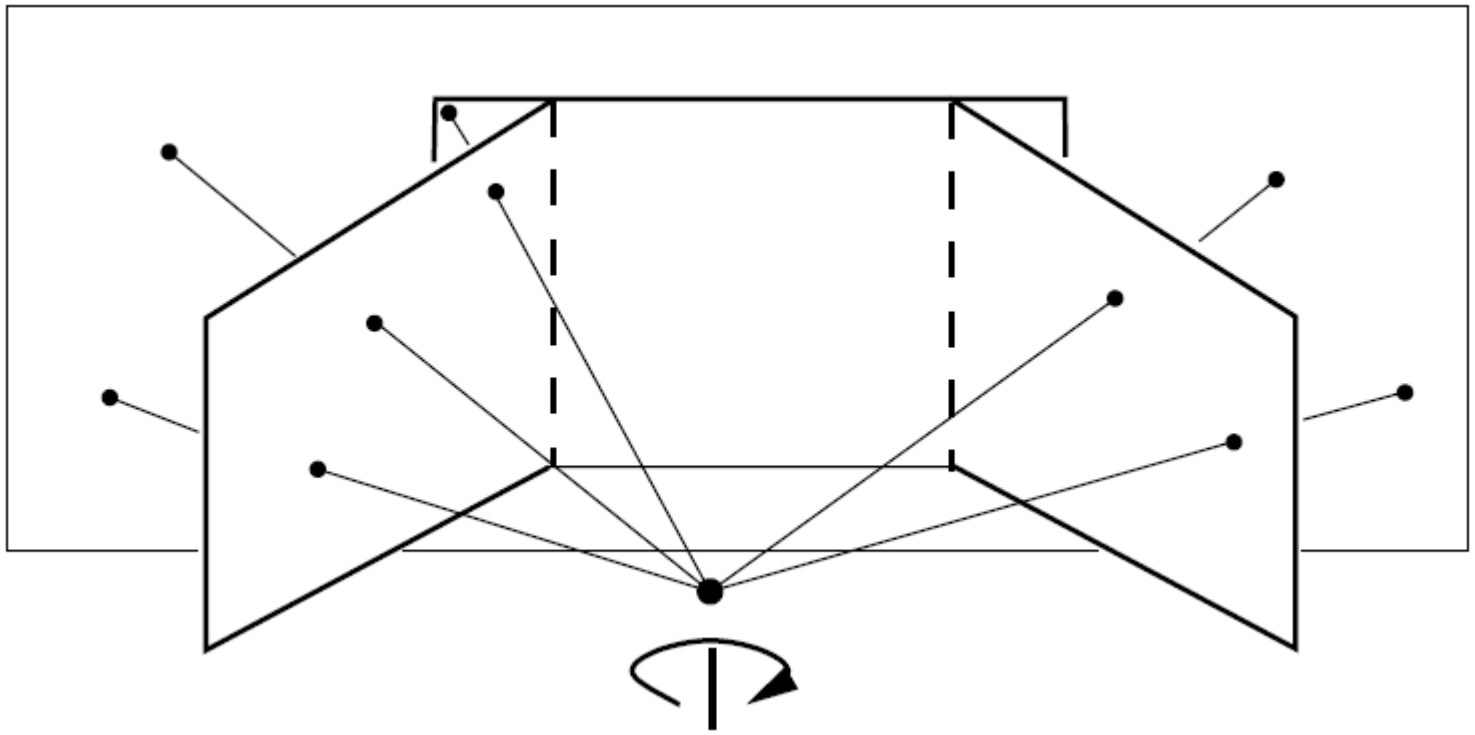


Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Cylindrical Panoramas

- Map image to cylindrical or spherical coordinates
  - We need to 'know' focal length
  - Work only if a single tilt (camera rotate at projection centre)

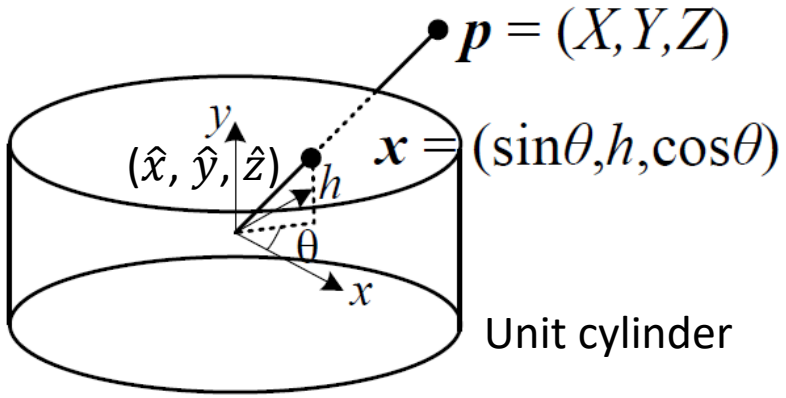
Image 384x300

F=180 (pixels)

F=280 (pixels)

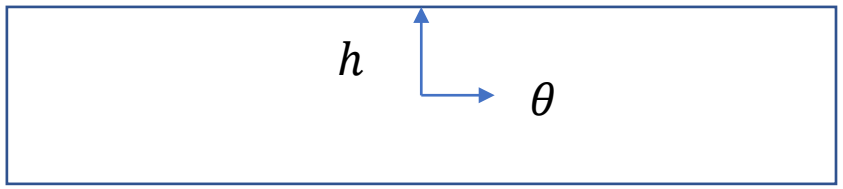
F=380 (pixels)

Cylindrical Projection

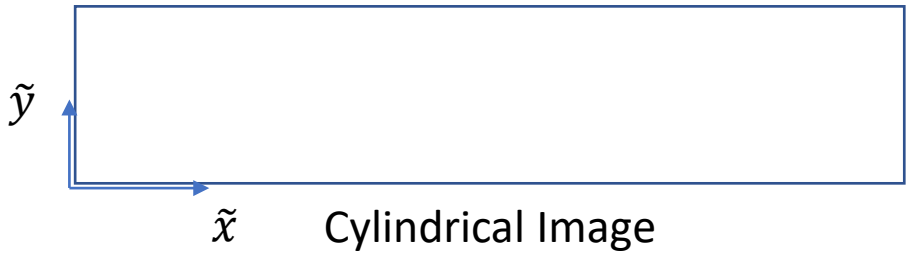


Unit cylinder

- 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 cylindrical image coordinates
- $(\tilde{x}, \tilde{y}) = (s\theta, sh) + (\widetilde{x_c}, \widetilde{y_c})$
- s defines the size of the final image

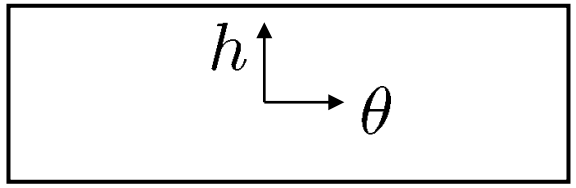


Unwrapped Cylinder

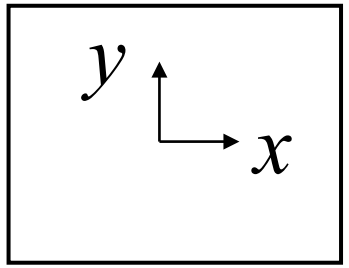


Cylindrical Image

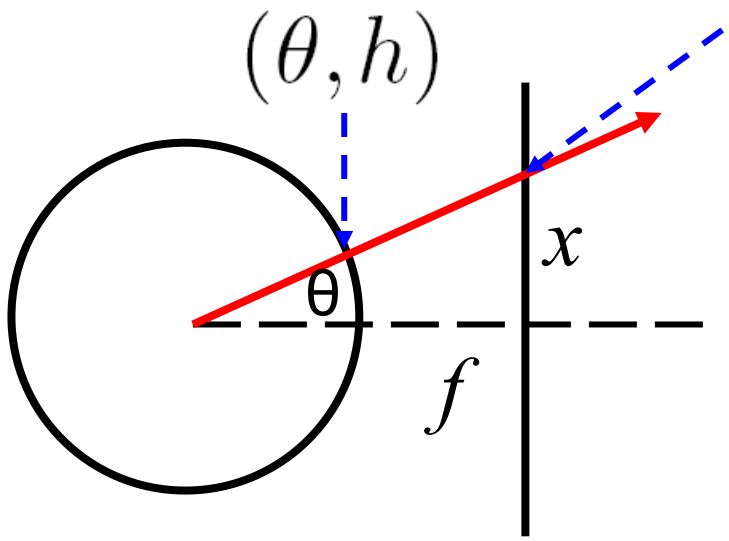
Cylindrical projection



unwrapped cylinder

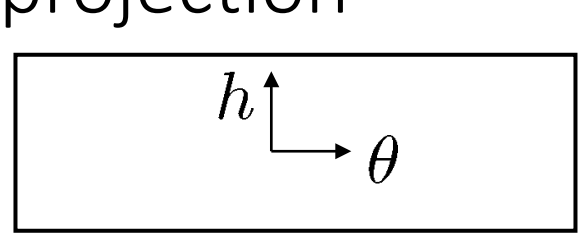


$$(\sin\theta, h, \cos\theta) \propto (x, y, f)$$

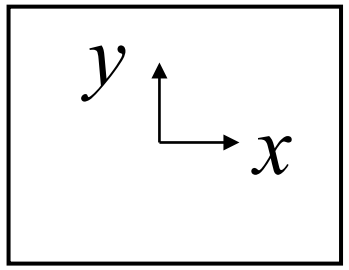


$$\theta = \tan^{-1}\frac{x}{f}$$

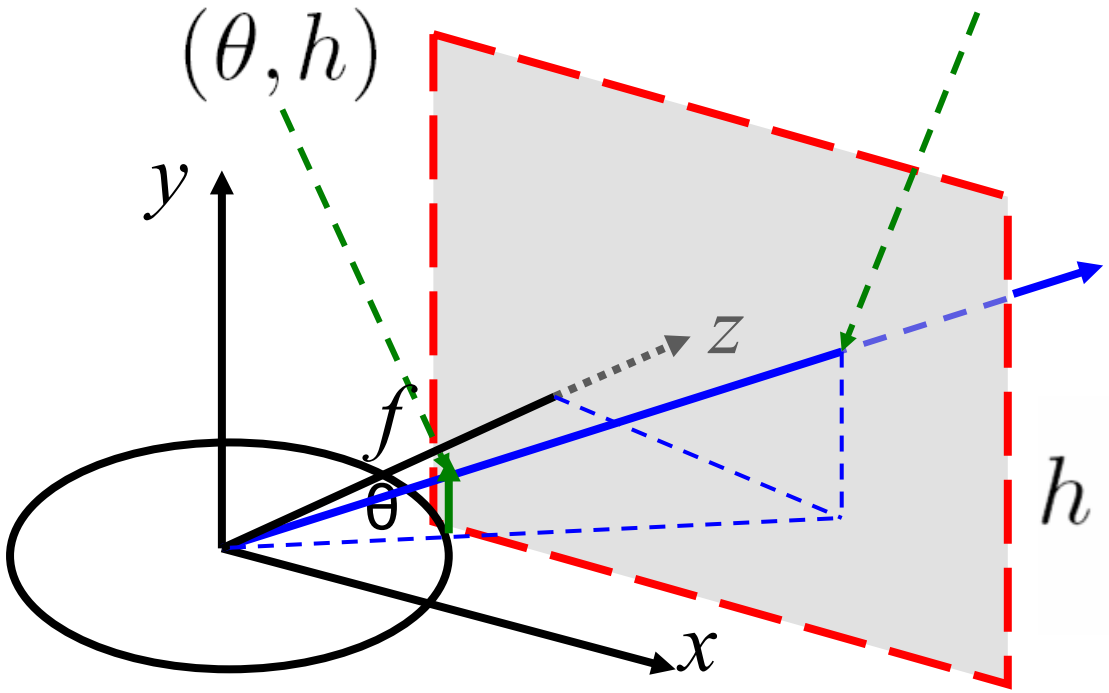
Cylindrical projection



unwrapped cylinder

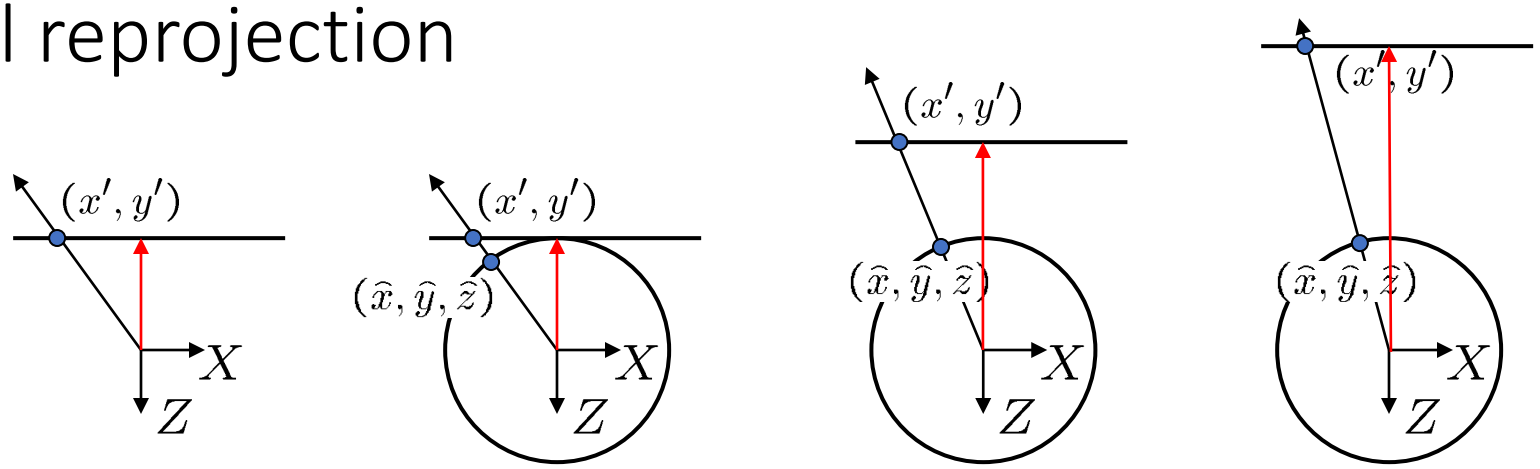


$$(\sin\theta, h, \cos\theta) \propto (x, y, f)$$



$$h = \frac{y}{\sqrt{x^2 + f^2}}$$

Cylindrical reprojection



top-down view

Focal length – the dirty secret…

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

Determining the focal length

- Initialize from homography H (Szeliski and Shum 1997)
- Use camera’s EXIF tag
- Try and Error

Practical Method for F

- Use program jhead
  - (http://www.sentex.net/~mwandel/jhead/)
- Mac, Windows, and Linux
- Sample outputs

Sample jhead output:

```
File name    : 0805-153933.jpg
File size    : 463023 bytes
File date    : 2001:08:12 21:02:04
Camera make  : Canon
Camera model : Canon PowerShot S100
Date/Time    : 2001:08:05 15:39:33
Resolution   : 1600 x 1200
Flash used   : No
Focal length :  5.4mm  (35mm equivalent: 36mm)
CCD Width    : 5.23mm
Exposure time: 0.100 s  (1/10)
Aperture     : f/2.8
Focus Dist.  : 1.18m
Metering Mode: center weight
Jpeg process : Baseline
```

Matlab Example

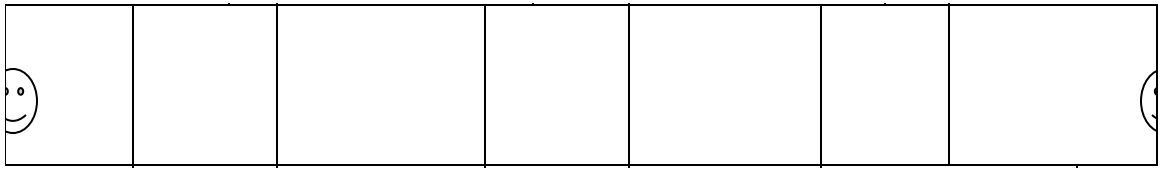
```
imgRGB = imread('peppers.png');
[imgInd,map] = rgb2ind(imgRGB,256);
[imgIndRows,imgIndCols] = size(imgInd);
[X,Y,Z] = cylinder(imgIndRows,imgIndCols);
surface(X,Y,Z,flipud(imgInd),...
    'FaceColor','texturemap',...
    'EdgeColor','none',...
    'CDataMapping','direct')
colormap(map)
view(-35,45)
```

```
imgRGB = imread('peppers.png');
[imgRows,imgCols,imgPlanes] = size(imgRGB);
[X,Y,Z] = cylinder(imgRows,imgCols);
warp(X,Y,Z,imgRGB);
```

Input images

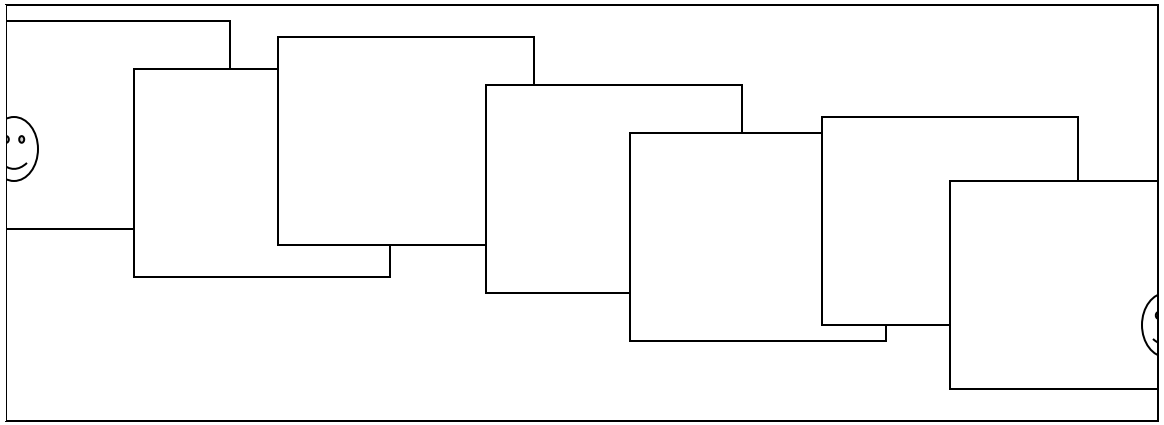
Cylindrical warping

Assembling the panorama



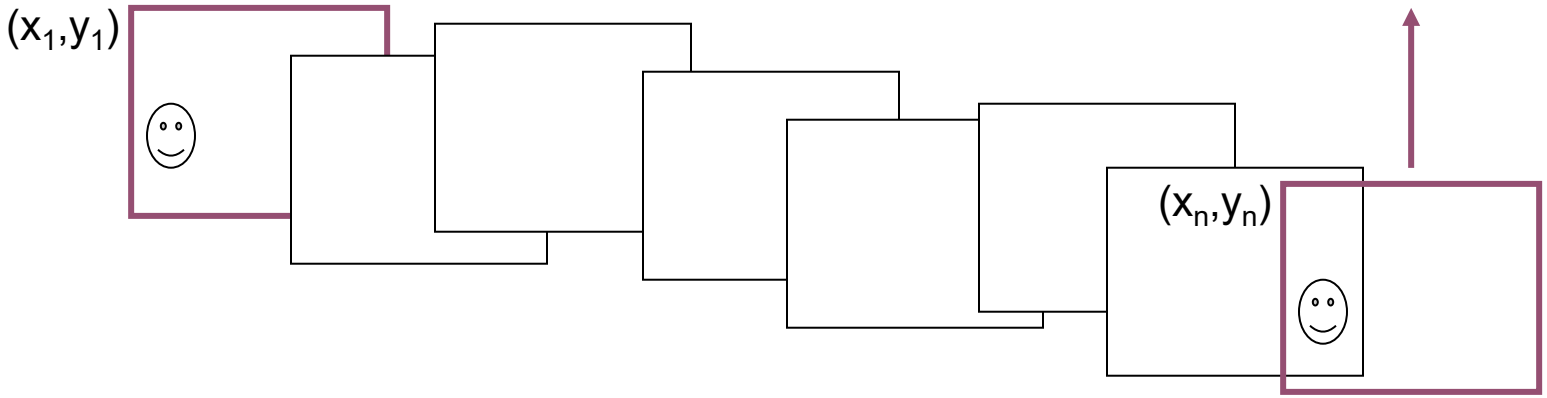
- Stitch pairs together, blend, then crop

Problem: Drift



- Error accumulation
  - small errors accumulate over time

Problem: Drift



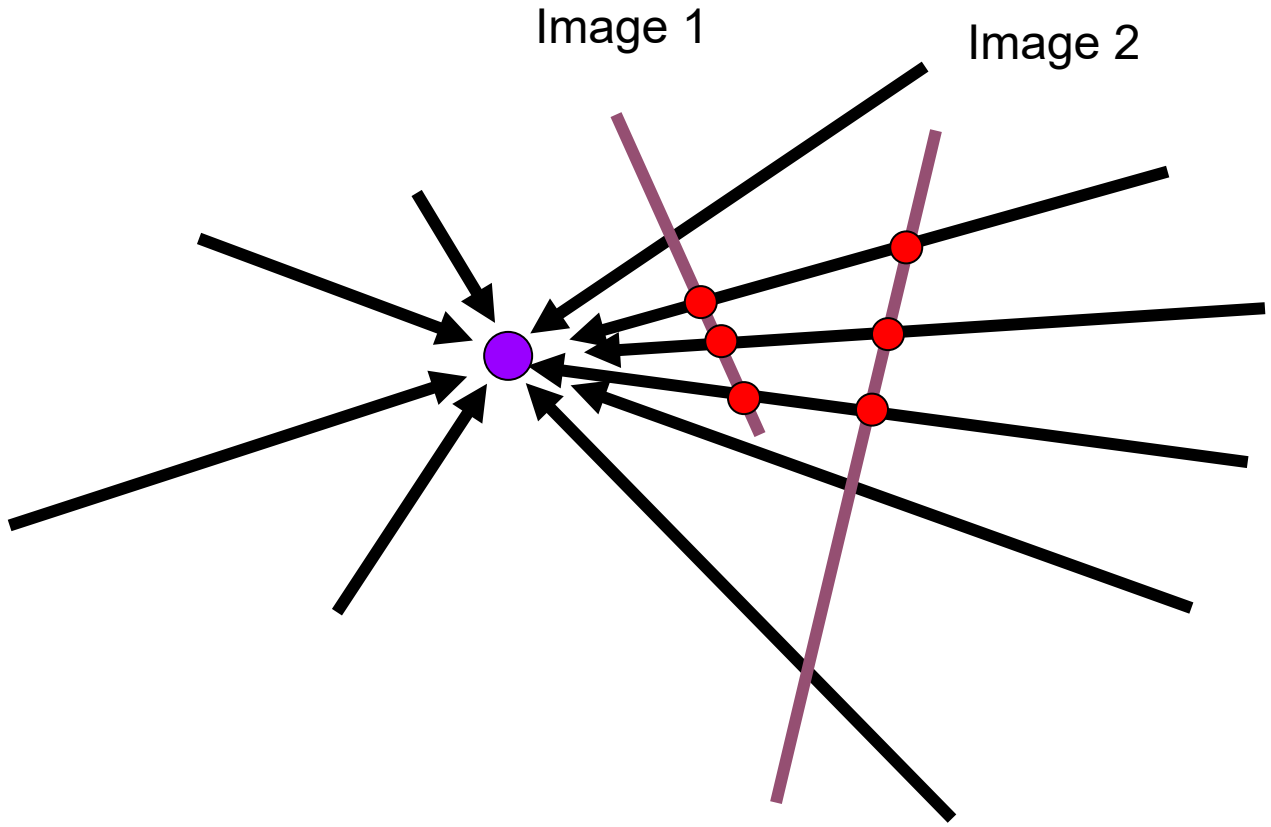
- copy of first image

- 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

Feature Based Methods

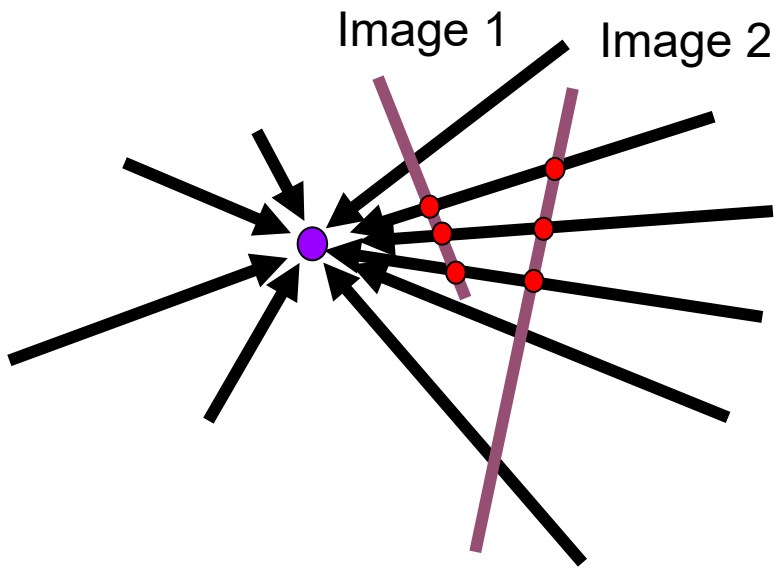
Two Images



How to estimate the transformation?

What is the transformation?

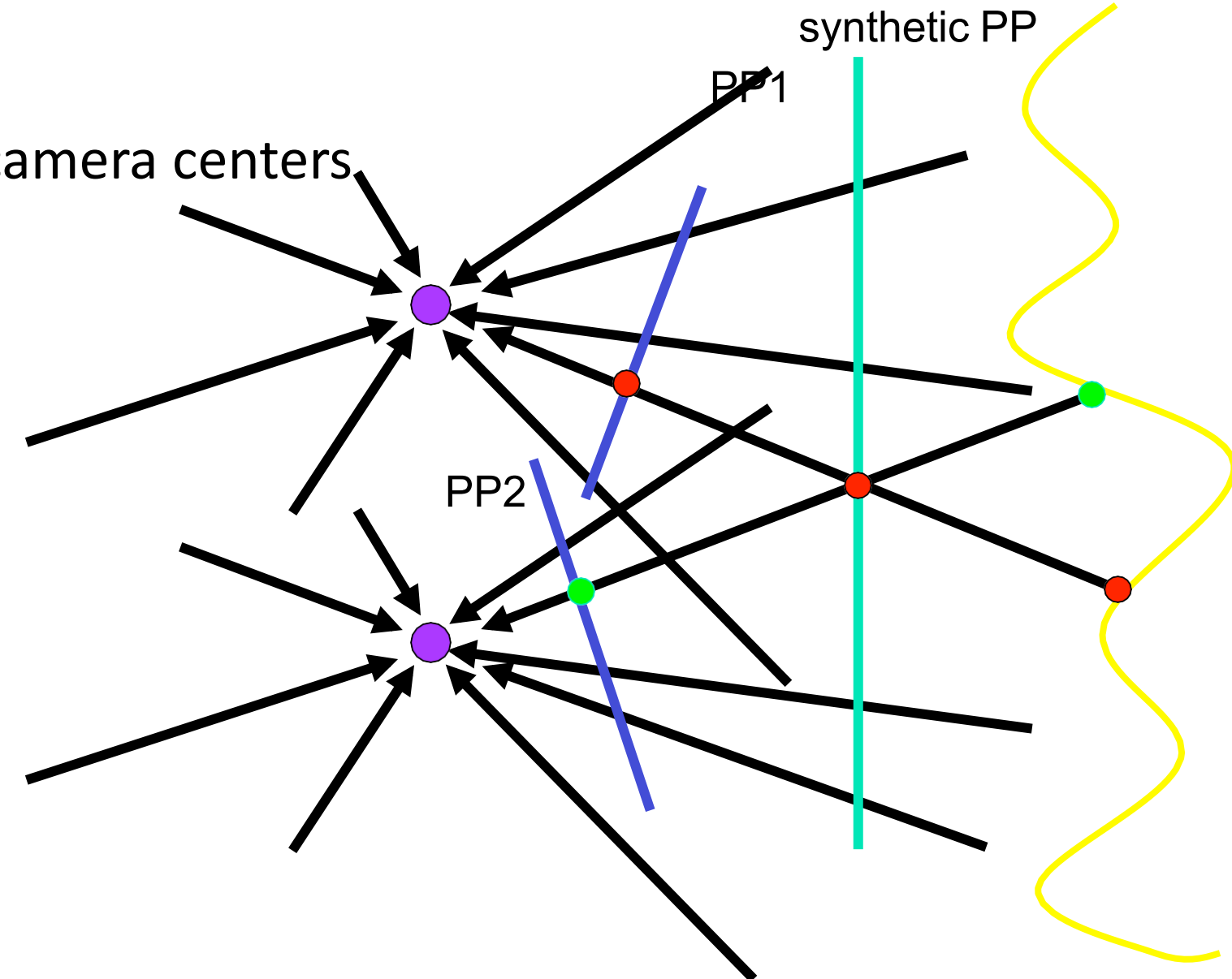
- For image 1, $\mathbf{x_1} = \mathrm{P_1}\mathbf{X} \Rightarrow \mathbf{X} = \mathrm{P_1}^{-1}\,\mathbf{x_1}$
- For Image 2, $\mathbf{x_2} = \mathrm{P_2}\mathbf{X} \Rightarrow \mathbf{X} = \mathrm{P_2}^{-1}\,\mathbf{x_2}$
- We have



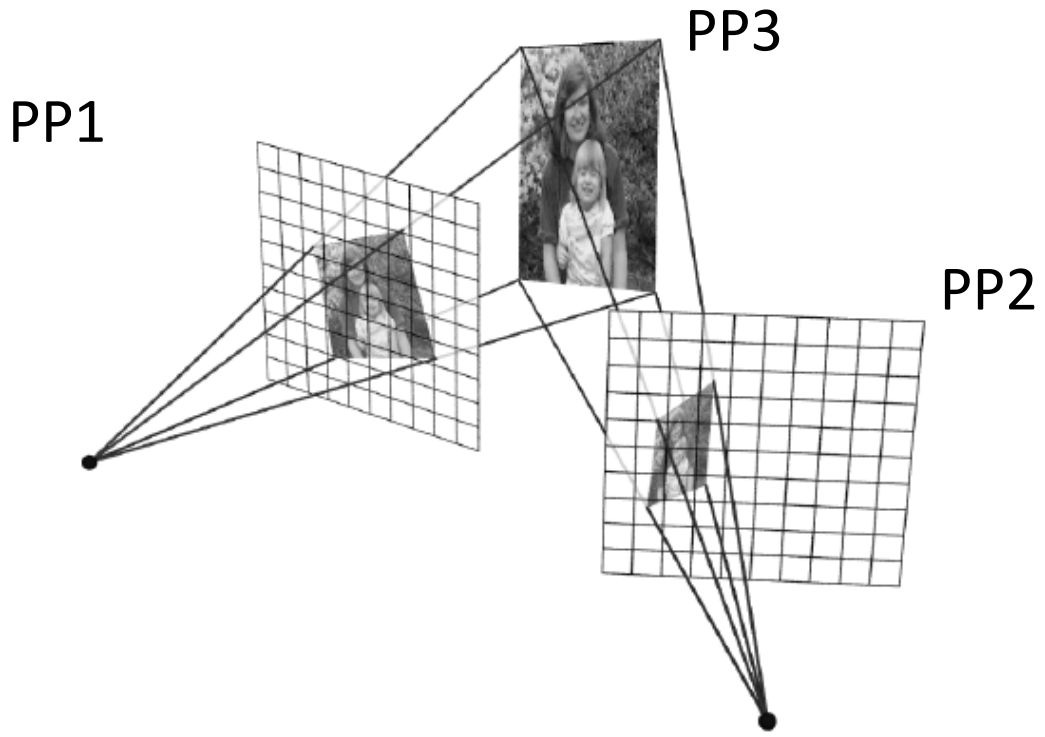
- $\mathrm{P_1}^{-1}\,\mathbf{x_1} = \mathrm{P_2}^{-1}\,\mathbf{x_2}$
- $\mathbf{x_1} = \mathrm{P_1}\mathrm{P_2}^{-1}\,\mathbf{x_2}$
- Since $\mathbf{X}$ planar on a plane, we can simplify $\mathrm{P_1}$ & $\mathrm{P_2}$ to $\mathrm{H_{12}}$ which is 3x3 matrix.
- $\mathbf{x_1} = \mathrm{H_{12}}\mathbf{x_2}$

Changing camera center

- When we have different camera centers
- Does it still work?



…Or: 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

Source: Alyosha Efros

In Short, we can use homography

1) Planar Scene

Image credit: https://www.oxygen.com

1) The scene which is far and the depth variation is neglectable

Image credit: https://www.shutterstock.com

Homography: Direct Linear Transform

Original Image

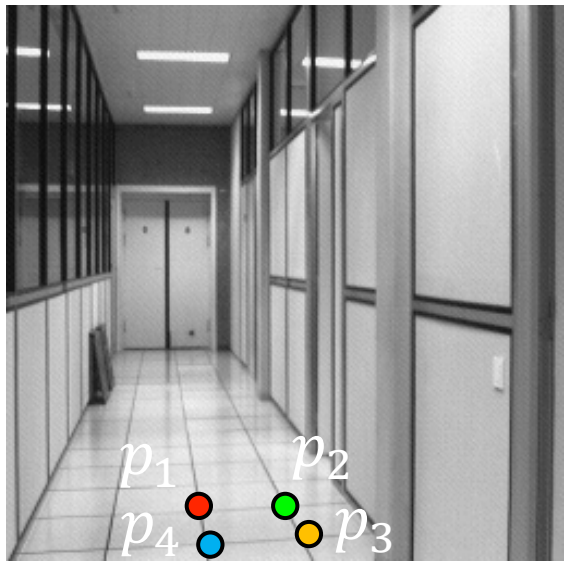
Fronto-parallel view of corridor wall

Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

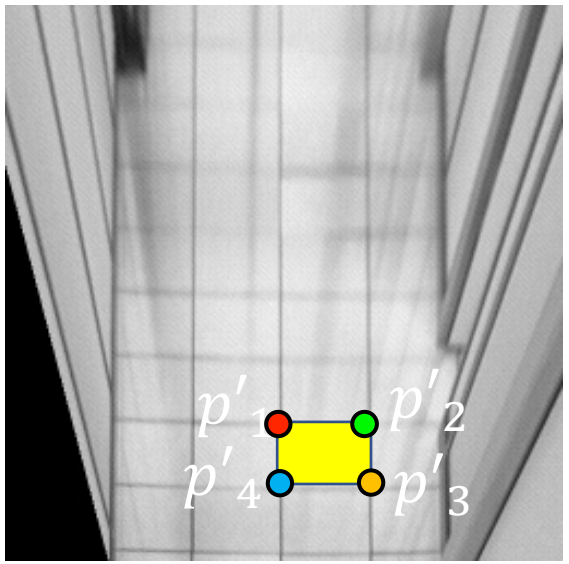
Homography: Direct Linear Transform

- Given the corresponding points p and p’, estimate the homography H such that

$$P' = HP$$



Original Image



Fronto-parallel view of corridor floor

Question: How many points do we need?

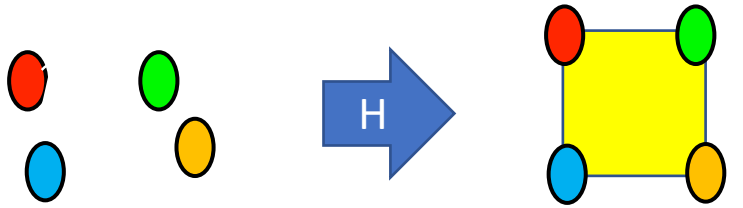
Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Synthetic View

- Steps:

- Compute the *homograhpy* H which maps the quadrilateral of the original image to a rectangle with correct aspect ratio.

- Projective warp the source image with this homography



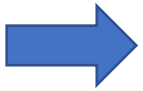
Determining the homography matrix

- The linear equation for the correspondence

$$P' = HP$$

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Expanding
  - $\lambda\, x' = h_1 x + h_2 y + h_3$
  - $\lambda y' = h_4 x + h_5 y + h_6$
  - $\lambda = h_7 x + h_8 y + h_9$



$$x' = \frac{h_1 x + h_2 y + h_3}{h_7 x + h_8 y + h_9}$$

$$y' = \frac{h_4 x + h_5 y + h_6}{h_7 x + h_8 y + h_9}$$

Rearranging the terms

- *The two equations*
  - $x' = \frac{h_1 x + h_2 y + h_3}{h_7 x + h_8 y + h_9}$
  - $y' = \frac{h_4 x + h_5 y + h_6}{h_7 x + h_8 y + h_9}$

Rearranging terms

$$x'(h_7 x + h_8 y + h_9) = h_1 x + h_2 y + h_3$$
$$y'(h_7 x + h_8 y + h_9) = h_4 x + h_5 y + h_6$$

- Finally

$$-h_1 x - h_2 y - h_3 + x' x h_7 + x' y h_8 + x' h_9 = 0$$
$$-h_4 x - h_5 y - h_6 + y' x h_7 + y' y h_8 + y' h_9 = 0$$

- For each point, we have two equations

$$-h_1 x - h_2 y - h_3 + x'xh_7 + x'yh_8 + x'h_9 = 0$$
$$-h_4 x - h_5 y - h_6 + y'xh_7 + y'yh_8 + y'h_9 = 0$$

- In matrix form:

$$\mathbf{A}_i \boldsymbol{h} = 0$$

$$\mathbf{A}_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & x'x & x'y & x' \\ 0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y' \end{bmatrix}$$

$$\boldsymbol{h} = [h_1 \quad h_2 \quad h_3 \quad h_4 \quad h_5 \quad h_6 \quad h_7 \quad h_8 \quad h_9]^T$$

Determining homograhy matrix h

- Stacking All correspondence points

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & x'x & x'y & x' \\ 0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y' \end{bmatrix}$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & x'x & x'y & x' \\ 0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y' \end{bmatrix}$$

$$\vdots$$

$$\begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & x'x & x'y & x' \\ 0 & 0 & 0 & -x & -y & -1 & y'x & y'y & y' \end{bmatrix}$$

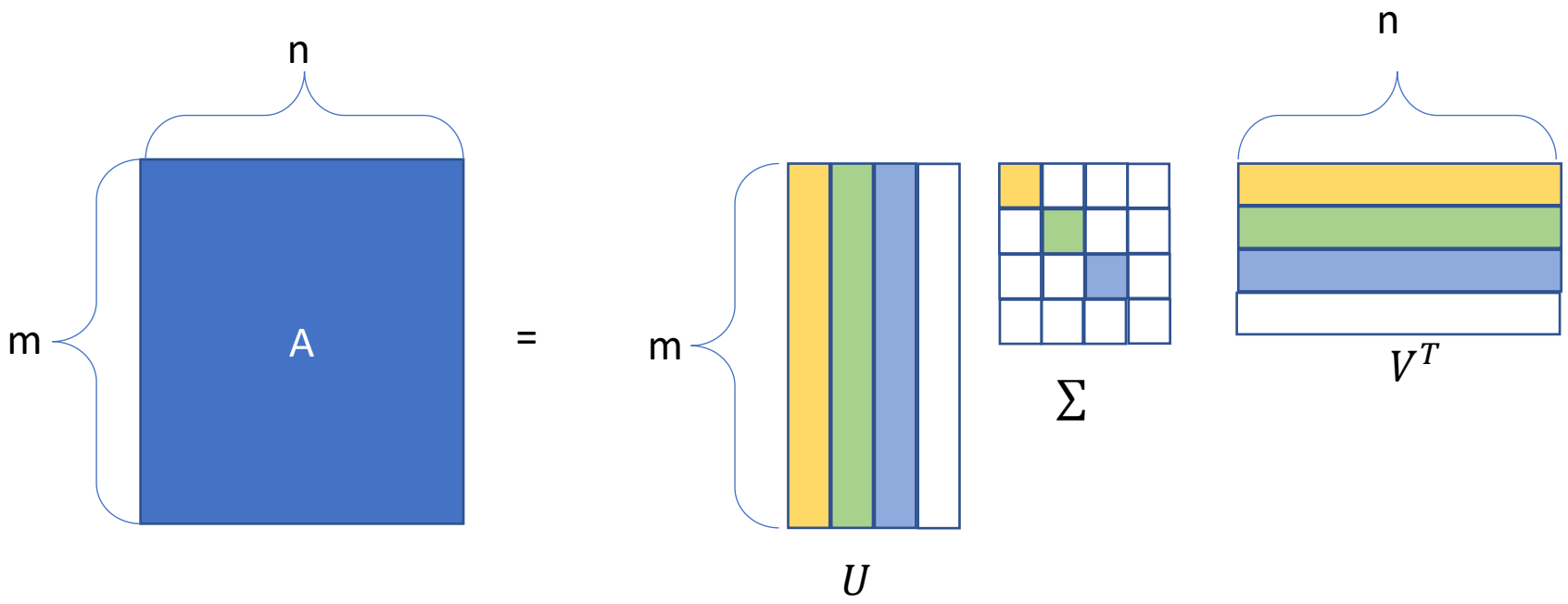
$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Solving $\boldsymbol{Ah} = 0$ How? SVD

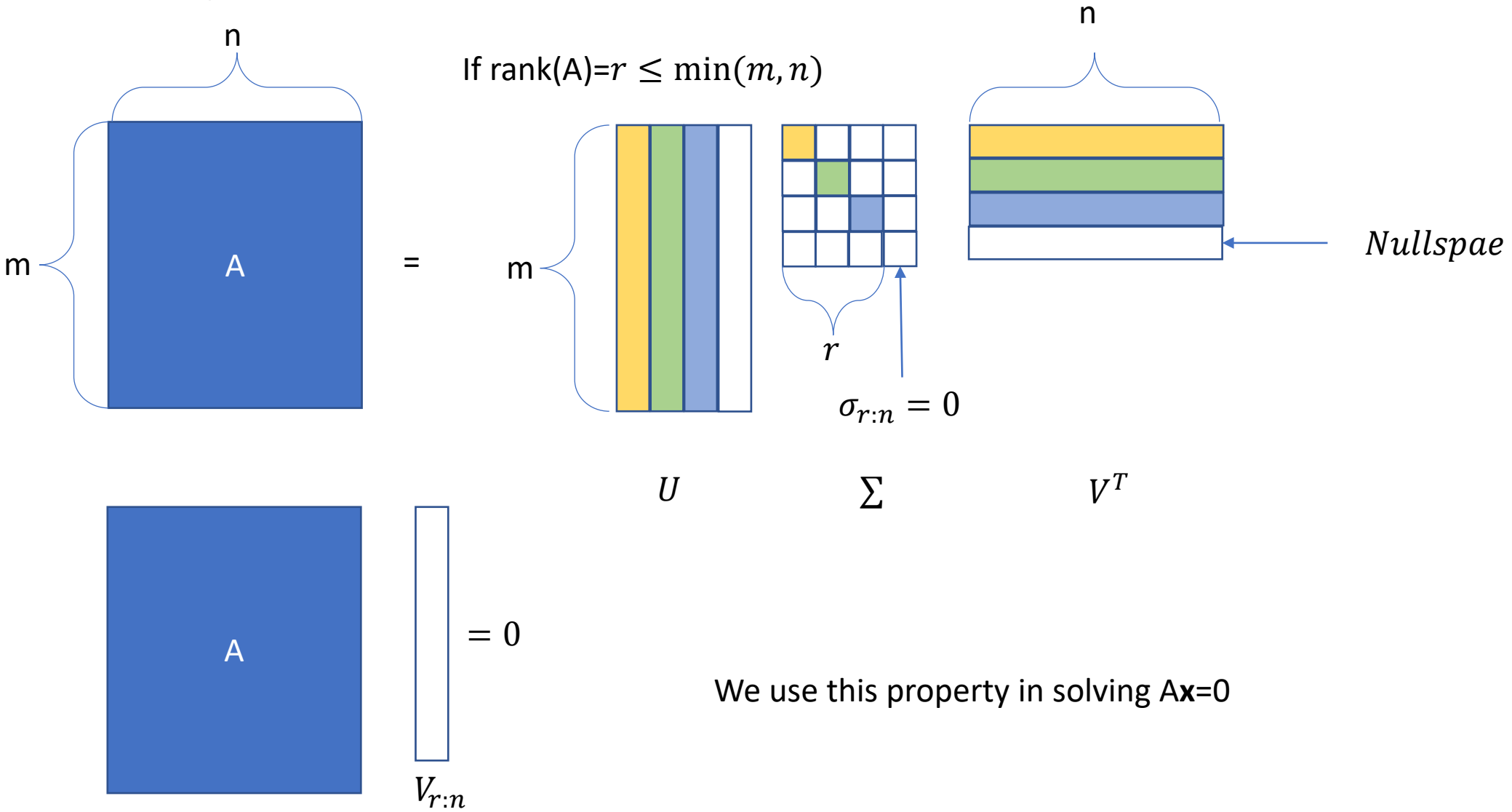
Singular Value Decomposition (SVD)

- Given an $m \times n$ matrix A, it can be decomposed into

$$A_{[m\times n]} = U_{[m\times k]} \sum_{[k\times k]} V^T{}_{[k\times n]} \qquad k = \mathrm{n}$$



Nullspace



We use this property in solving A**x**=0

Least Square Fit (Linear homogeneous Equations)

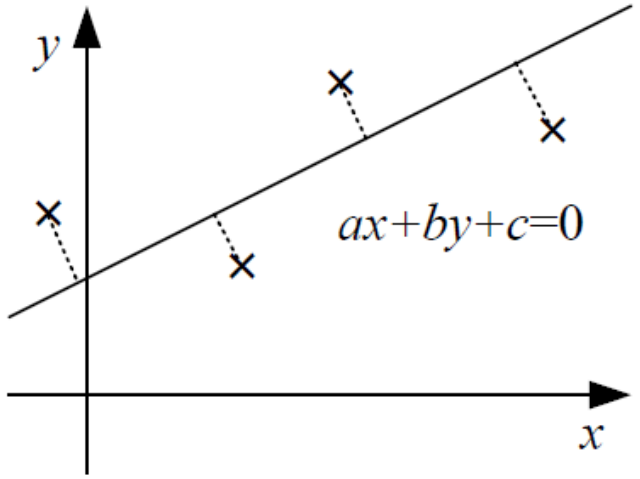
- Suppose we have a system of equations

$$\mathrm{Ax} = 0$$

- Matrix $\mathrm{A} \in \mathbb{C}^{m \times n}$ is known and we need to solve x
- As $\mathrm{Ax} \neq 0$, we aim to

$$\min_{\mathbf{x}} \|\mathrm{A}\mathbf{x}\|^2$$

- One trivial solution $\mathbf{x}$ =0, but we are not interested. $\|\mathbf{x}\|$ =1



Least Square Fit (Linear homogeneous Equations)

- We have

$$\min_{\mathbf{x}} \|A\mathbf{x}\|^2 \quad \text{and} \quad \|\mathbf{x}\| = \mathbf{1}$$

- Apply SVD on $A = U\Sigma V^T$, we have

$$\min_{\mathbf{x}} \|A\mathbf{x}\|^2 = \min_{\mathbf{x}} \left\|U\Sigma V^T\mathbf{x}\right\|^2$$

- Since U and V are **orthonormal**

$$\left\|U\Sigma V^T\mathbf{x}\right\|^2 = \left\|\Sigma V^T\mathbf{x}\right\|^2 \quad \text{and} \quad \left\|\Sigma V^T\right\| = \|\Sigma\|$$

Least Square Fit (Linear homogeneous Equations)

- Let $\mathbf{y} = \mathrm{V}^{\mathrm{T}}\mathbf{x}$, we have

$$\min_{\mathbf{x}}\|\mathrm{A}\mathbf{x}\|^2 = \min_{\mathbf{x}}\left\|\Sigma\mathrm{V}^{\mathrm{T}}\mathbf{x}\right\|^2 = \min_{\mathbf{y}}\|\Sigma\mathbf{y}\|^2 \text{ where } \|\mathbf{y}\| = \mathbf{1}$$

$$\Sigma\mathrm{y} = \begin{bmatrix} S_1 & 0 & \dots & 0 \\ \vdots & S_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & S_n \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{bmatrix} \text{ where } \|\mathbf{y}\| = \mathbf{1} \quad \text{Since } S_1 > S_2 > \cdots S_n, \mathbf{y} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{V}\mathbf{y} = [\mathrm{V}_1|\mathrm{V}_1|\dots|\mathrm{V}_\mathrm{n}] \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

- Finally we have $\mathbf{x}=\mathrm{V}_\mathrm{n}$

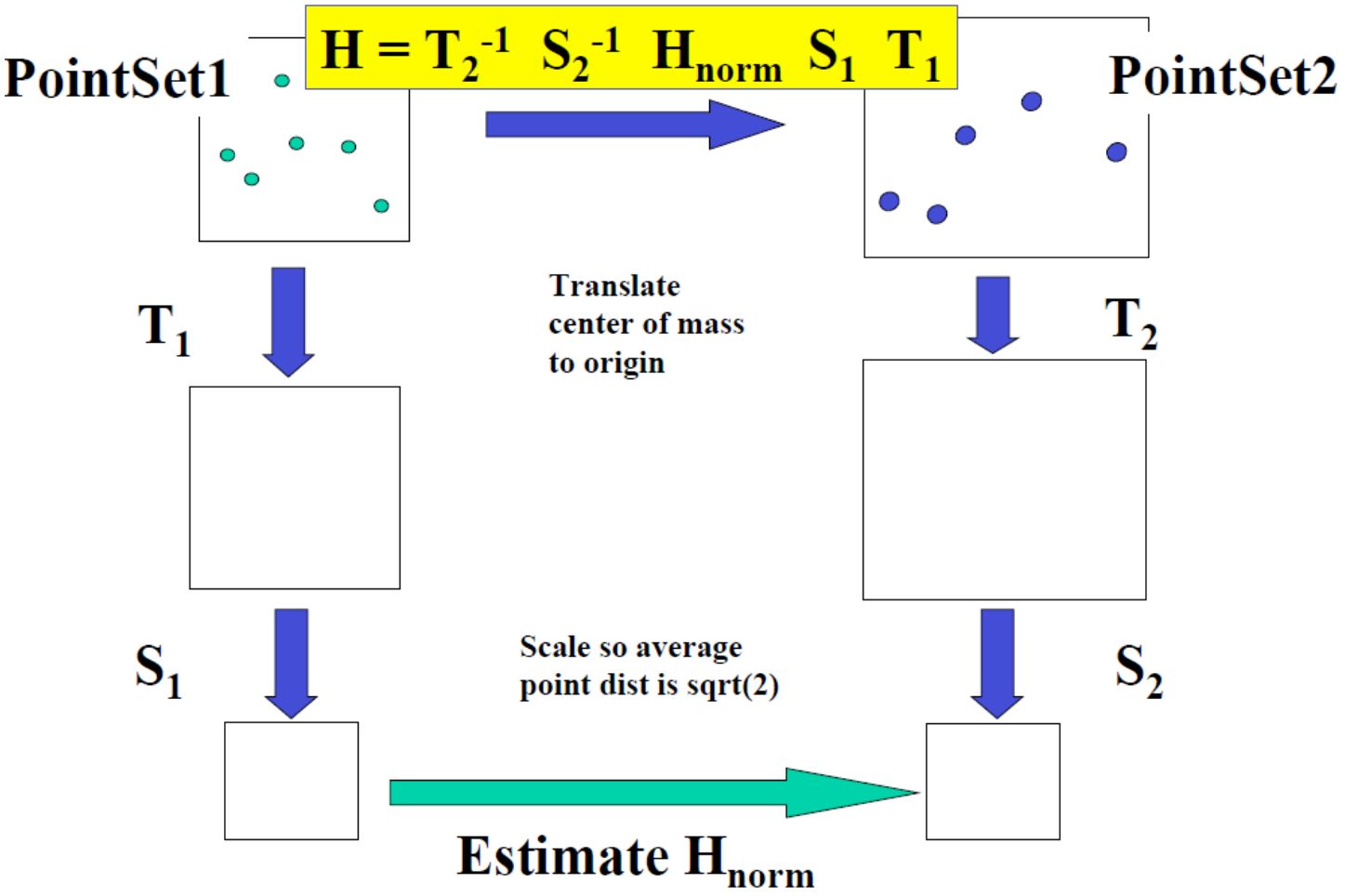
Caution: Numeric Conditioning

- R.Hartley: “In Defense of the Eight Point Algorithm”

- Observation: Linear estimation of projective transformation parameters from point correspondences often suffer from poor “conditioning” of the matrices involves. This means the solution is sensitive to noise in the points (even if there are no outliers).

- To get better answer, precondition the matrices by performing a normalization of each points
  - Translating center of mass to origin
  - Scaling so that average distance of points form origin is sqrt(2)
  - Repeat for every points in two image independently.

Pre-conditioning



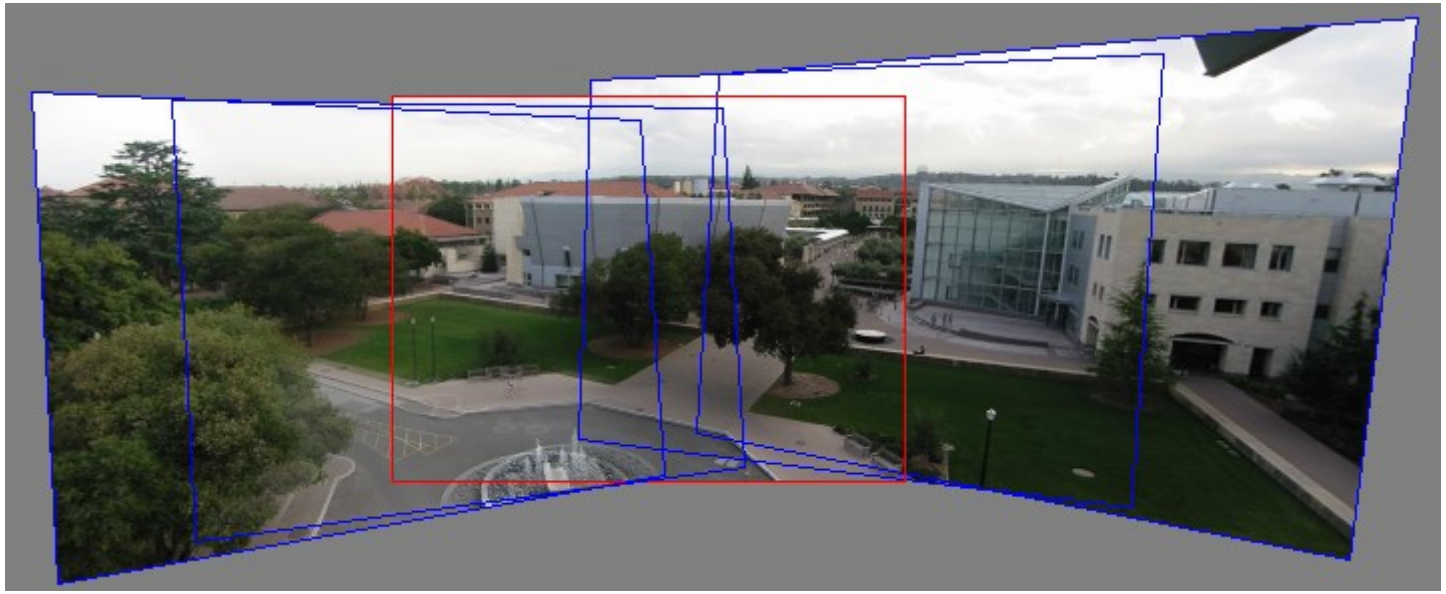
Slide Credit: Robert Collins

Steps to solve H by DLT

- Given: The correspondence $\boldsymbol{x}_i$ and $\boldsymbol{x}'_i$
- Aim: To estimate H such that $\boldsymbol{x}' = \mathrm{H}\boldsymbol{x}$

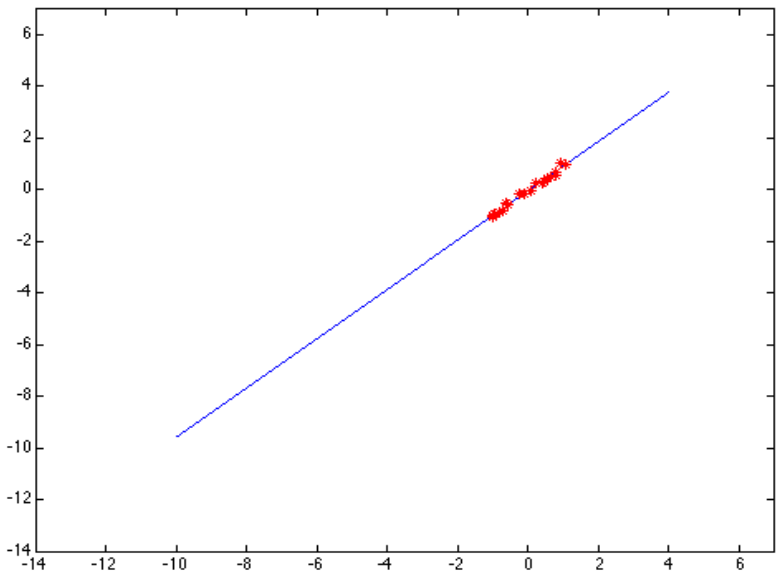
1 Normalize the points
2 For each correspondence, construct $\mathbf{A}_i$ (2x9)
3 Concatenate all points into $\mathbf{A}$ (2nx9)
4 Compute SVD of $\mathbf{A} = \mathrm{U}\Sigma\mathrm{V}^T$
5 The singular vector $\boldsymbol{v}_i$ with smallest singular value is the answer
6 Reshape $\boldsymbol{v}_i$ to 3x3 matrix which is H

Panoramas

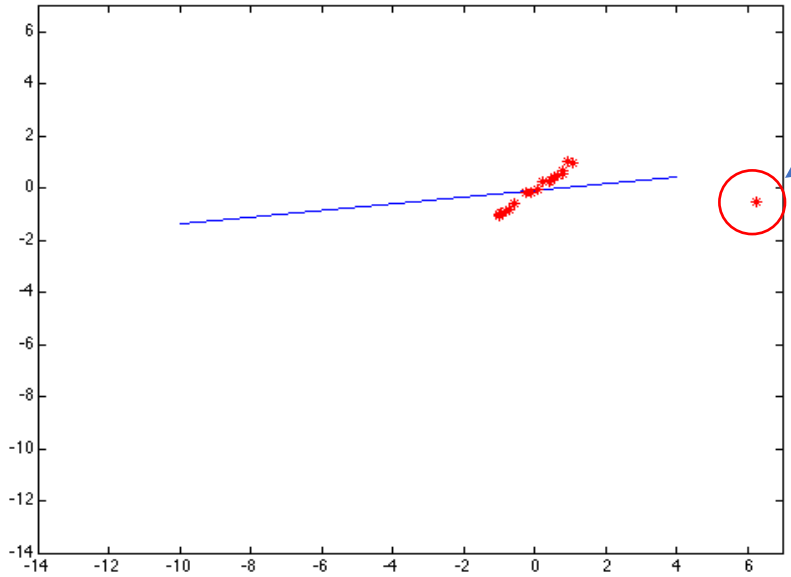


1. Pick one image (red)
2. Warp the other images towards it (usually, one by one)
3. blend

Robustness of Least Square



Good data



Present of Outliners

- Least Squares estimation is sensitive to outliers, present of a few outlier can greatly affect the result
- We need an estimation method which are robust to outliers.

How to take off outliners?

- Least square cannot due with outliners.
- We need to use RANSAC

Recall - RANdom SAmple Conensus (RANSAC)

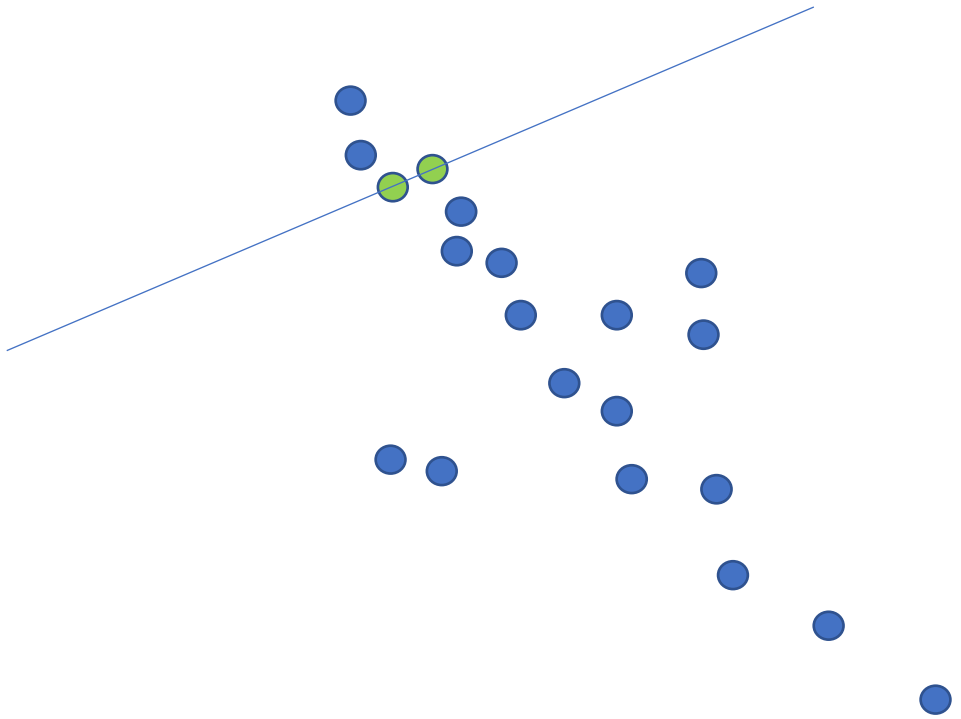
- RANSAC was proposed by Fischler & Bolles (1981)

- Processes:
  - Randomly select the sample
  - Fit for each sample
  - Classify the data points as two types Outliers and Inliers
  - Vote for the best model.

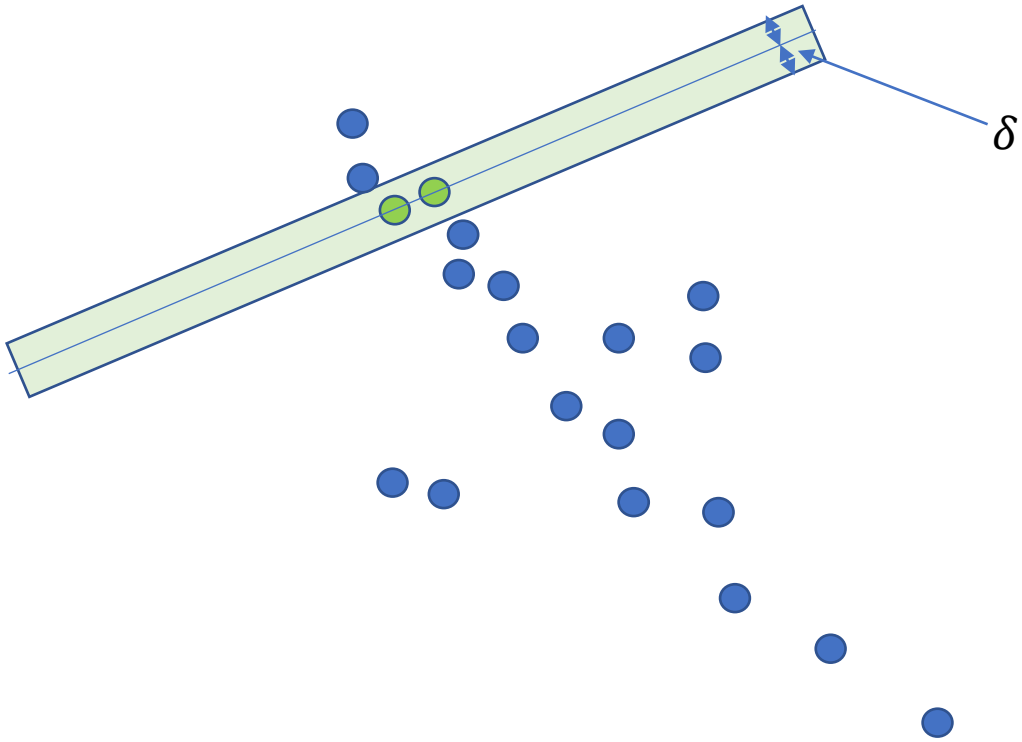
- Assumptions:
  - We have “fewer” outliers than inliers
  - We have “fewer” missing data

Source: M.A. Fischler and R.C. Bolles(1981). “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography”. Comm. Of the ACM 24:381-395

RANSAC Procedure



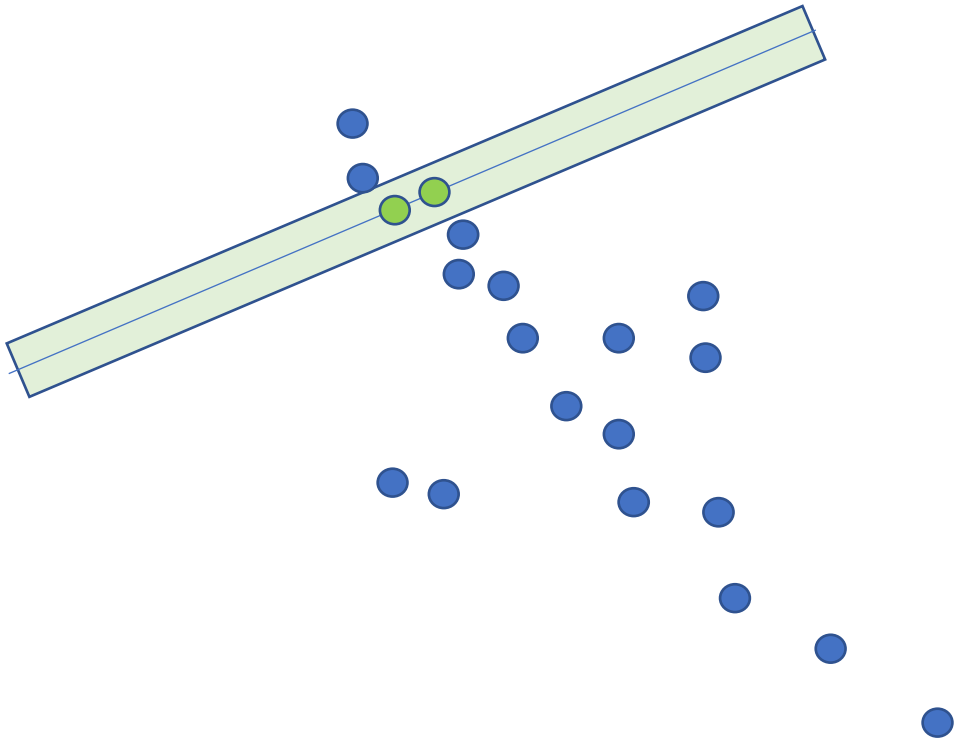
Step 1: Two points are selected randomly.
Step 2: Define a line with selected points



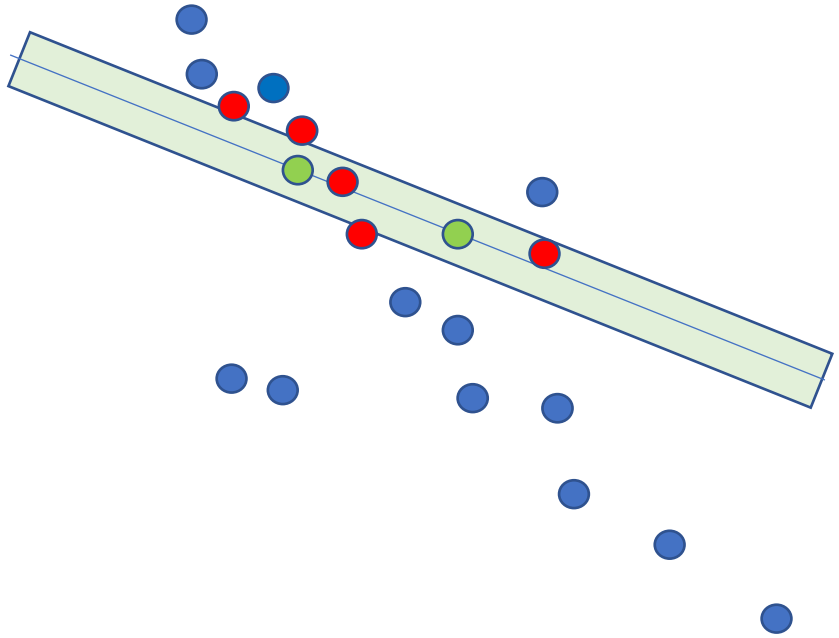
Step 3: Define a threshold distance δ

Step 4: Count number of points within δ

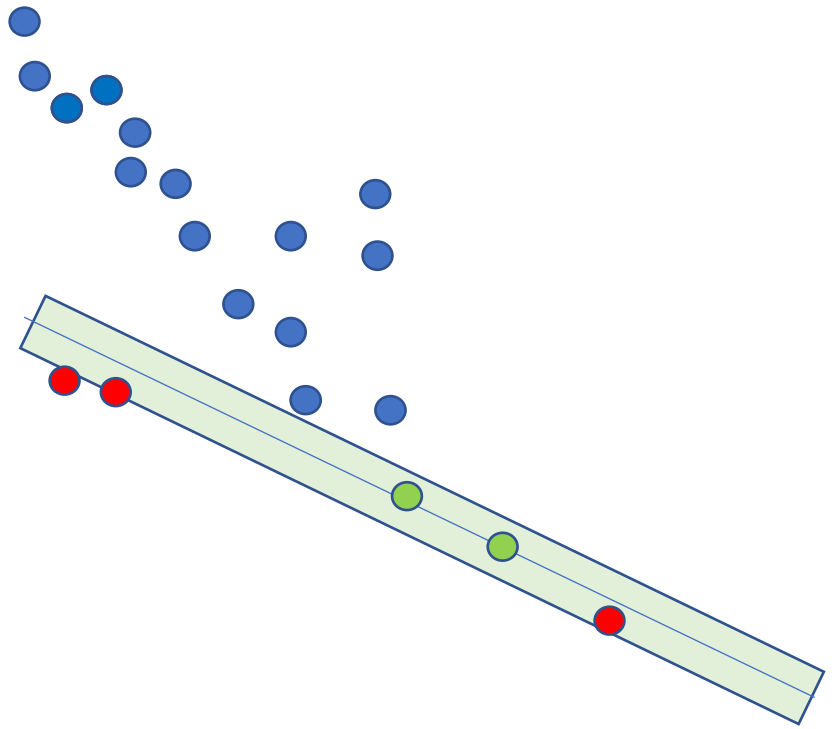
Count=2



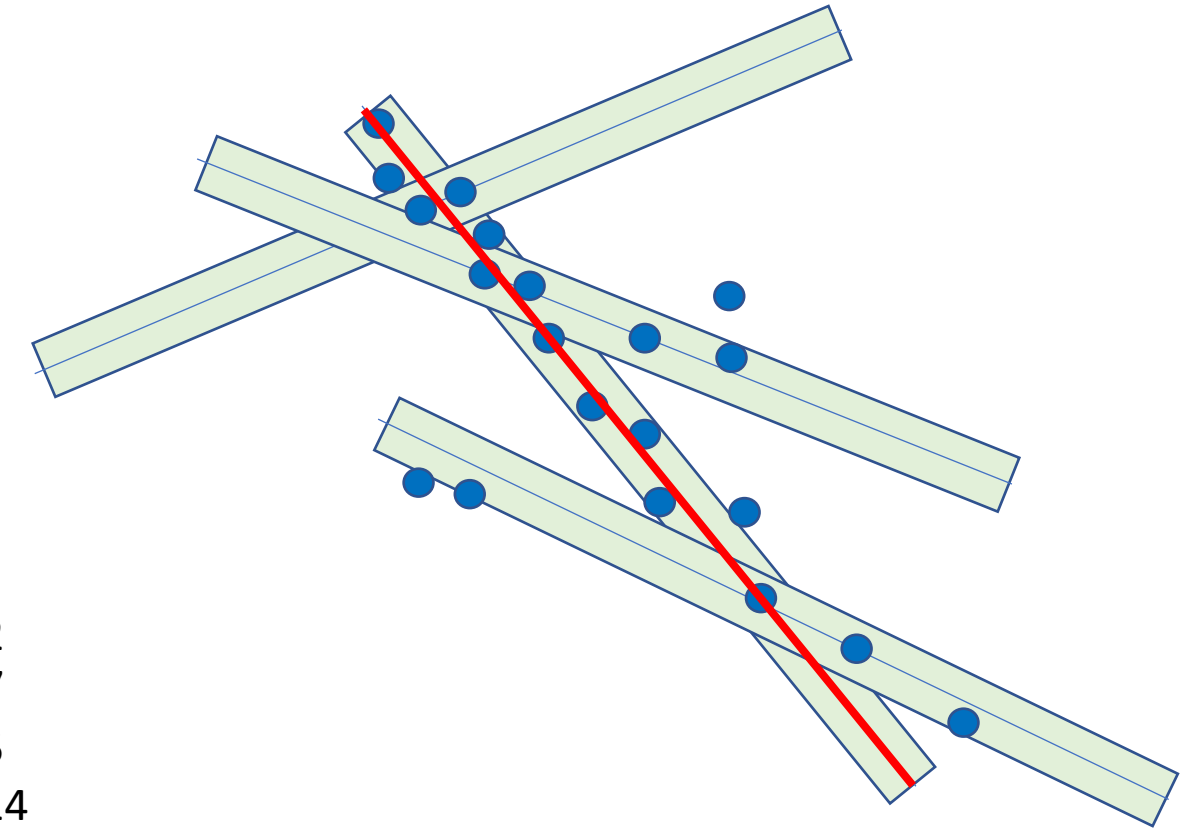
Count=2



Count=7



Count=5



Count=2
Count=7
Count=5
Count=14

How Many Samples to Choose?

- Try every possible sample?
  - Computationally infeasible and unnecessary
- How many samples do we need?
- Let
  - e=probability that a point is an outlier
  - s=number of points in a sample
  - N=number of samples we need
  - p=desired probability that we get a good sample
- Solve the following for N:

$$1 - (1 - (1 - \mathrm{e})^{\mathrm{s}})^{\mathrm{N}} = \mathrm{p}$$

Number of samples to choose

Let

- e=probability that a point is an outlier
- s=number of points in a sample
- N=number of samples we need
- p=desired probability that we get a good sample.

$$1 - (1 - (1 - e)^s)^N = p$$

Probability of selecting one point is an inlier

Number of samples to choose

Let

- e=probability that a point is an outlier
- s=number of points in a sample
- N=number of samples we need
- p=desired probability that we get a good sample.

$$1 - (1 - \underbrace{(1 - e)^s}_{\text{Probability of selecting s point is an inlier (sample only contains inliers)}})^N = p$$

Number of samples to choose

Let

- e=probability that a point is an outlier
- s=number of points in a sample
- N=number of samples we need
- p=desired probability that we get a good sample.

$$1 - (1 - (1 - e)^{s})^{N} = p$$

Probability of selecting s points that one or more points were outliers
(Samples is contaminated)

Number of samples to choose

Let

- e=probability that a point is an outlier
- s=number of points in a sample
- N=number of samples we need
- p=desired probability that we get a good sample.

$$1 - \underbrace{(1 - (1 - e)^s)^N}_{\text{Probability of selecting N samples that one or more points were outliers (contaminated)}} = p$$

Number of samples to choose

Let

- e=probability that a point is an outlier
- s=number of points in a sample
- N=number of samples we need
- p=desired probability that we get a good sample.

$$1 - (1 - (1 - e)^s)^N = p$$

Probability at least one sample was not contaminated
(At least one sample is composed of inliers only)

How many Samples?

For N selection, at least one random sample is free from outliers is given by:

$$1 - (1 - (1 - \mathrm{e})^{\mathrm{s}})^{\mathrm{N}} = \mathrm{p}$$

- Then we have

$$N = \frac{\log(1 - \mathrm{p})}{\log(1 - (1 - \mathrm{e})^{\mathrm{s}})}$$

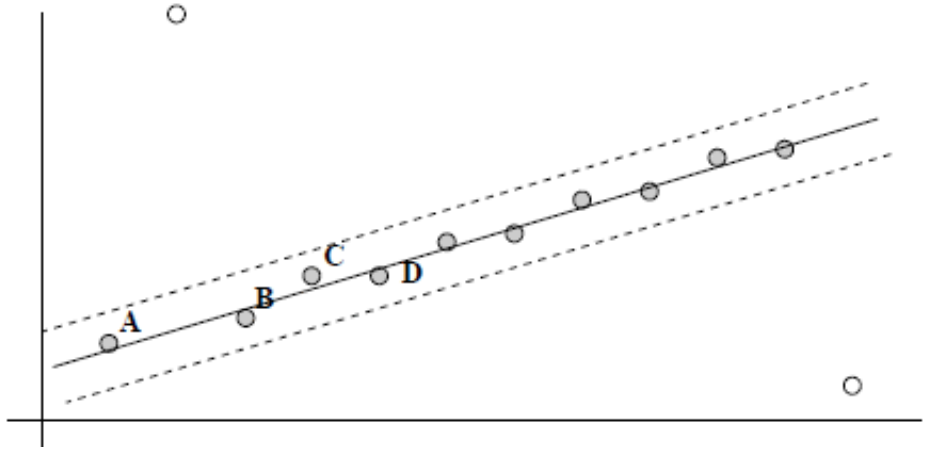
- For p=0.99, N is given

Sample size	Proportion of outliers ϵ						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

Source: R Hartley, A. Zisserman

Example: Number of sample for line fitting

- Number of sample n=12
- Minimal Sample Size s=2
- 2 outliers: e=2/12 = 20%
- For 99% chance of getting a pure-inlier sample is N=5



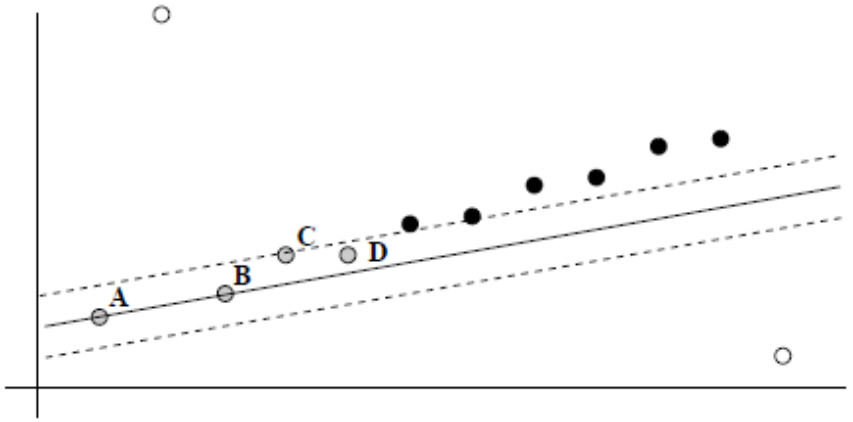
Sample size	Proportion of outliers ϵ						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

- For all trial Number of trial $= {}_{12}C_2 = \frac{12!}{2!(10!)}$
- =66

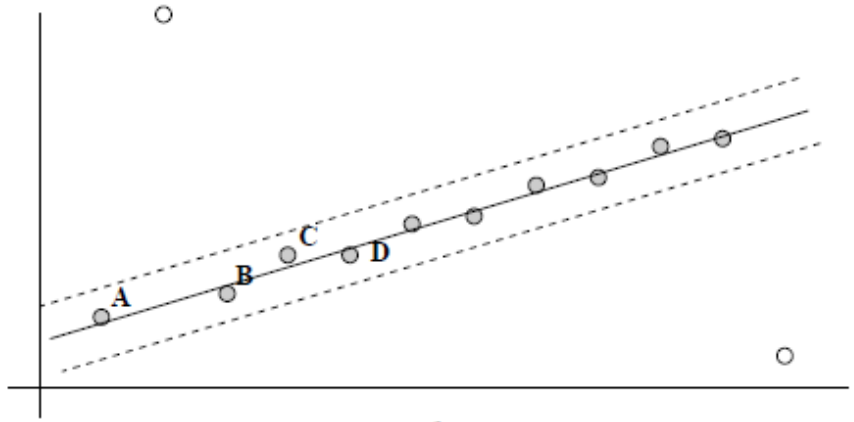
Source: R Hartley, A. Zisserman

Final Step

- Re-estimate the model using all the inliers by least square fit

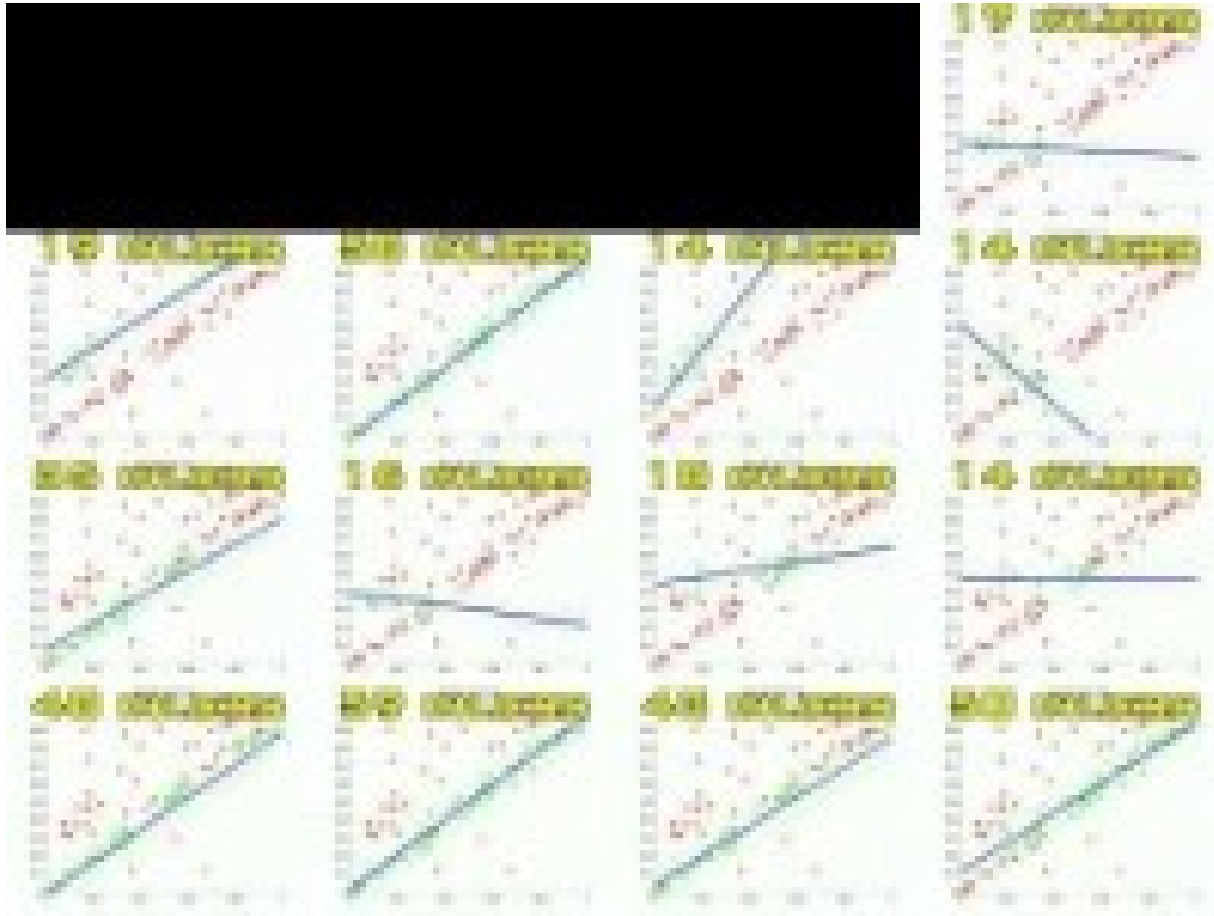


a



b

Ransac Song



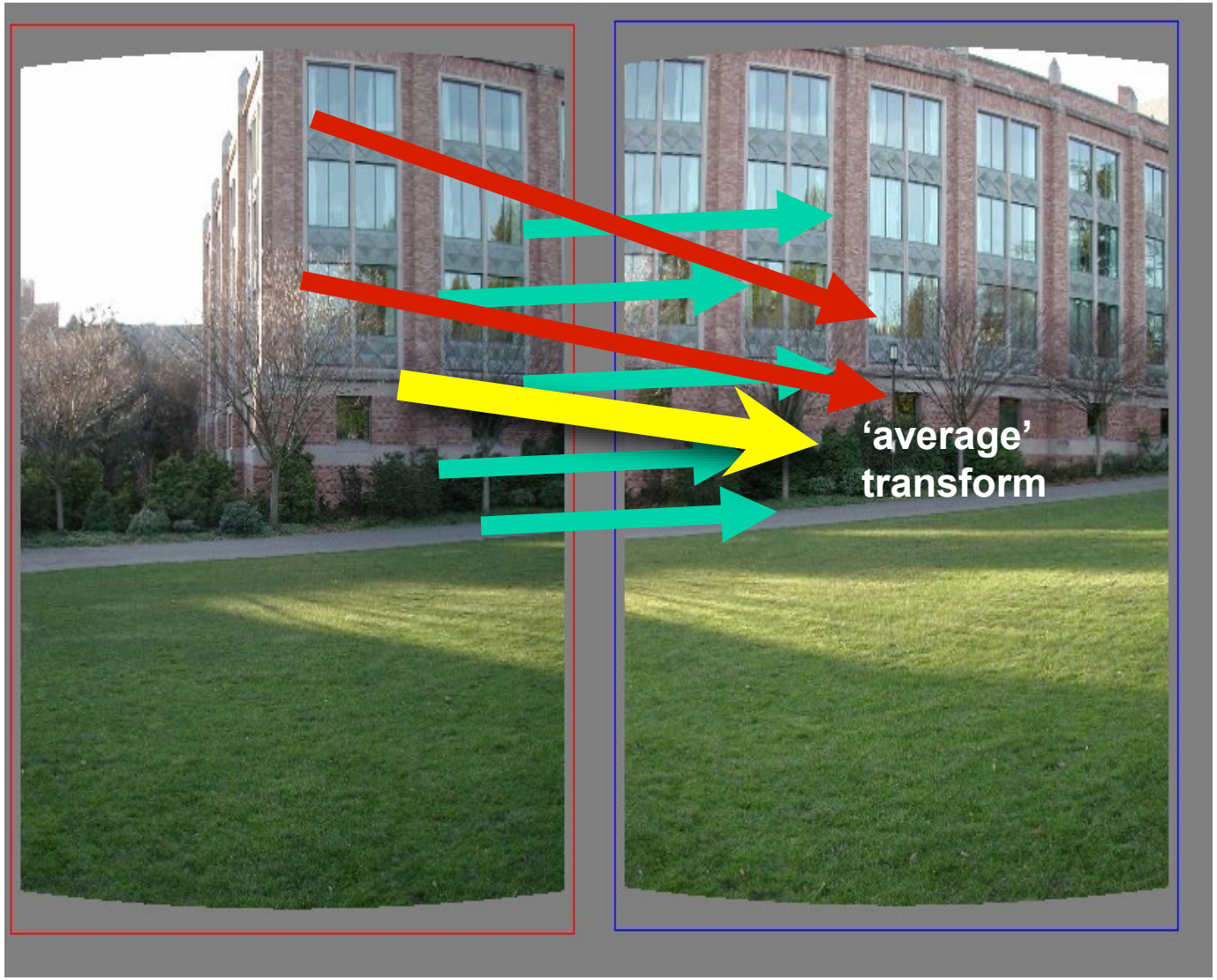
Given two images…

find matching features (e.g., SIFT) and a translation transform

Matched points will usually contain bad correspondences

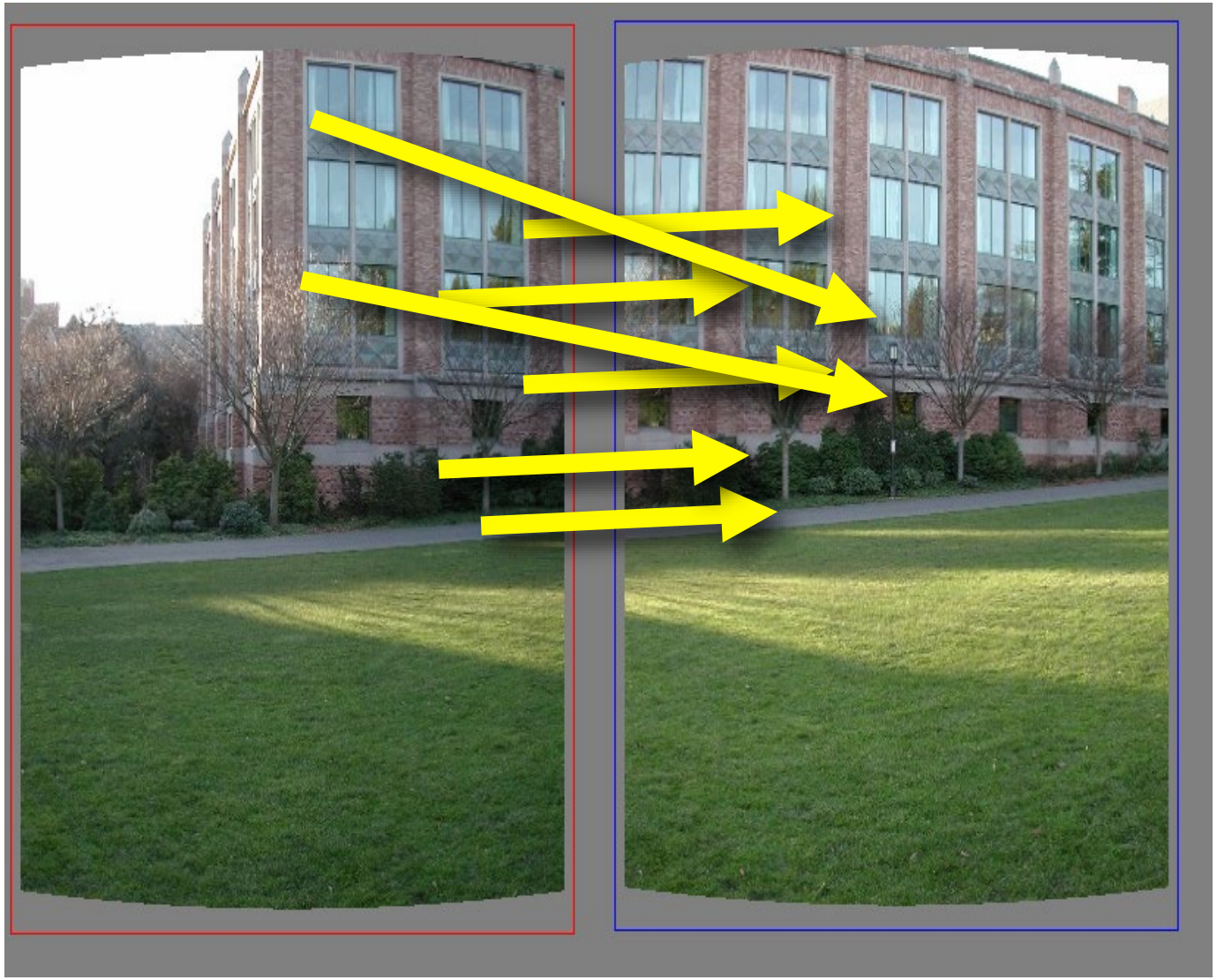
how should we estimate the transform?

LLS will find the ‘average’ transform

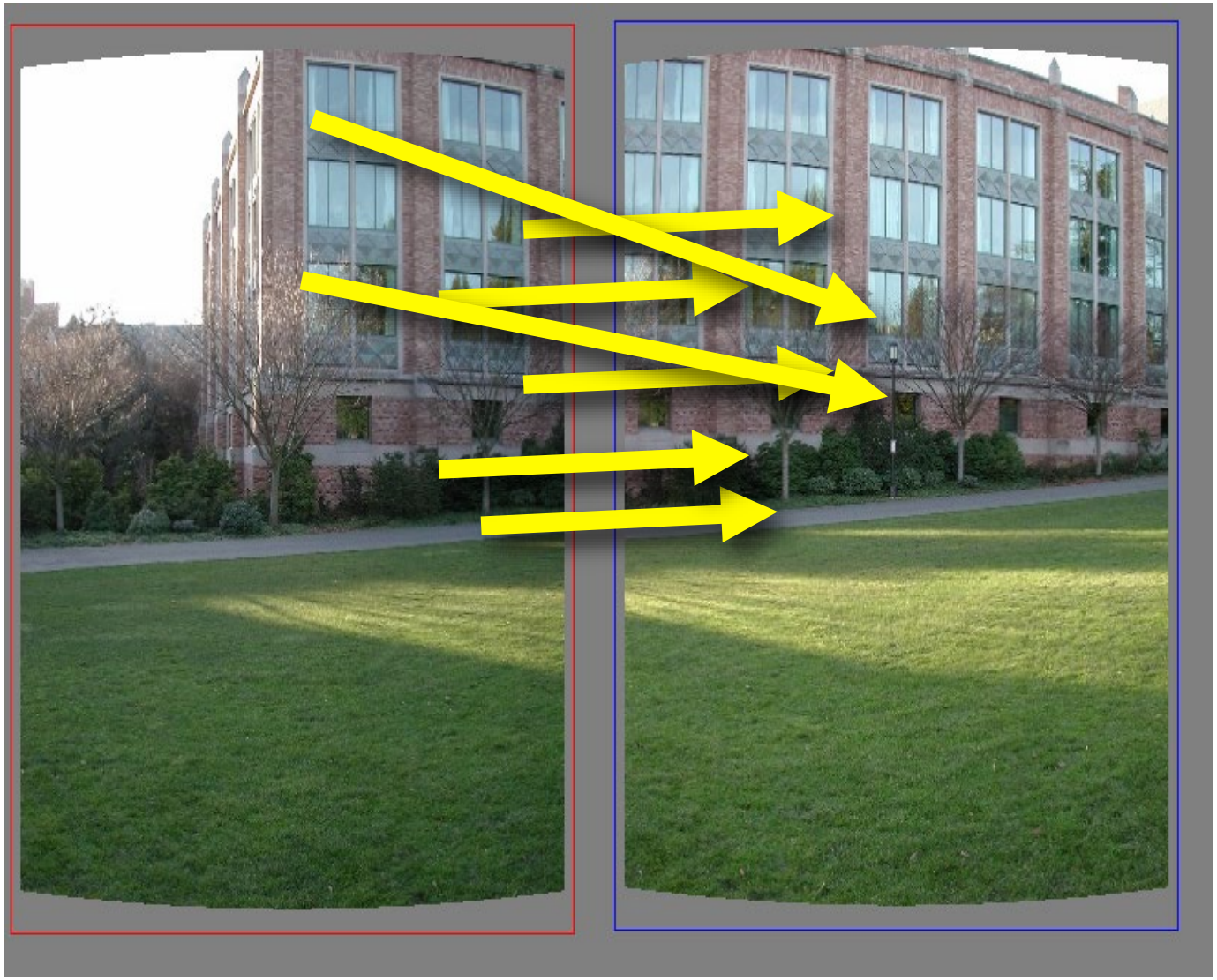


solution is corrupted by bad correspondences

Use RANSAC

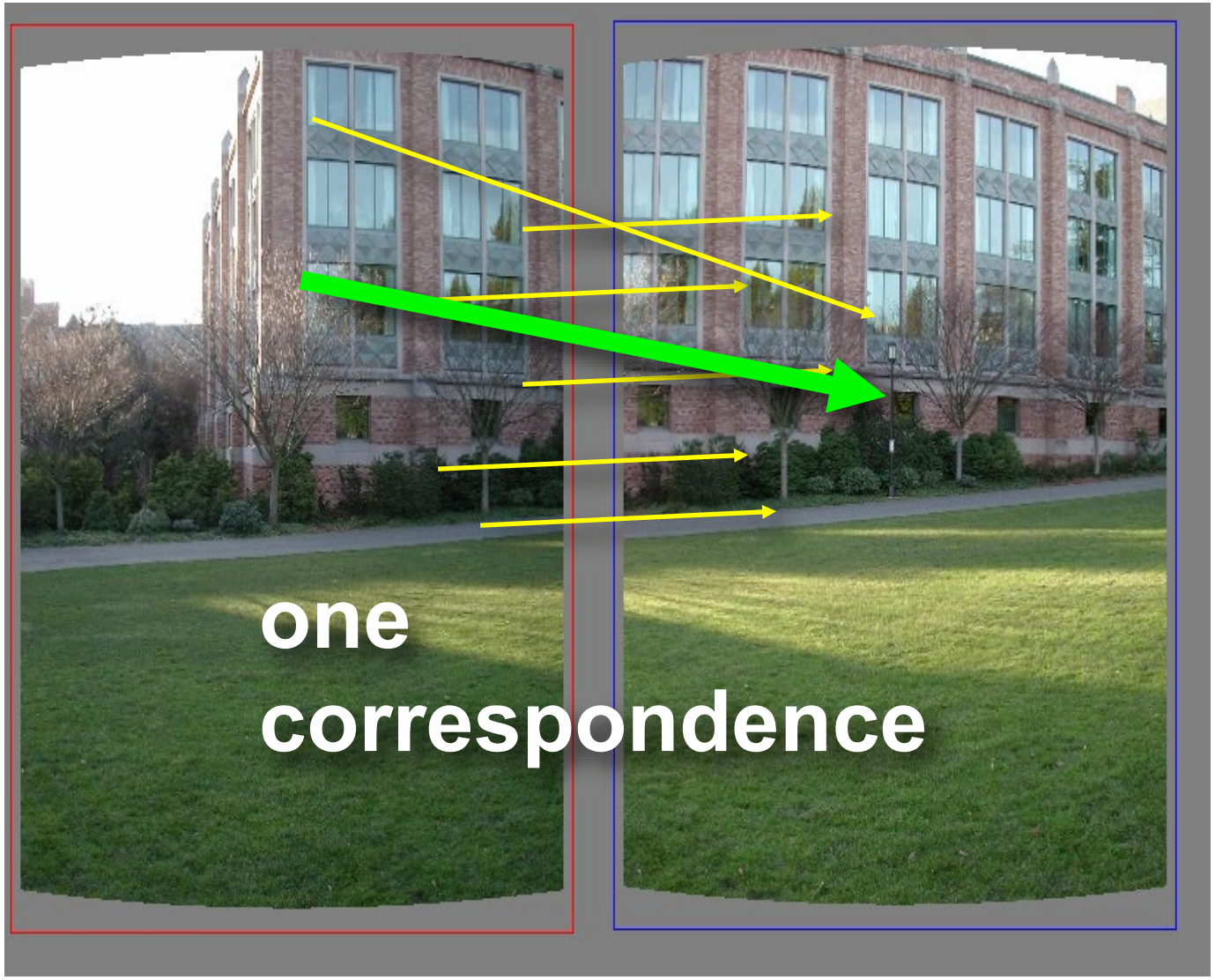


How many correspondences to compute translation transform?

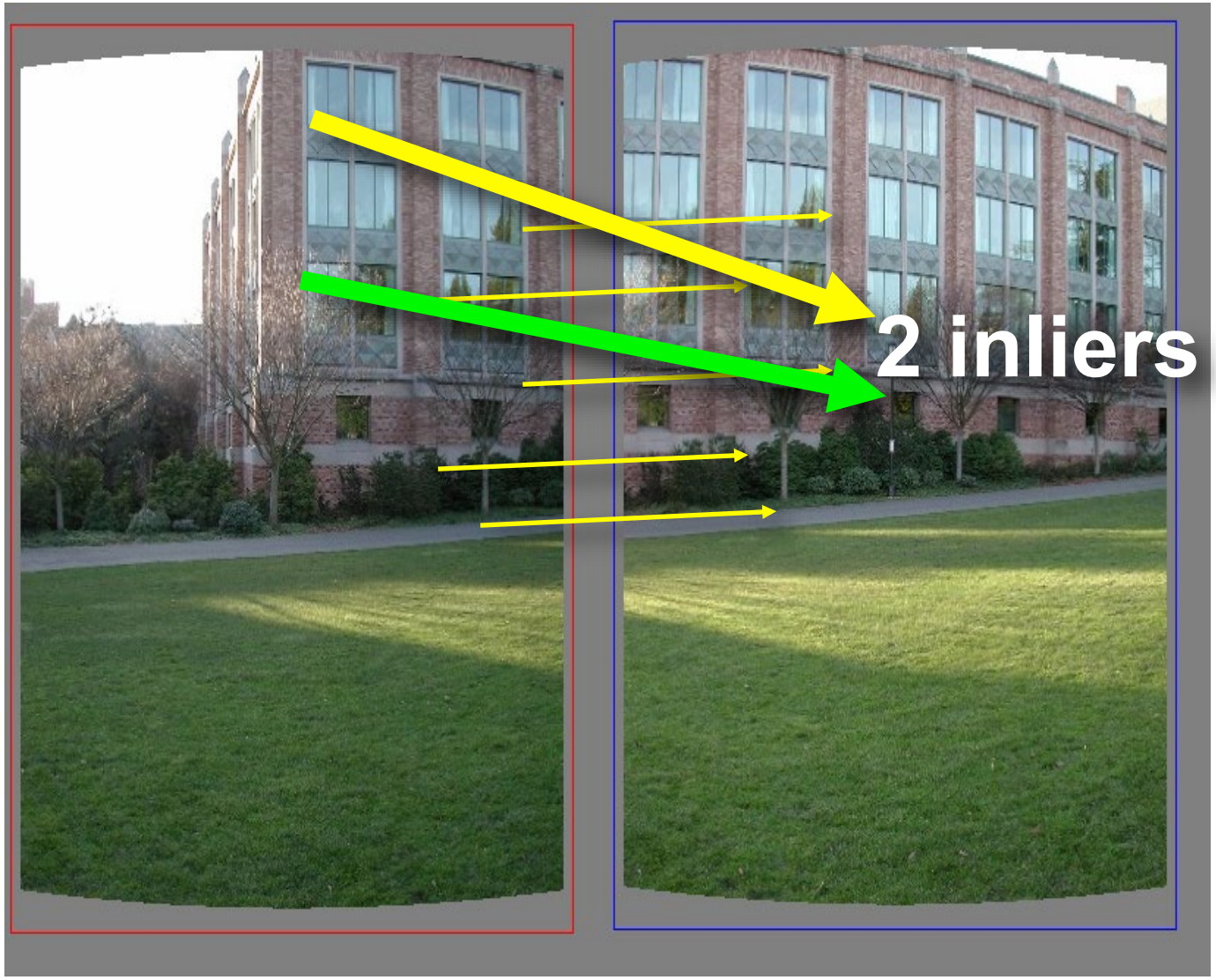


Need only **one correspondence**, to find translation model

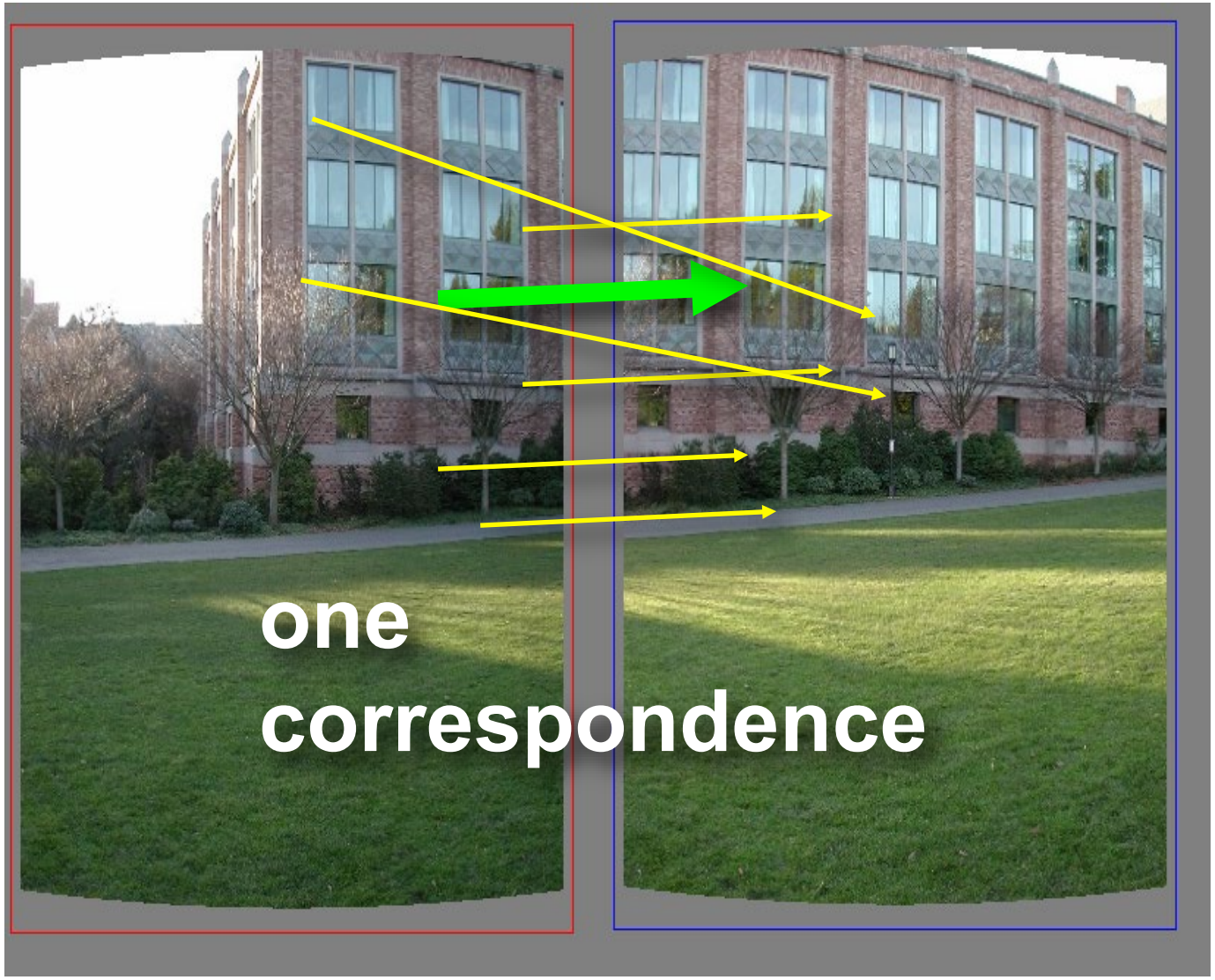
Pick one correspondence, count inliers



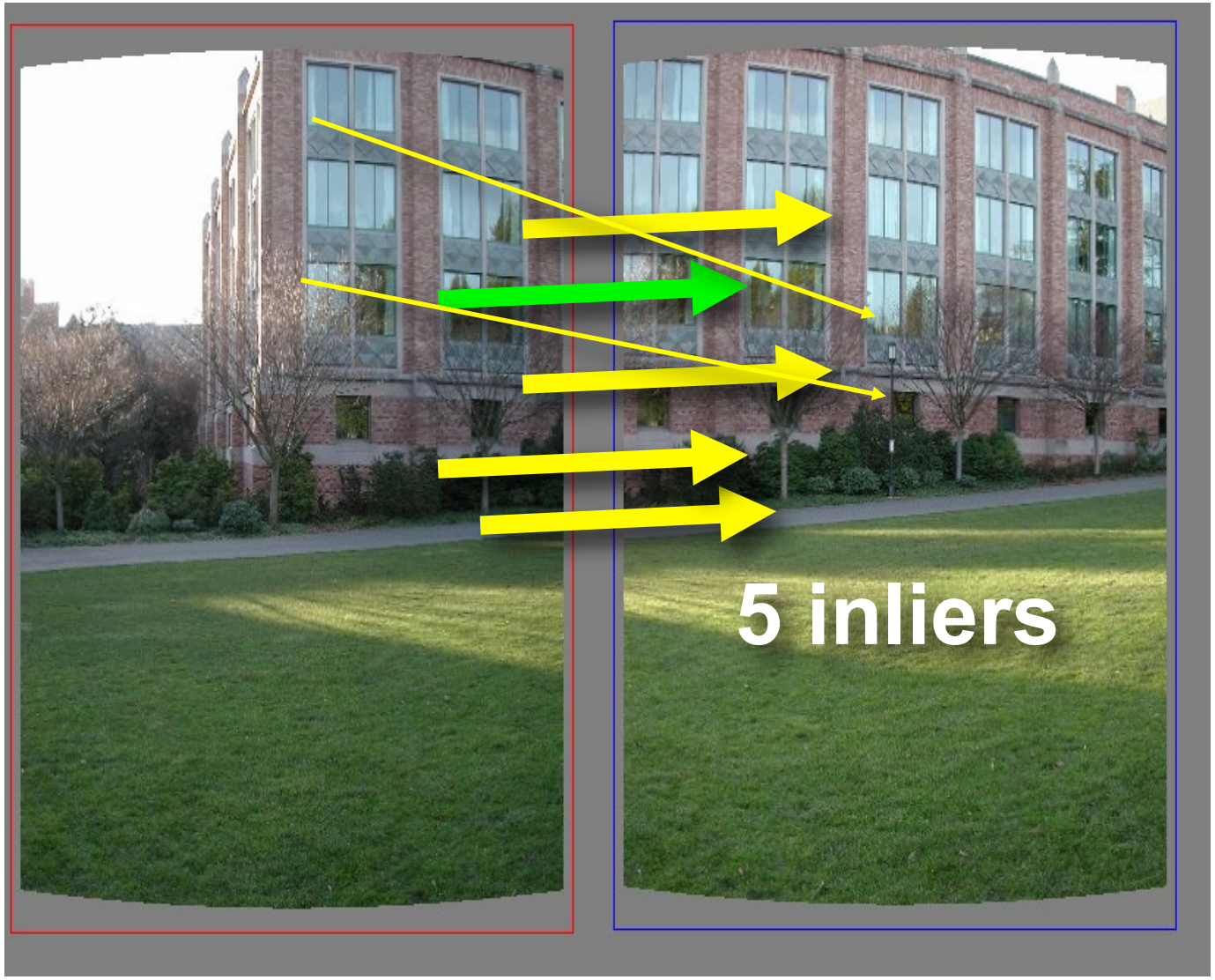
Pick one correspondence, count inliers



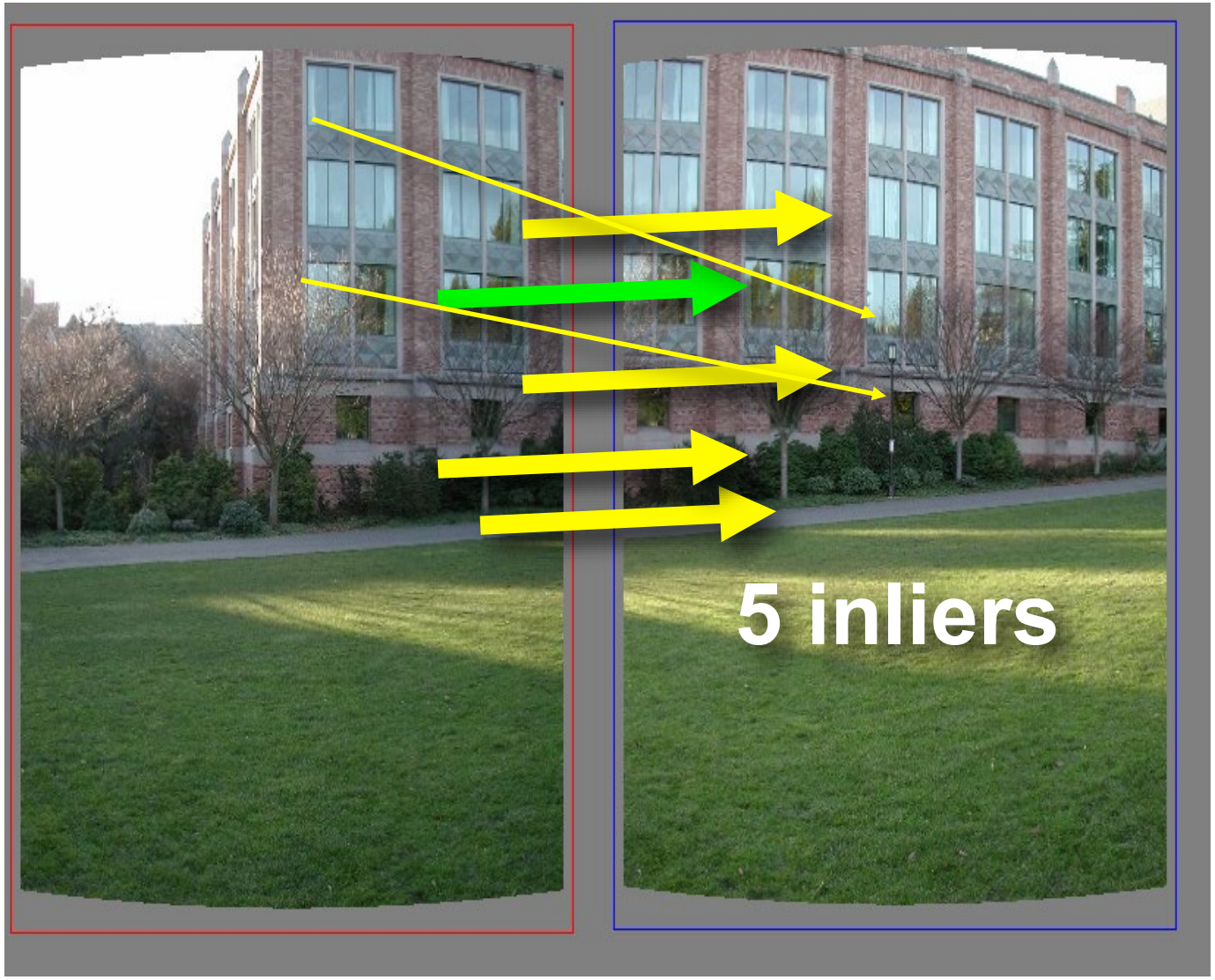
Pick one correspondence, count inliers



Pick one correspondence, count inliers



Pick one correspondence, count inliers



Pick the model with the highest number of inliers!

Example

Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Result

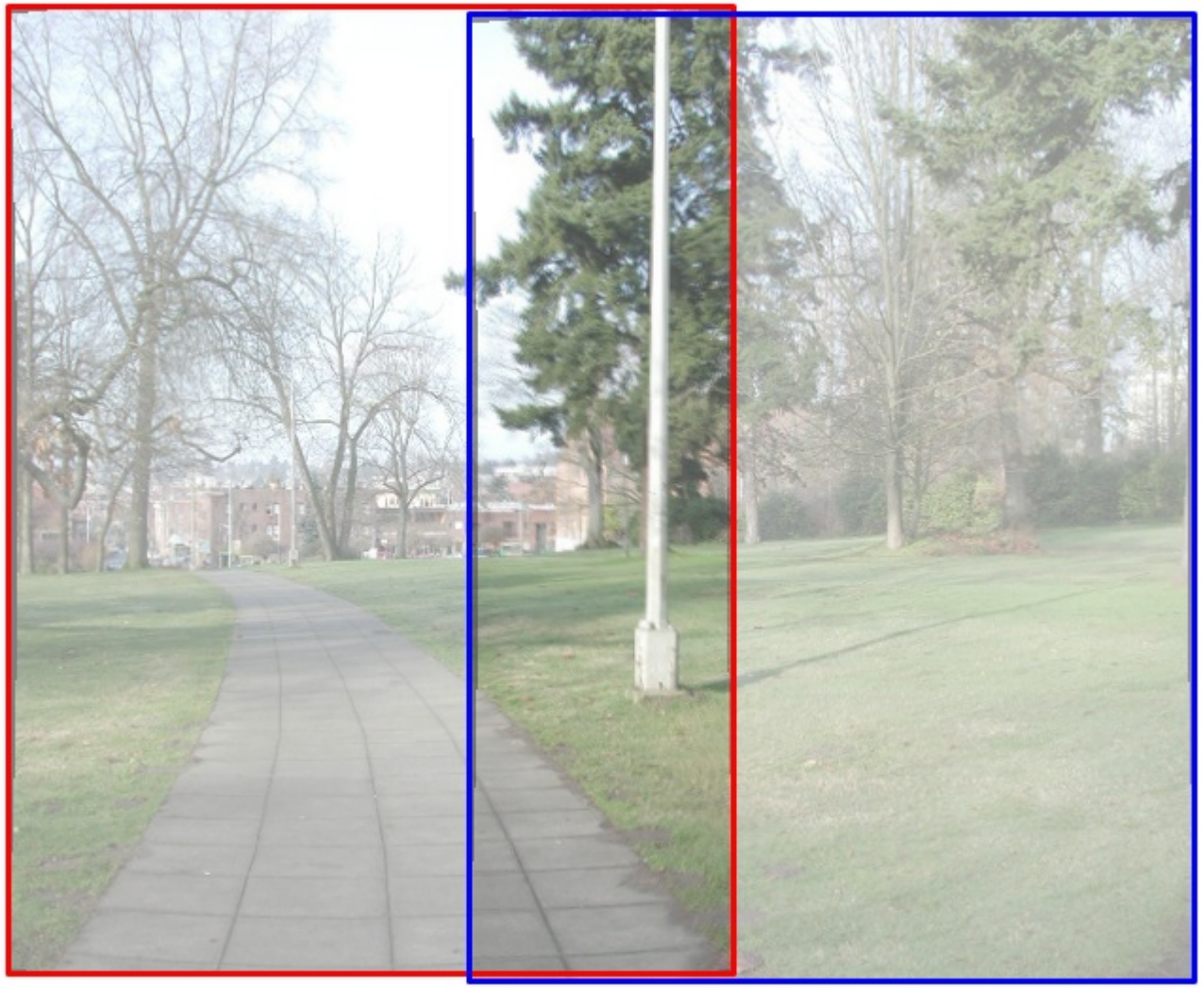
Image Credit: R. Hartley and A. Zisserman, “Multiple View Geometry in Computer Vision”

Image Blending

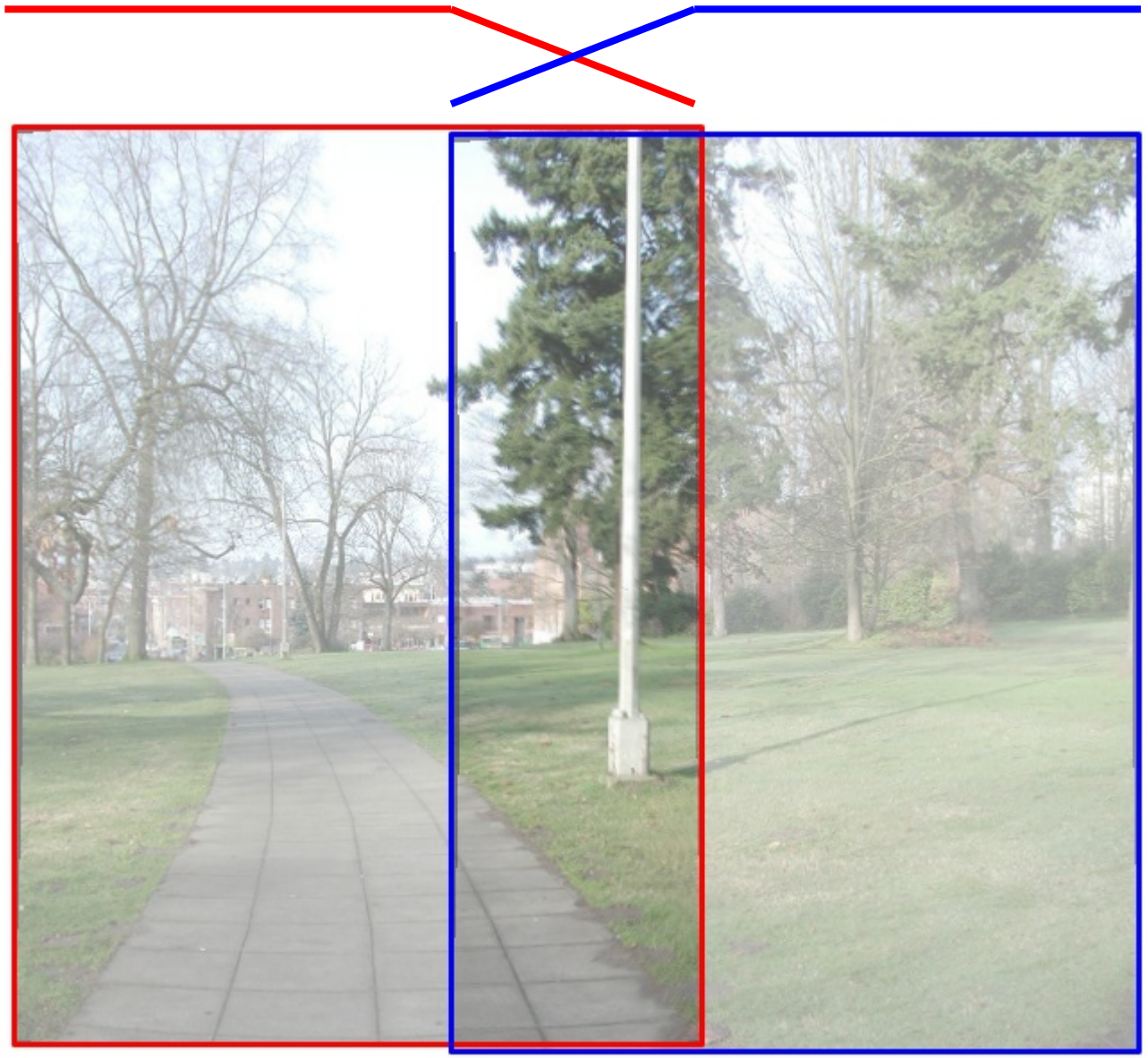
Blending

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

Blending

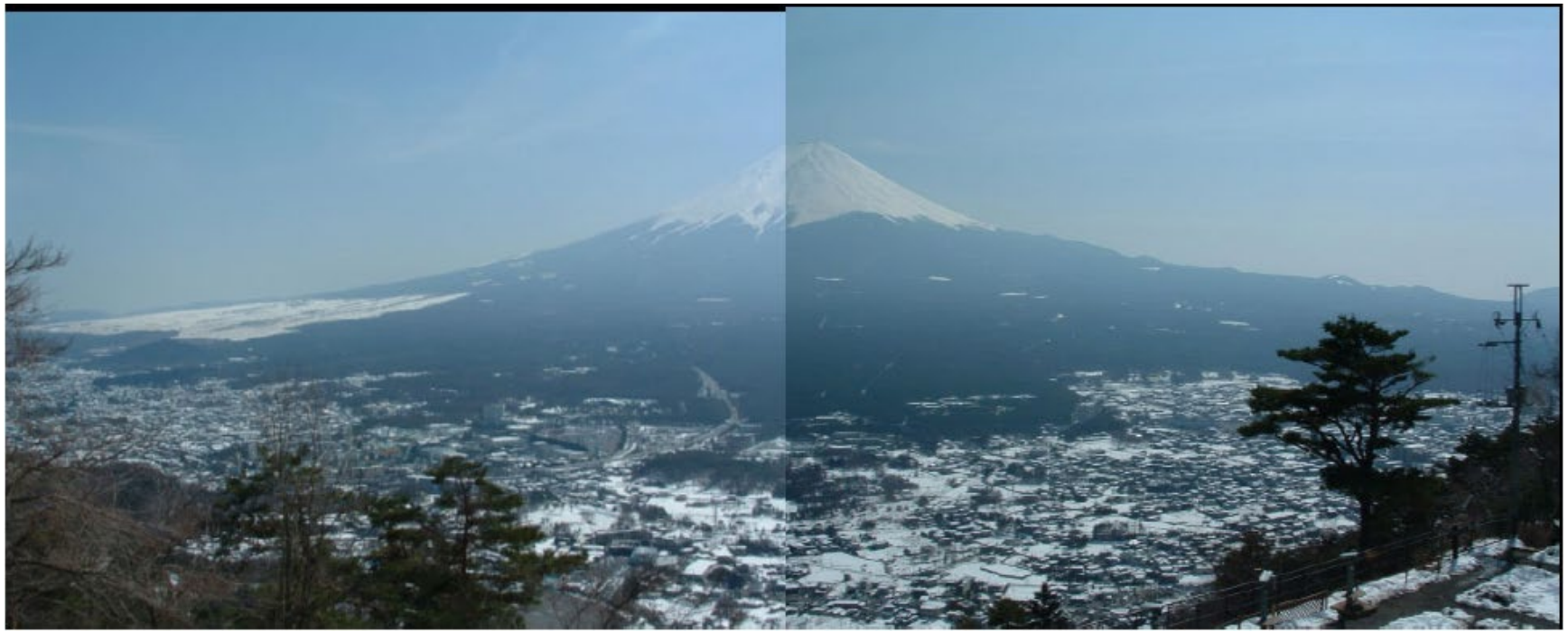


Blending



Blending

Gradient-domain stitching



Gradient-domain stitching

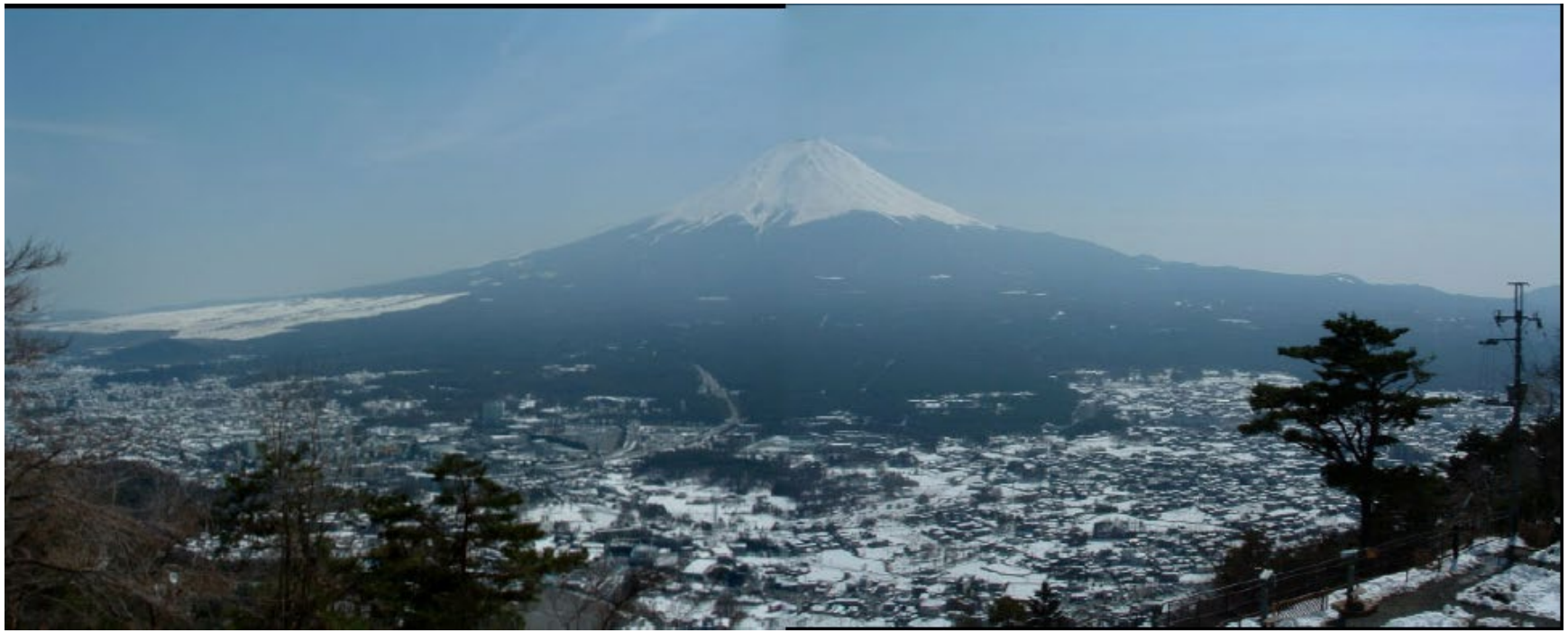
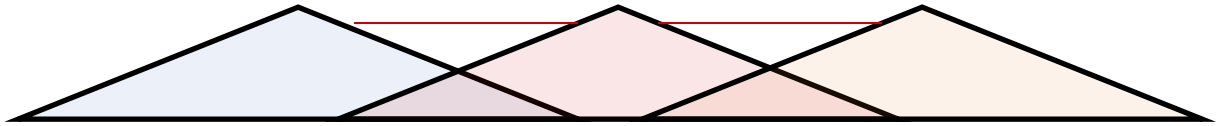


Image feathering

- Weight each image proportional to its distance from the edge (distance map [Danielsson, CVGIP 1980]



- 1. Generate *weight map* for each image
- 2. Sum up all of the weights and divide by sum: weights sum up to 1: $w_i' = w_i / (\sum_i w_i)$

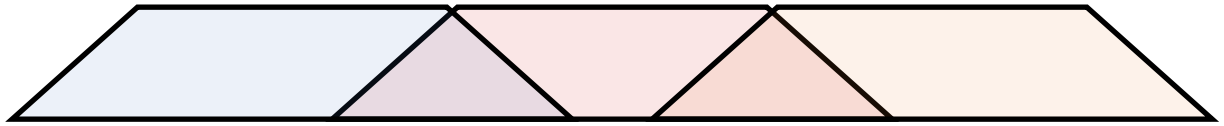
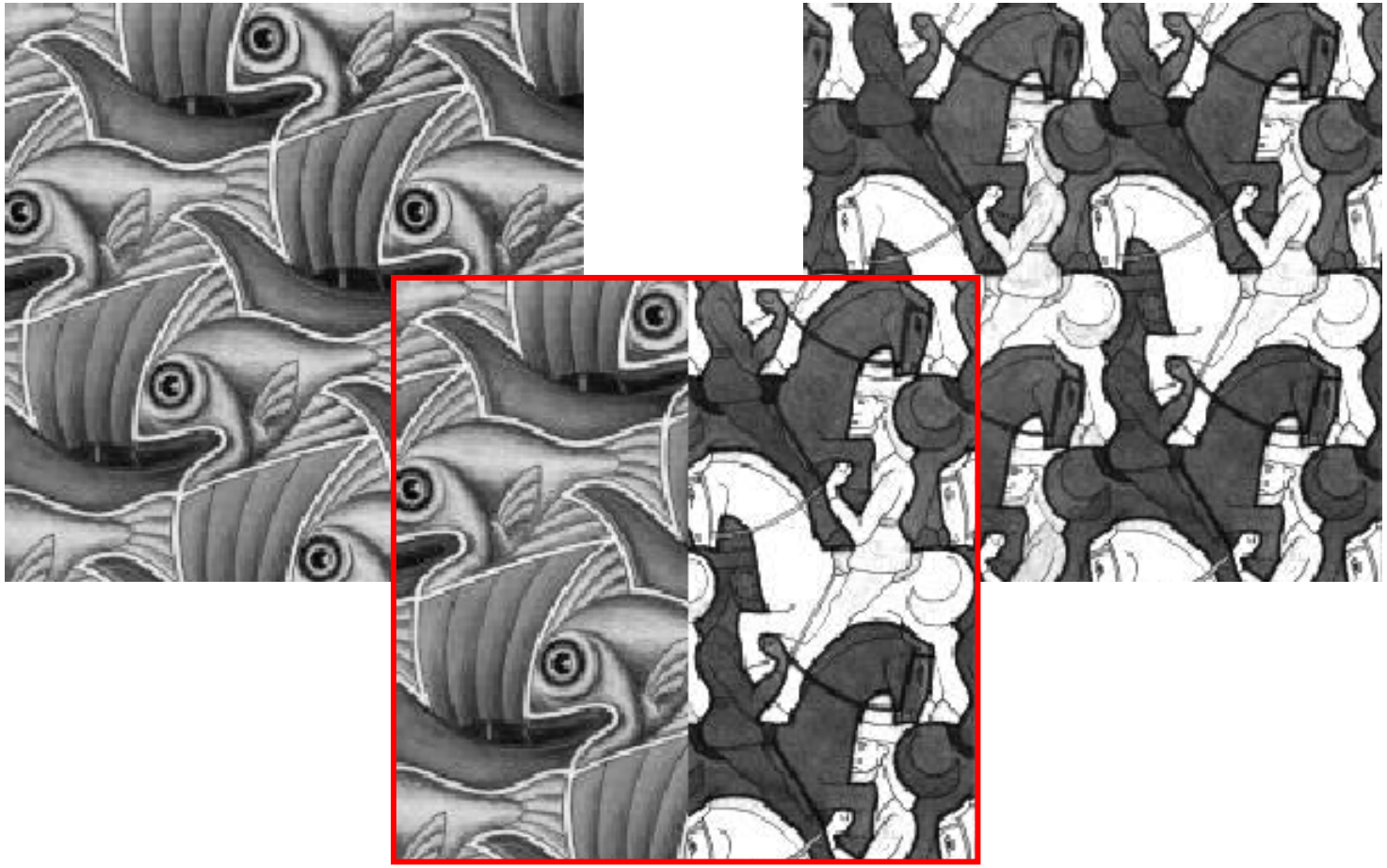
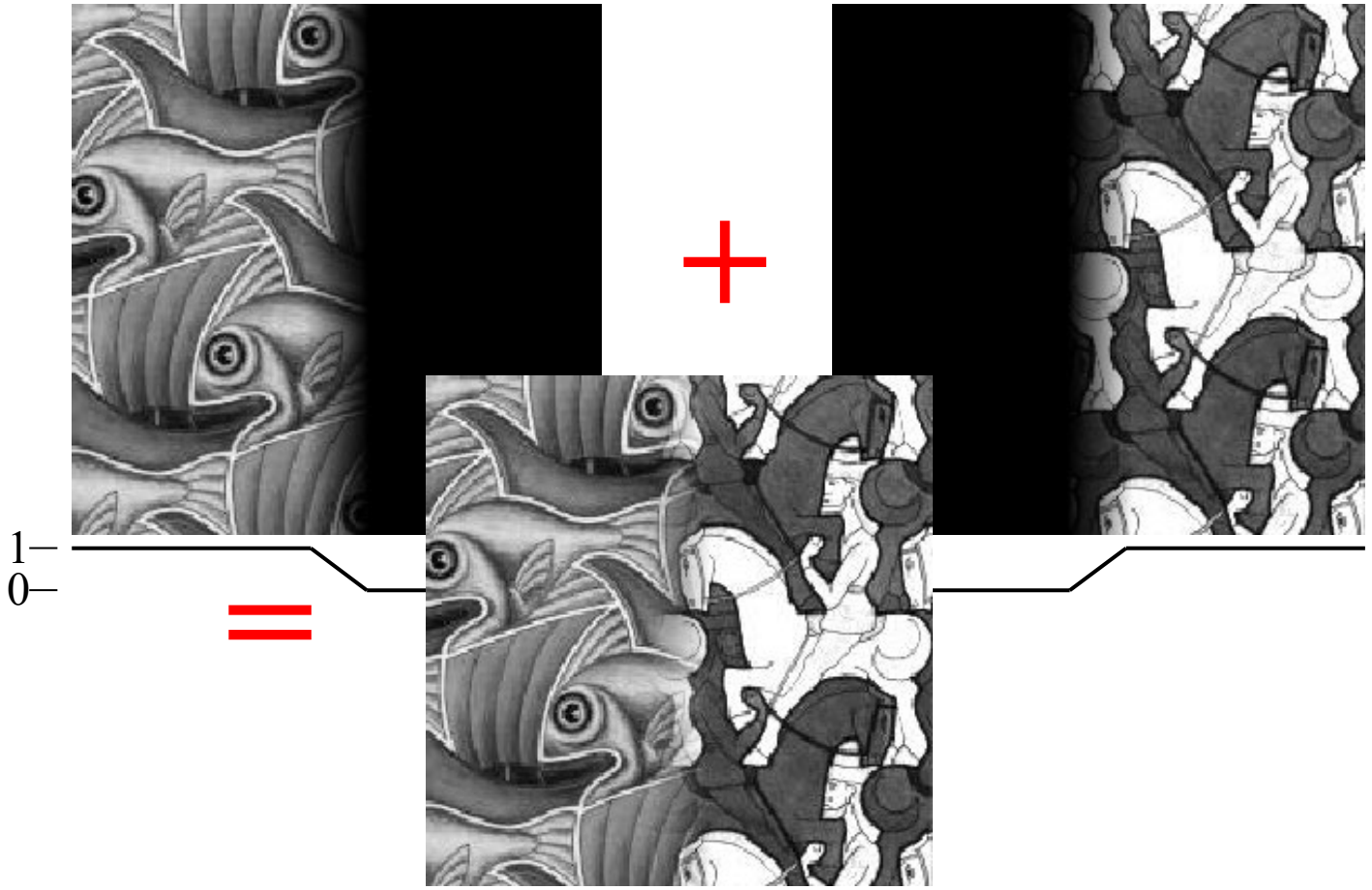


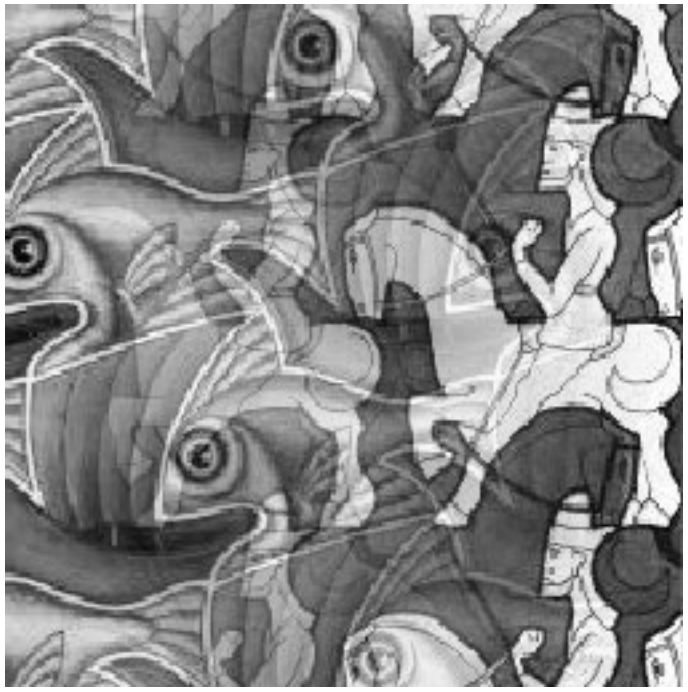
Image Feathering

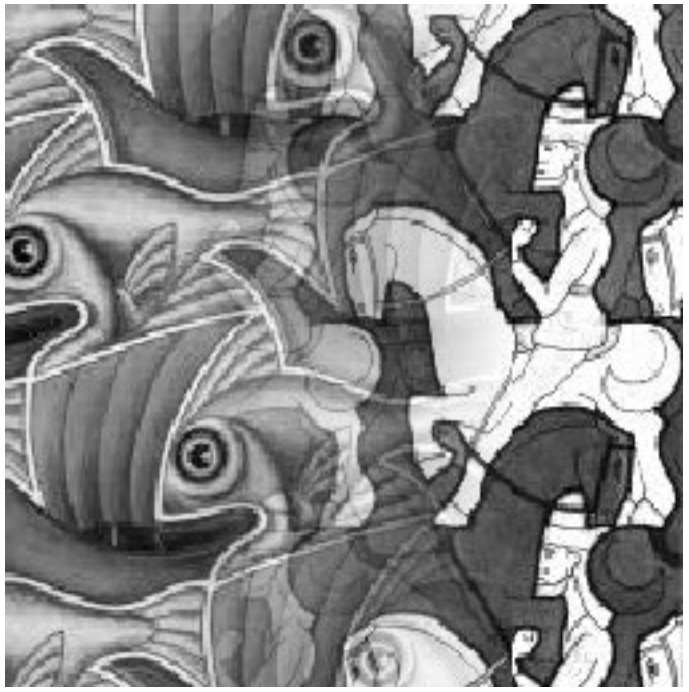


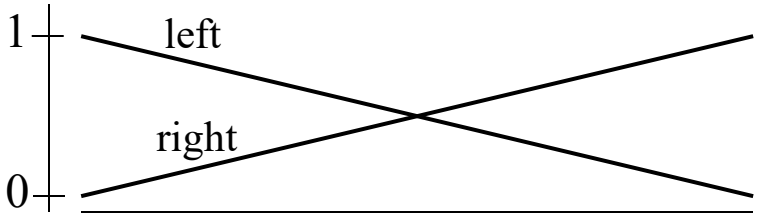
Feathering

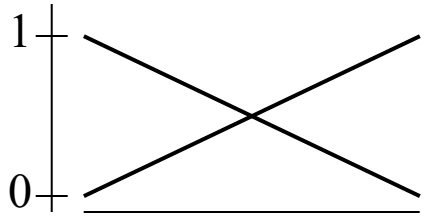


Effect of window size

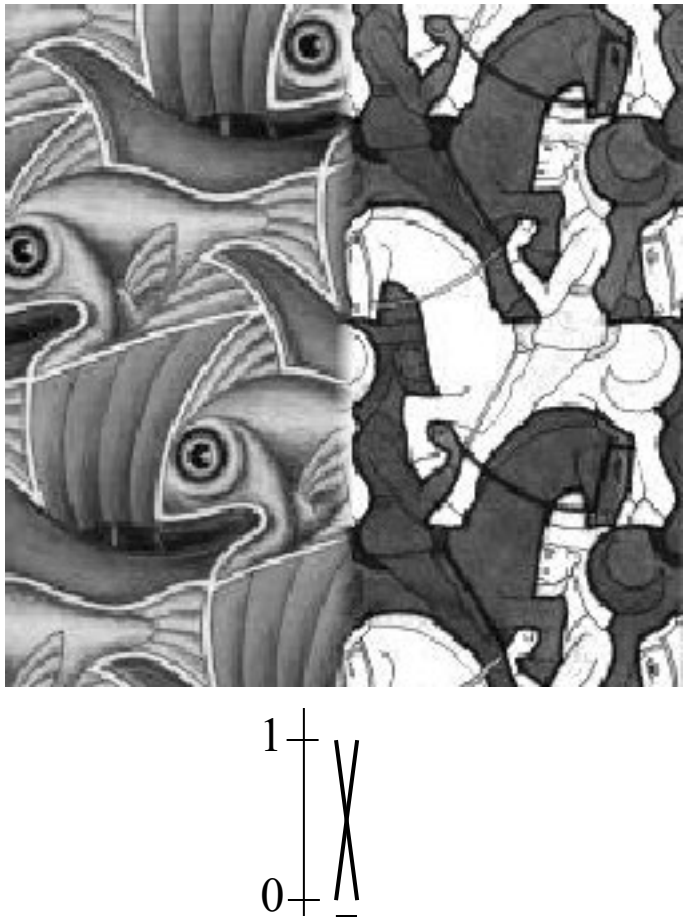


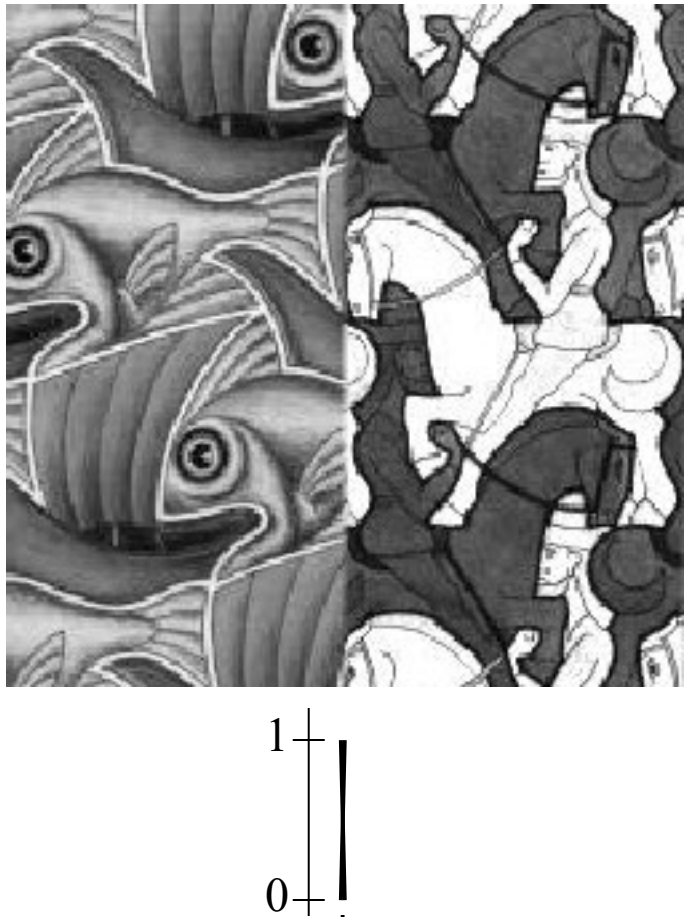




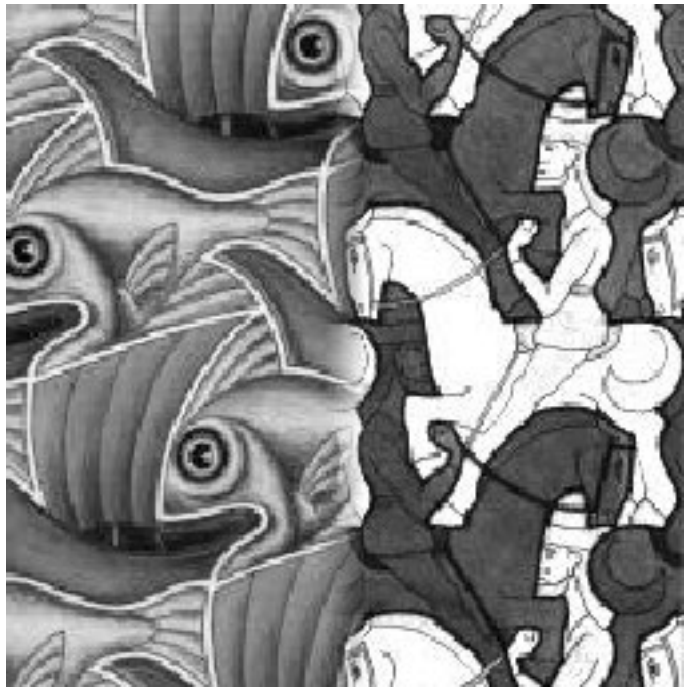


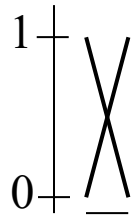
Effect of window size





Good window size

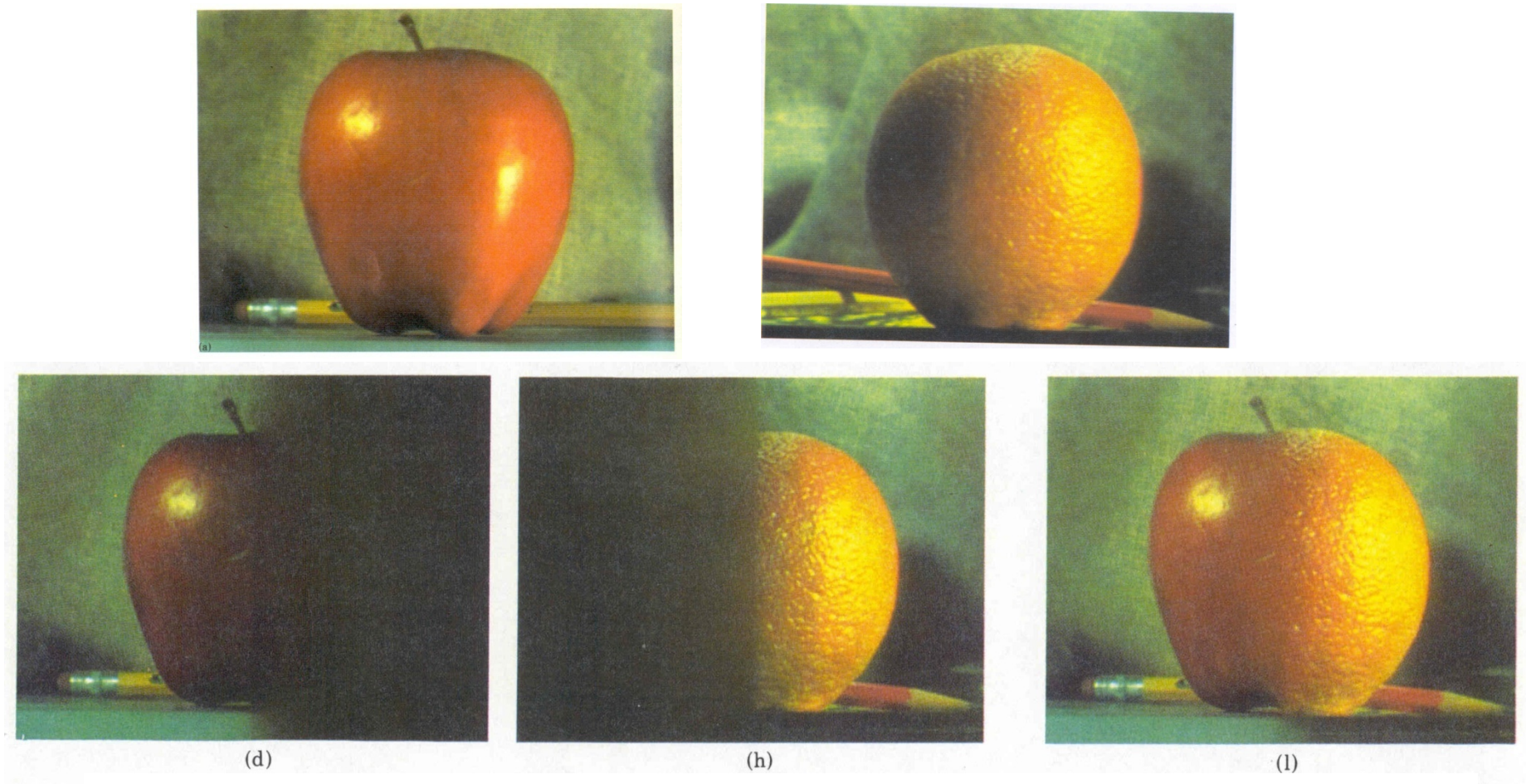




"Optimal" window: smooth but not ghosted

- Doesn't always work...

Pyramid Blending

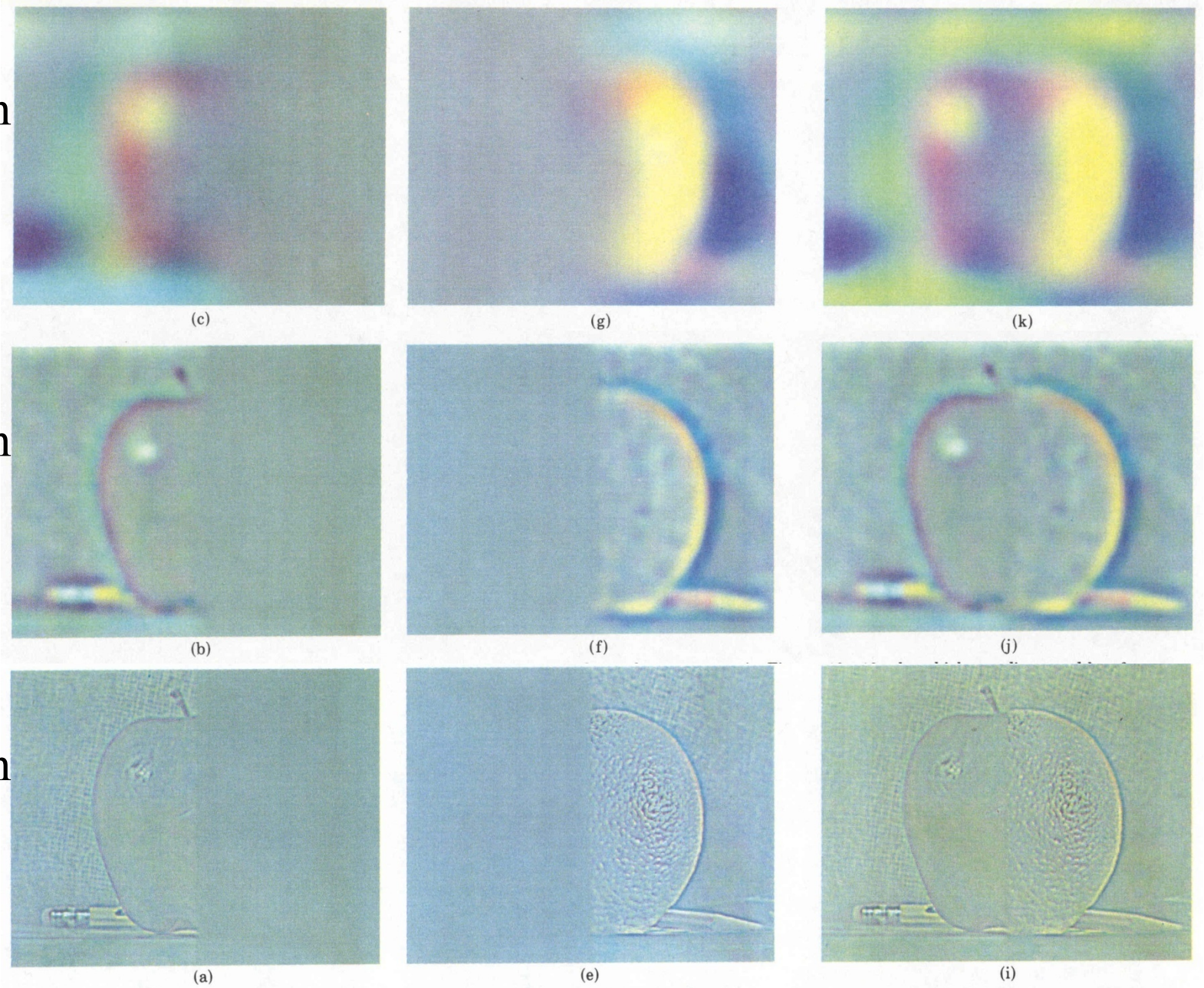


Burt, P. J. and Adelson, E. H., A multiresolution spline with applications to image mosaics, ACM Transactions on Graphics, 42(4), October 1983, 217-236.

Laplacian level 4

Laplacian level 2

Laplacian level 0



left pyramid right pyramid blended pyramid

Laplacian image blend

1. Compute Laplacian pyramid
2. Compute Gaussian pyramid on *weight* image (can put this in A channel)
3. Blend Laplacians using Gaussian blurred weights
4. Reconstruct the final image

Panoramic Video Textures

Output Video

http://grail.cs.washington.edu/projects/panovidtex/

[Agarwala et al, 2005]