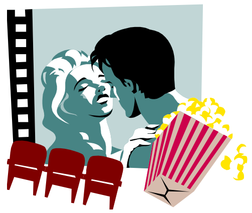
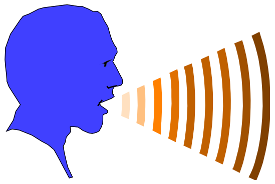
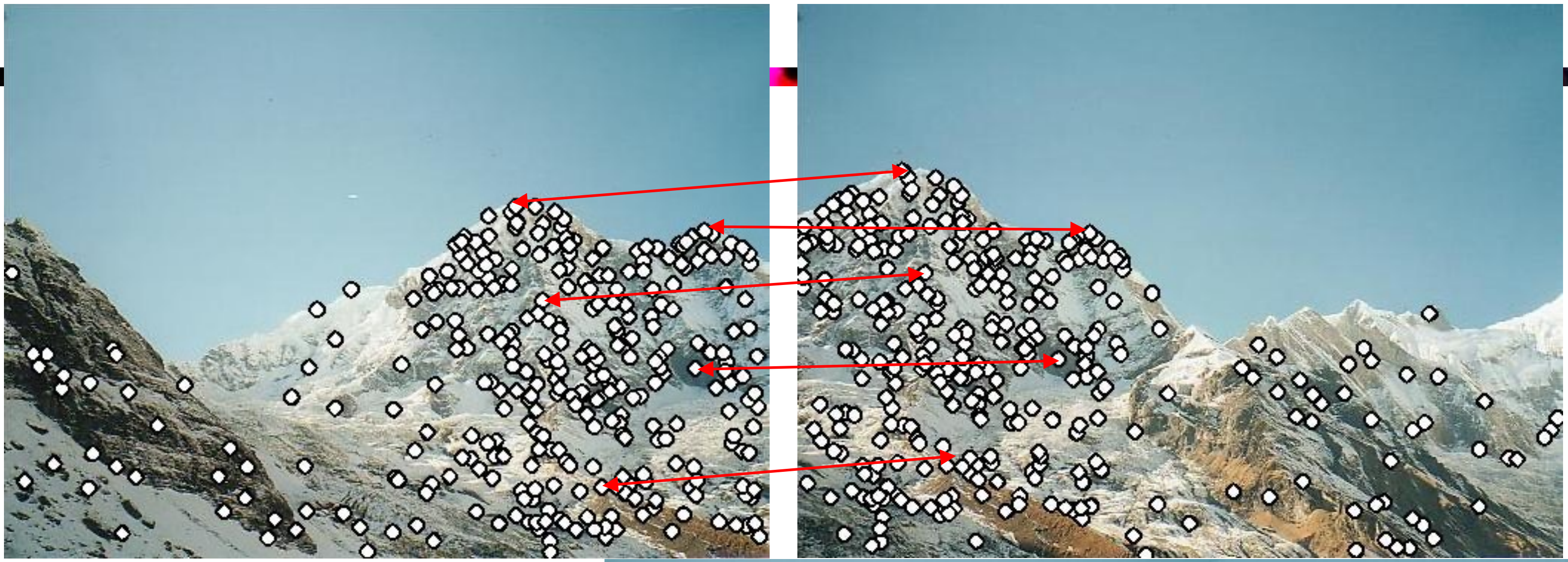
 

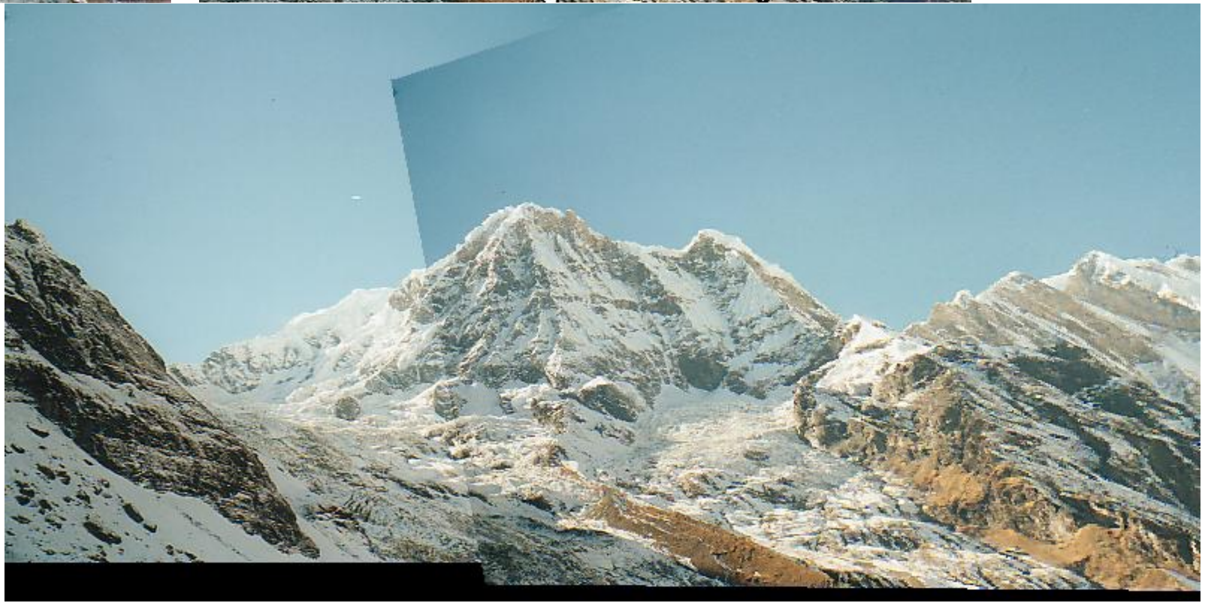
Image and Video Processing

Image Alignment and Stitching through Feature Correspondence

Yao Wang
Tandon School of Engineering, New York University

Panorama Stitching





From https://courses.cs.washington.edu/courses/cse455/16wi/notes/index.html, lecture on descriptors

Generating Background Sprite

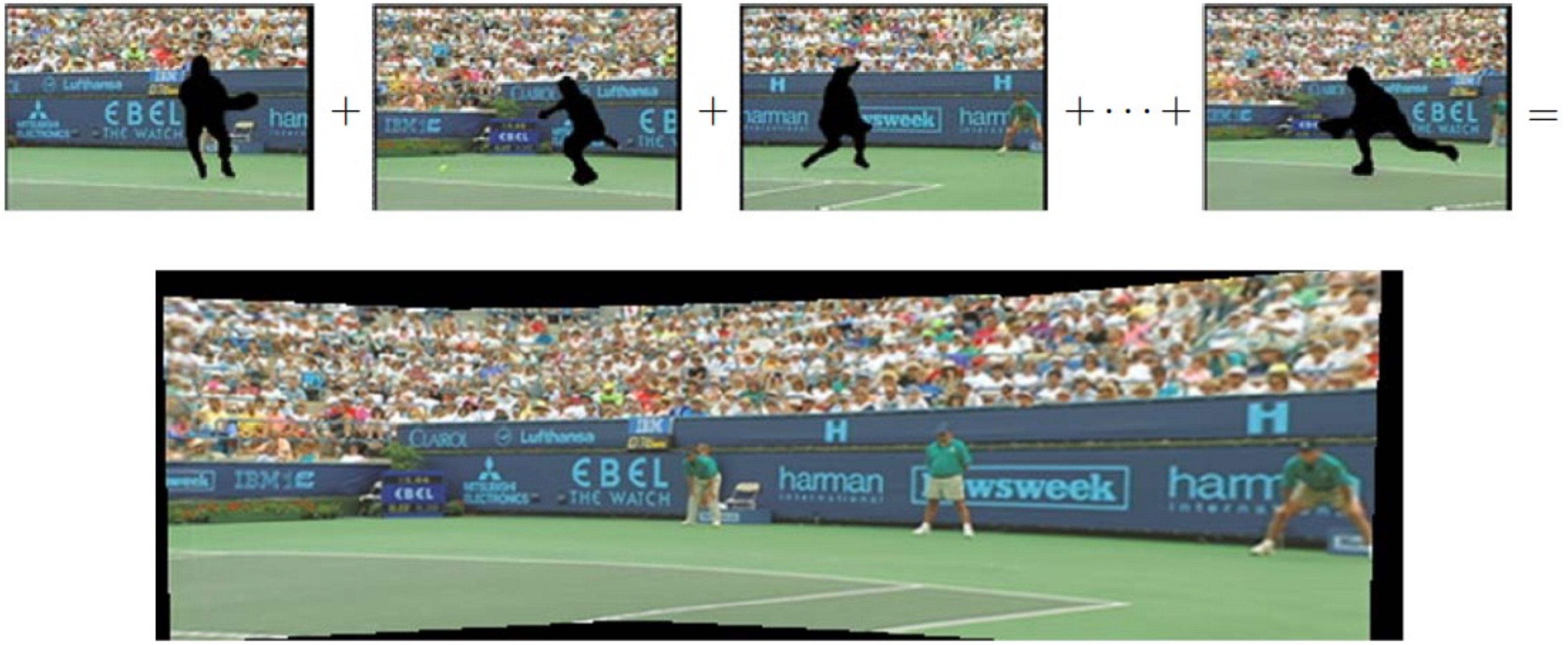


Figure 9.6 Video stitching the background scene to create a single *sprite* image that can be transmitted and used to re-create the background in each frame (Lee, ge Chen, lung Bruce Lin *et al.* 1997) © 1997 IEEE.

From [Szeliski 2010]

Overall Approach for Stitching Two Images and Problems to Study

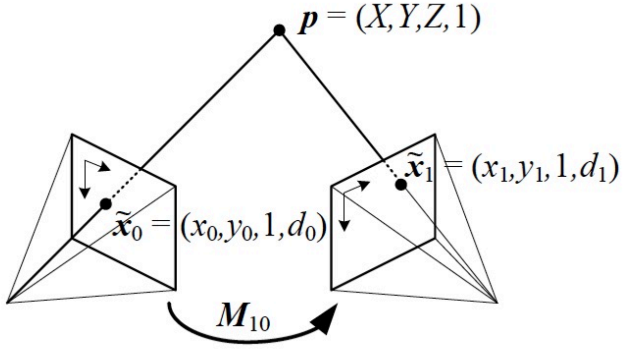
- Assume two images are related by some global geometric mapping $\mathbf{h}(\mathbf{x})$ due to camera motion (camera view point difference)
  - $\mathbf{h}(\mathbf{x})$: Image G at point $\mathbf{x}=(x,y)$ corresponds to point $\mathbf{x}' = \mathbf{h}(\mathbf{x}) = \{h_x(x,y), h_y(x,y)\}$ in image F
  - $G(x,y) = F(h_x(x,y), h_y(x,y))$
  - What mapping model $\mathbf{h}(\mathbf{x})=\{h_x(x,y), h_y(x,y)\}$ to use?
  - How to find the model parameters?
- Feature based methods
  - Detect feature points in each image
  - Find corresponding feature pairs based on their descriptors
  - Determine a global mapping based on point correspondence
- Warp one image using the determined mapping
- Stitch the reference image and the warped image

Lecture Outline

- Global transformation models
  - Camera model: 3D->2D projection (perspective projection)
  - Camera motion -> 2D transformation (homography, affine)
- Feature correspondence based on local descriptors
- Finding global transformation based on point correspondence
  - Least squares
  - Robust Estimator and RANSAC
- Image warping
- Image blending
- Panoramic view stitching

How are two images related?

- If two images F and G are taken of the same scene from different view points, they are related by a geometric mapping or transformation


$p = (X,Y,Z,1)$
$\tilde{x}_1 = (x_1, y_1, 1, d_1)$
$\tilde{x}_0 = (x_0, y_0, 1, d_0)$
M_{10}

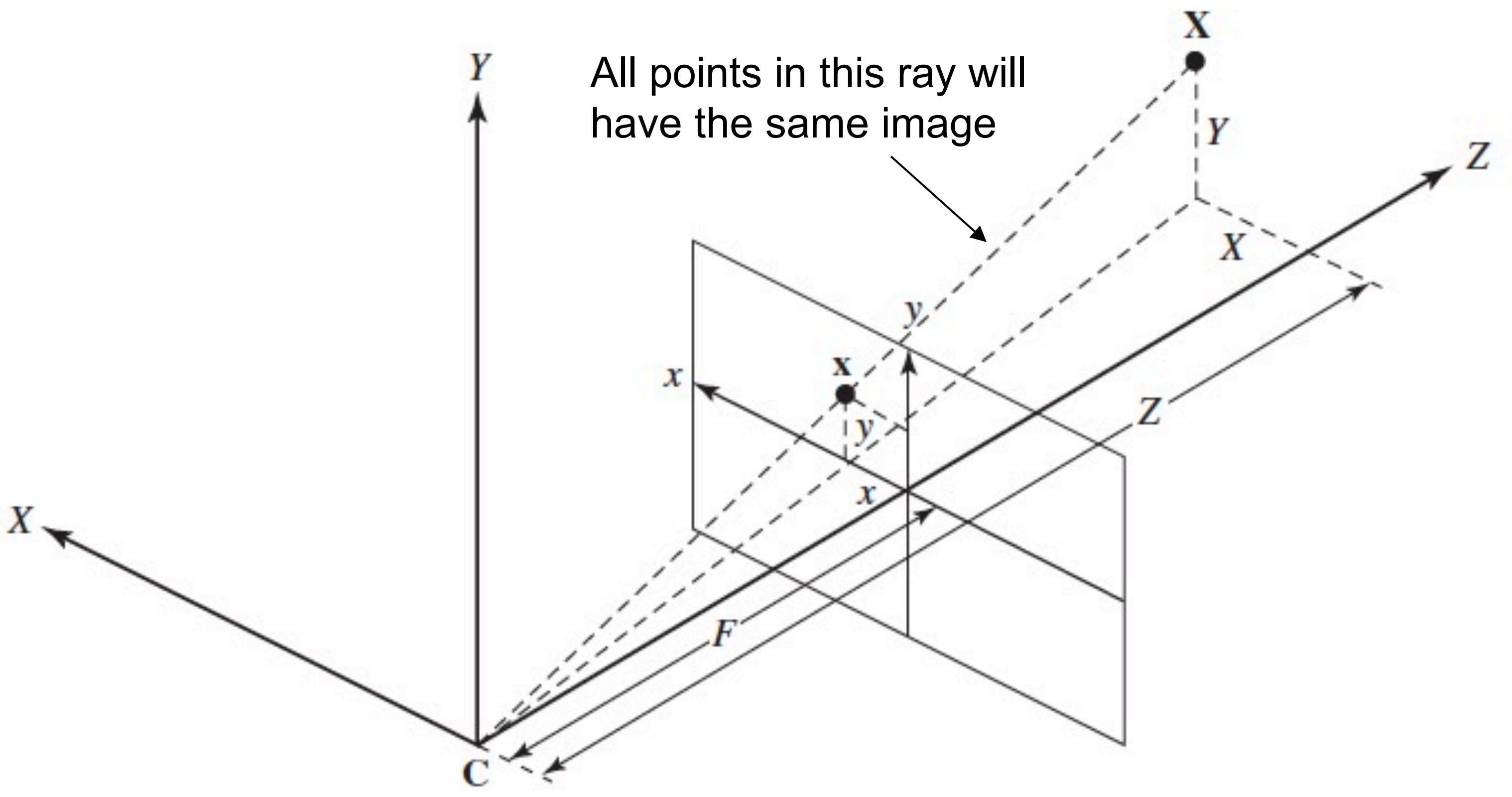
$\mathbf{x}_0$ in F corresponds to $\mathbf{x}_1$ in G

Mapping function:
$\mathbf{x}_1 = \mathbf{h}(\mathbf{x}_0)$
or
$x_1=h_x(x_0,y_0)$, $y_1=h_y(x_0,y_0)$

- What determines the mapping function?
  - Need to know camera 3D->2D projection geometry
  - Need to know how to model camera motion

Pinhole Camera Model: Perspective Projection

All points in this ray will have the same image



$$\frac{x}{F}=\frac{X}{Z},\frac{y}{F}=\frac{Y}{Z}\Rightarrow x=F\frac{X}{Z},y=F\frac{Y}{Z}$$

x, y are inversely related to Z

Representation Using Homogeneous Coordinate

Original coordinate: $\mathbf{x}=\begin{bmatrix} x \\ y \end{bmatrix}$

Homogeneous coordinate: $\tilde{\mathrm{x}}=\begin{bmatrix} wx \\ wy \\ w \end{bmatrix}$,

Conversion: $x=\dfrac{\tilde{x}}{w}, y=\dfrac{\tilde{y}}{w}$,

Representation of perspective projection in homogeneous coordinate

$$\tilde{\mathrm{x}}=\begin{bmatrix} F & 0 & 0 & 0 \\ 0 & F & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}=\mathbf{P}\tilde{\mathbf{X}}$$

$$\Longrightarrow$$

$$\frac{x}{F}=\frac{X}{Z}, \frac{y}{F}=\frac{Y}{Z} \Rightarrow x=F\frac{X}{Z}, y=F\frac{Y}{Z}$$

x, y are inversely related to Z

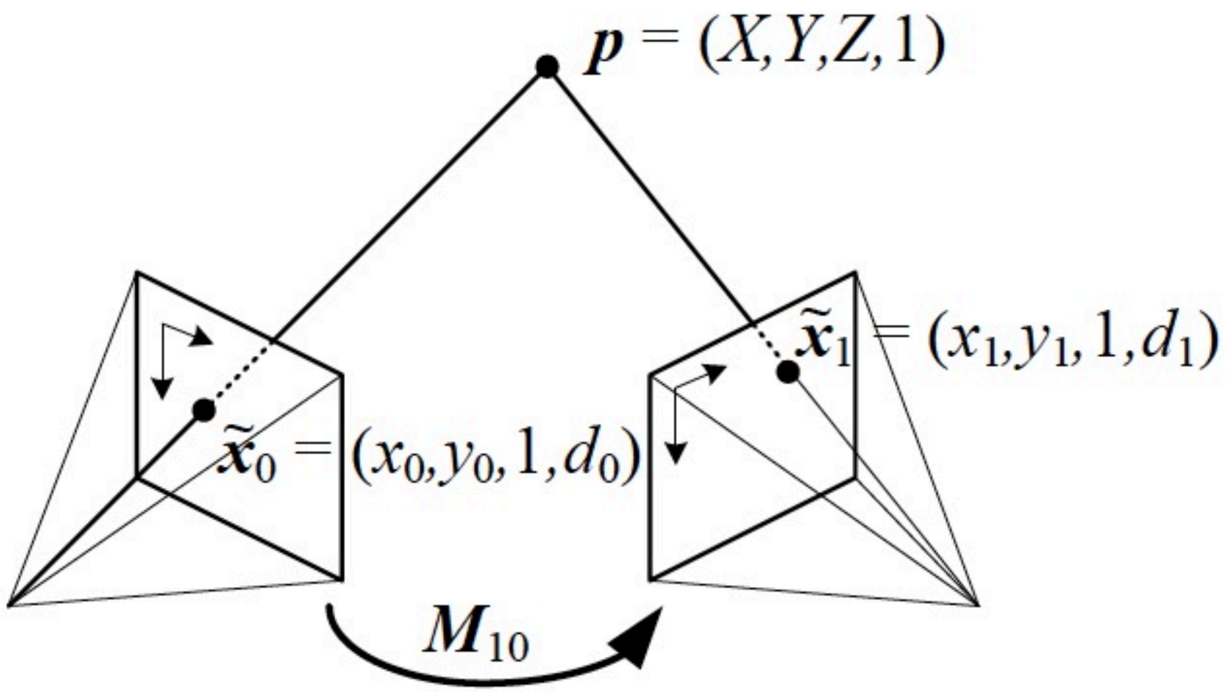
This representation assumes the camera coordinate is aligned with the world coordinate defining $\mathbf{X}$

If we consider the possible rotation and translation of the camera coordindate with the world coordinate and the scaling of the pixel coordinate, we will have the more general relation

$\tilde{\mathrm{x}}=\mathbf{K}\ [\mathbf{R}\ \mathbf{t}]\tilde{\mathbf{X}}$

where $\mathbf{K}$ depends on the camera intrinsic parameters, and $\mathbf{R}, \mathbf{t}$ are camera extrinsic parameters (rotation and translation with respect to world coordinate.

Mapping between Two Cameras



$\boldsymbol{p} = (X,Y,Z,1)$
$\tilde{\boldsymbol{x}}_1 = (x_1,y_1,1,d_1)$
$\tilde{\boldsymbol{x}}_0 = (x_0,y_0,1,d_0)$
$\boldsymbol{M}_{10}$

The two cameras are rotated and translated with each other. How are the image coordinates (x_0,y_0) and (x_1,y_1) related?

3D Rotation and Translation

Rotation and translation wrp. the center **C**:

$\mathbf{X}' = [\mathbf{R}](\mathbf{X}-\mathbf{C})+\mathbf{T}+\mathbf{C} = [\mathbf{R}]\mathbf{X}+\mathbf{T}'; \quad [\mathbf{R}]: \theta_x, \theta_y, \theta_z; \quad \mathbf{T}: T_x, T_y, T_z$

$$[\mathbf{R}] = [\mathbf{R}_z]\cdot[\mathbf{R}_y]\cdot[\mathbf{R}_x]$$

$$[\mathbf{R}_x] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \quad [\mathbf{R}_y] = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \quad [\mathbf{R}_z] = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$[\mathbf{R}] = \begin{bmatrix} \cos\theta_y\cos\theta_z & \sin\theta_x\sin\theta_y\cos\theta_z - \cos\theta_x\sin\theta_z & \cos\theta_x\sin\theta_y\cos\theta_z + \sin\theta_x\sin\theta_z \\ \cos\theta_y\sin\theta_z & \sin\theta_x\sin\theta_y\sin\theta_z + \cos\theta_x\cos\theta_z & \cos\theta_x\sin\theta_y\sin\theta_z - \sin\theta_x\cos\theta_z \\ -\sin\theta_y & \sin\theta_x\cos\theta_y & \cos\theta_x\cos\theta_y \end{bmatrix}$$

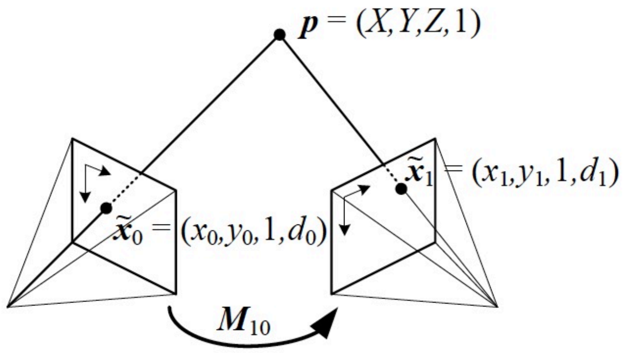
When all rotation angles are small:

$$[\mathbf{R}] \approx [\mathbf{R}'] = \begin{bmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{bmatrix}$$

When rotating around Z-axis only:

$$[R] = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

General Mapping Function


$\boldsymbol{p} = (X, Y, Z, 1)$
$\tilde{\boldsymbol{x}}_1 = (x_1, y_1, 1, d_1)$
$\tilde{\boldsymbol{x}}_0 = (x_0, y_0, 1, d_0)$
$\boldsymbol{M}_{10}$

Assume left camera uses the world coordinate,

$x_0 = FX/Z, y_0 = FY/Z,$

Right camera is rotated and translated, so that $p=(X,Y,Z)^T$ appears as p' for camera 2

$$p' = \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = Rp + T$$

$$\xrightarrow[\text{Perspective Projection}]{}$$

$x_1 = FX'/Z', y_1 = FY'/Z'.$

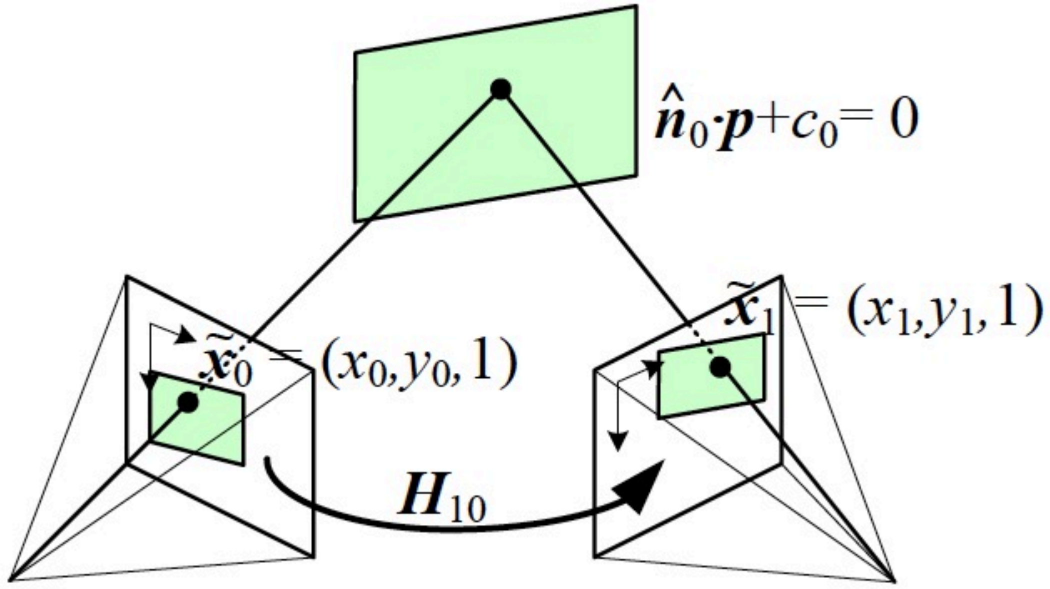
Can show:

$$x_1 = F\frac{(r_1 x_0 + r_2 y_0 + r_3 F)Z + T_x F}{(r_7 x_0 + r_8 y_0 + r_9 F)Z + T_z F}$$

$$y_1 = F\frac{(r_4 x_0 + r_5 y_0 + r_6 F)Z + T_y F}{(r_7 x_0 + r_8 y_0 + r_9 F)Z + T_z F}$$

The mapping function depends on Z and hence varies for every image point (corresponding to 3D points with different Z) in general.

Planar Homography: Mapping for Points on the Same Plane between Two Camera Views



$\hat{n}_0 \cdot p + c_0 = 0$
$\tilde{x}_0 = (x_0, y_0, 1)$
$\tilde{x}_1 = (x_1, y_1, 1)$
H_{10}

Let the plane be represented by Z=aX+bY+c, previous general relation becomes

$$x_1 = \frac{a_0 + a_1 x_0 + a_2 y_0}{c_0 + c_1 x_0 + c_2 y_0}, \quad y_1 = \frac{b_0 + b_1 x_0 + b_2 y_0}{c_0 + c_1 x_0 + c_2 y_0}$$

The parameters depend on plane parameters (a,b,c) and camera parameters (F, R, T)

Note there is inherent scale ambiguity (unless the 3D position of a point is known!). Often we set $c_0 = 1$

Using homogeneous coordinate:

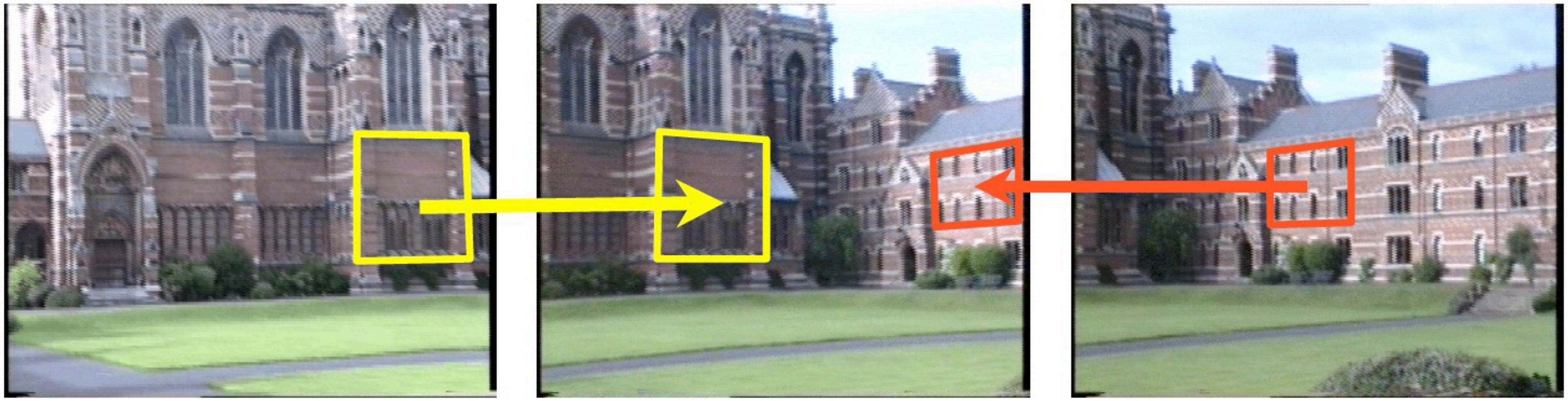
$$\begin{bmatrix} x_1 \\ y_1 \\ w \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ c_1 & c_2 & c_0 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

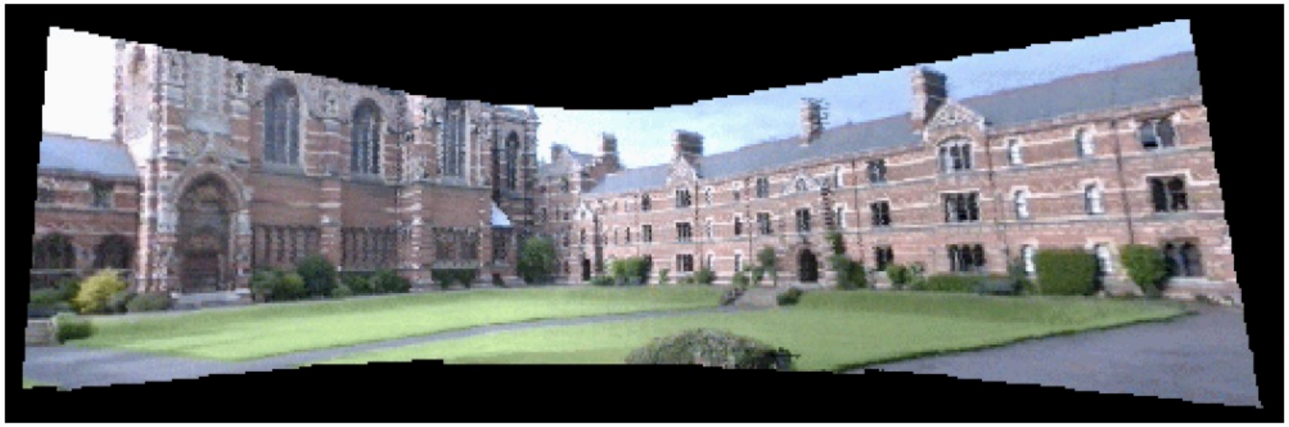
$\tilde{\mathbf{x}}_1 = \mathbf{H}\tilde{\mathbf{x}}_0$ 8 parameters assuming c_0=1

What if the imaged scene is not planar?

- The relation is approximately true for all image points if the imaged scene is far away and can be considered to be at the same depth.
- If a scene contains multiple planes (e.g. image of a building with different plane surfaces), each corresponding plane pair are related by a different homography.

Multiplane Homographies





from Hartley & Zisserman

Need to segment different planes and use different homography mapping to each plane segment

Special Cases

- General relation

$$x' = F\frac{(r_1x + r_2y + r_3\,F)Z + T_xF}{(r_7x + r_8y + r_9\,F)Z + T_zF}$$
$$y' = F\frac{(r_4x + r_5y + r_6\,F)Z + T_yF}{(r_7x + r_8y + r_9\,F)Z + T_zF}$$

- Case 1: Arbitrary camera motion, but the scene is a flat surface ($Z = aX + bY + c$)

$$x' = \frac{h_1x + h_2y + h_3}{h_7x + h_8y + h_9}$$
$$y' = \frac{h_4x + h_5y + h_6}{h_7x + h_8y + h_9}$$

Homography

- Case 2: No translation, only rotation ($T_x = T_y = T_z = 0$)
  - Is also a homography!

$$x' = F\frac{r_1x + r_2y + r_3\,F}{r_7x + r_8y + r_9\,F}$$
$$y' = F\frac{r_4x + r_5y + r_6\,F}{r_7x + r_8y + r_9\,F}$$

- Case 3: When rotation and translation is in the image plane and the scene has a constant depth (Z=const)

$$[R] = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathrm{T} = \begin{bmatrix} T_x \\ T_x \\ 0 \end{bmatrix}$$

- Can reduce to affine mapping:

$$x' = a_1x + a_2y + a_0$$
$$y' = b_1x + b_2y + b_0$$

Approximation of Homography by Affine and Bilinear Model

- Affine (6 parameters):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_0 + a_1 u + a_2 v \\ b_0 + b_1 u + b_2 v \end{bmatrix}$$

$$\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ w \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

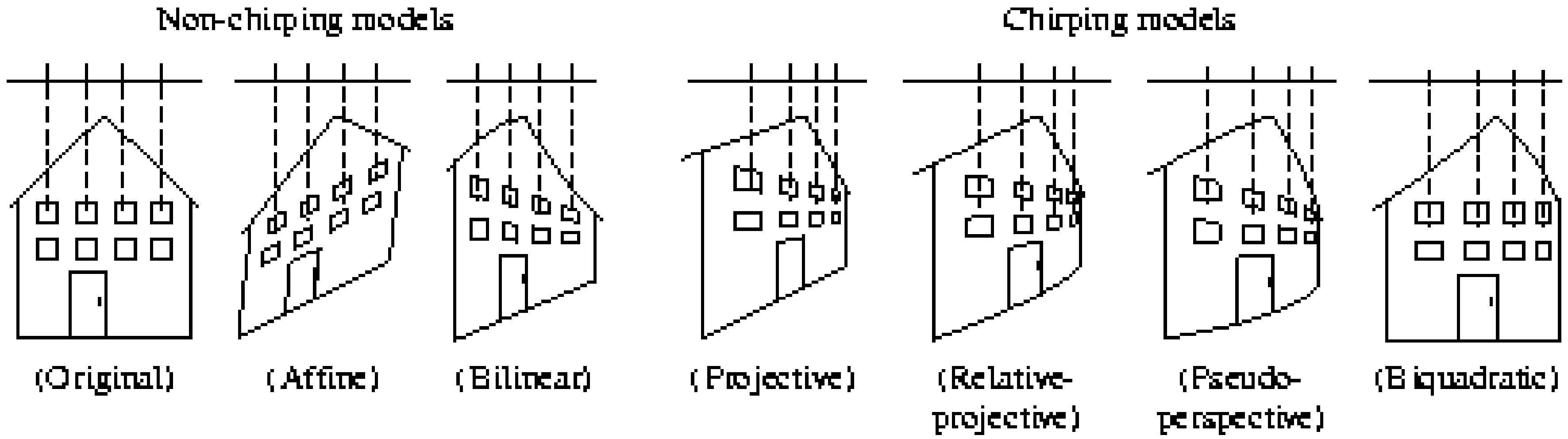
  - Affine model sufficiently capture mapping due to in-plane camera motion (scaling, roll and translation in x,y only), and Z is fairly constant (far away)
  - Also known as affine homography

- Bilinear (8 parameters):

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_0 + a_1 u + a_2 v + a_3 uv \\ b_0 + b_1 u + b_2 v + b_3 uv \end{bmatrix}$$

Here we use (u,v) to represent (x,y) before, (x,y) to represent (x’,y’) before

Homography and Its Approximations



Features of projective mapping (homography):

- Chirping: increasing perceived spatial frequency for far away objects
- Converging (Keystone): parallel lines converge in distance
- Keep straight line straight!

Affine: No converging no chirping. Keep straight lines straight. Sufficient for in-plane motion!

Bilinear: Has converging, but no chirping. Slanted lines becomes curves

Pop Quiz

- What does planar homography mean?
- How many parameters define the homography?
- Under what circumstances does the general mapping between two camera views reduce to homography?
- Under what circumstances does planar homography reduces to affine mapping?
- How many parameters define the affine mapping?

Pop Quiz: Solution

- What does planar homography mean?
  - Mapping between projected images of a 3D plane in two different camera views
- How many parameters define the homography?
  - 8: physically related to 3 rotation angles, 3 translation angles and the 3 plane parameters. But 1 parameter inherently underdetermined
- Under what circumstances does the general mapping between two camera views reduce to homography?
  - When the imaged scene can be approximated by a plane
  - Or when the two cameras are only rotations of each other
- Under what circumstances does planar homography reduces to affine mapping?
  - If the two cameras lie in the same plane (in-plane rotation, scaling, and shifting only)
- How many parameters define the affine mapping?
  - 6

How to determine the homography mapping functions?

- If we know the camera relation precisely, we can derive the mapping function directly assuming the 3D points lie in a plane (approximately)
- Otherwise, we have to infer the mapping function from the images!
- Feature based
  - Find corresponding feature points in two images based on their local descriptors:
    - (u_n, x_n), n=1,2,…,N
  - Find the mapping parameters **a** so that u_n and x_n are related by the mapping for all or most pairs as best as it can.
- Intensity based
  - Directly minimizing intensity difference under the mapping
    - $E(a) = \sum_x \left(I_1(x) - I_2(h(x,a))\right)^2$
- We will focus on feature-based approach in this lecture

Feature Correspondence Based on Local Descriptors (Overall Approach)

- Given two images
- Detect features in each image (e.g. Harris or SIFT feature detector or Multi-scale Harris)
- Determine a descriptor for each feature point (SIFT descriptor)
- For each feature point in image one, find the feature point in another image with most similar feature descriptor

Feature Correspondence Based on Local Descriptors (Details)

- How to measure feature similarity?
  - For SIFT descriptor, L2 distance (=Euclidean distance) is often used

$$d(f_1, f_2) = \|f_1 - f_2\|_2 = \left(\sum_k \left(f_{1,k} - f_{2,k} \right)^2 \right)^{1/2}$$

- Should we just take the one with the shortest distance?
  - What if the closest match still has large distance?
    - Compare to a threshold T. Reject if distance> T
  - What if there are two points with very similar distances
    - There are uncertainty with the match
    - Reject both or keep both
    - OK to have some false matches at this stage
      - To be corrected based on global motion-based matching

More Details

- For each feature point in image 1
- Compute its distance to each feature point in image 2 and find the two points with shortest distance d_1 and d_2 ($d_1 <= d_2$)
- If d1 < T_d and $d_1/d_2 < T_r$, (e.g. $T_r = 0.8$) take the point with minimal distance as the corresponding point
- Otherwise, reject this feature point from image 1

- Time consuming!

Fast Search Algorithms

- Hashing
- K-d tree
- See [Szeliski2012] Richard Szeliski, Computer Vision: Algorithms and Applications. 2012. Sec. 4.1.3

Determine mapping function from candidate matching points

- Simple approach
  - Least squares fitting
- Robust estimation
  - Need to ignore effect of outliers (false matching candidates)
  - Iteratively reweighted least squares (IRLS)
  - Least median of squares (LMS)
  - RANSAC

Simple Approach

- Assume the mapping function is a polynomial with N parameters in each dimension (total unknows=2N)

$$\begin{cases} x = a_0 + a_1 u + a_2 v + a_3 uv + \ldots \\ y = b_0 + b_1 u + b_2 v + b_3 uv + \ldots \end{cases}$$

- Step 1: Identify K≥N corresponding points between two images, i.e.

$$(u_i, v_i) \leftrightarrow (x_i, y_i), i = 1,2\ldots, K.$$

- Step 2: Determine the coefficients a_i, b_i, i = 0,…,N-1 by solving

$$\begin{cases} x(u_i, v_i) = a_0 + a_1 u_i + a_2 v_i + \cdots = x_i, \\ y(u_i, v_i) = b_0 + b_1 u_i + b_2 v_i + \cdots = y_i, \end{cases} \quad i = 1,2,\ldots, K$$

- How to solve this?

How to Solve the Previous Equations?

- Convert to matrix equation:

$$\mathbf{A}\mathbf{a} = \mathbf{x}, \quad \mathbf{A}\mathbf{b} = \mathbf{y}$$

$$\textit{where}$$

$$\mathbf{A} = \begin{bmatrix} 1 & u_1 & v_1 & \cdots \\ 1 & u_2 & v_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ 1 & u_K & v_K & \cdots \end{bmatrix}, \mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix}$$

If K = N, and the matrix **A** is non-singular, then

$$\mathbf{a} = \mathbf{A}^{-1}\mathbf{x}, \quad \mathbf{b} = \mathbf{A}^{-1}\mathbf{y}$$

If K > N, then we can use a least square solution (best fit for all points)

$$\mathbf{a} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x}, \quad b = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T y$$

If K < N, or **A** is singular, then more corresponding feature points must be identified.
What does “singular” imply?

Least Squares Fitting

- Use more than N points when unknown is 2N
- Find a mapping that “best fit” all points by minimizing the average fitting error (least squares solution)

$$E(\mathbf{a}) = \sum_k (A_n\mathbf{a} - x_n)^2 = \left\| \mathbf{A}\mathbf{a} - \mathbf{x} \right\|^2 -> \min$$

$$\frac{\partial E}{\partial \mathbf{a}} = 2\mathbf{A}^T(\mathbf{A}\mathbf{a} - \mathbf{x}) = 0$$

$$\mathbf{a} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{x}$$

- More robust to matching errors

Pop Quiz

- What is A matrix for affine mapping?
- How many feature pairs do we need to determine affine mapping parameters?
- How do we determine the parameters given K corresponding points?

Pop Quiz: Solution

- What is A matrix for affine mapping?

$$A = \begin{bmatrix} 1 & u_1 & v_1 \\ \cdots & \cdots & \cdots \\ 1 & u_K & v_K \end{bmatrix}, \mathrm{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \mathrm{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$$

- How many feature pairs do we need to determine affine mapping parameters?
  - We need to find 3 or more pairs of corresponding points
- How do we determine the parameters given K corresponding points?
  - If we have only 3 pairs (K=3), we can solve the following equation:

$$\mathbf{a} = \mathbf{A}^{-1}\mathbf{x}, \quad \mathbf{b} = \mathbf{A}^{-1}\mathbf{y}$$

  - If we have more than 3 pairs (K>3), we solve the following over-determined equation:

$$\mathbf{a} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x}, \quad b = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T y$$

How to solve for homography parameters?

- Not a polynomial
- How to set up a equation?

$$x = \frac{a_0 + a_1 u + a_2 v}{1 + c_1 u + c_2 v}, \quad y = \frac{b_0 + b_1 u + b_2 v}{1 + c_1 u + c_2 v}$$

- How many pairs are needed?

Solution 1 (assuming c_0=1)

$$x = \frac{a_0 + a_1 u + a_2 v}{1 + c_1 u + c_2 v}, \quad y = \frac{b_0 + b_1 u + b_2 v}{1 + c_1 u + c_2 v} \Rightarrow$$

$$a_0 + a_1 u + a_2 v - c_1 u x - c_2 v x = x$$

$$b_0 + b_1 u + b_2 v - c_1 u y - c_2 v y = y$$

$$\begin{bmatrix} \dots \\ \dots \\ 1 & u_n & v_n & 0 & 0 & 0 & -u_n x_n & -v_n x_n \\ 0 & 0 & 0 & 1 & u_n & v_n & -u_n y_n & -v_n y_n \\ \dots \\ \dots \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \dots \\ \dots \\ x_n \\ y_n \\ \dots \\ \dots \end{bmatrix} \Rightarrow \mathbf{A}\mathbf{a} = \mathbf{x}$$

Need at least 4 pairs of corresponding points!
Least squares solution with > 4 pairs will be more reliable.

With homography, one cannot solve parameters for x and y mappings separately!

Alternate Solution (Direct Linear Transform or DLT)

$$x = \frac{a_0 + a_1 u + a_2 v}{c_0 + c_1 u + c_2 v}, \quad y = \frac{b_0 + b_1 u + b_2 v}{c_0 + c_1 u + c_2 v} \Rightarrow$$

$$a_0 + a_1 u + a_2 v - c_0 x - c_1 u x - c_2 v x = 0$$

$$b_0 + b_1 u + b_2 v - c_0 y - c_1 u y - c_2 v y = 0$$

Does not restrict c_0=1
Find normalized solution: ||a||=1

$$\begin{bmatrix} \dots \\ \dots \\ 1 & u_n & v_n & 0 & 0 & 0 & -x_n & -u_n x_n & -v_n x_n \\ 0 & 0 & 0 & 1 & u_n & v_n & -y_n & -u_n y_n & -v_n y_n \\ \dots \\ \dots \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_0 \\ b_1 \\ b_2 \\ c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} \dots \\ \dots \\ 0 \\ 0 \\ \dots \\ \dots \end{bmatrix} \Rightarrow \mathbf{A}_{2Lx9}\mathbf{a}_{9x1} = 0$$

Least squares problem: minimize $\|\mathbf{Aa}\|_2^2 = \mathbf{a}^T\mathbf{A}^T\mathbf{Aa}$

Solution: $\mathbf{a}^* =$ eigen vector of $\mathbf{A}^T\mathbf{A}$ with the minimal eigenvalue $\lambda_{\min}$ $(\mathbf{a}^T\mathbf{A}^T\mathbf{Aa} = \lambda_{\min}^2)$

Affine vs. Homography

homography matrix: 9 elements, but only 8 degrees of freedom (scale ambiguity)

$$H = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix}$$

$$\begin{array}{c} \begin{array}{cc} \text{rotation, scale, shear} & \text{translation} \end{array} \\ \begin{bmatrix} \boxed{\begin{matrix} h_{00} & h_{01} \\ h_{10} & h_{11} \end{matrix}} & \boxed{\begin{matrix} h_{02} \\ h_{12} \end{matrix}} \\ \begin{matrix} 0 & 0 \end{matrix} & 1 \end{bmatrix} \end{array}$$

In-plane camera motion only

$$\begin{array}{rc} & \begin{bmatrix} \boxed{\begin{matrix} h_{00} & h_{01} \\ h_{10} & h_{11} \end{matrix}} & \boxed{\begin{matrix} h_{02} \\ h_{12} \end{matrix}} \\ \boxed{\begin{matrix} h_{20} & h_{21} \end{matrix}} & 1 \end{bmatrix} \\ \text{perspective, etc.} & \end{array}$$

<u>affine homography</u> (no perspective)
- 6 elements (easier)
- estimate from 3 point correspondences

<u>homography</u> (without ambiguity)
- 8 elements
- estimate from 4 point correspondences

Need more points for the solution to be robust

[courtesy B. Girod, (c) 2013 Stanford University]

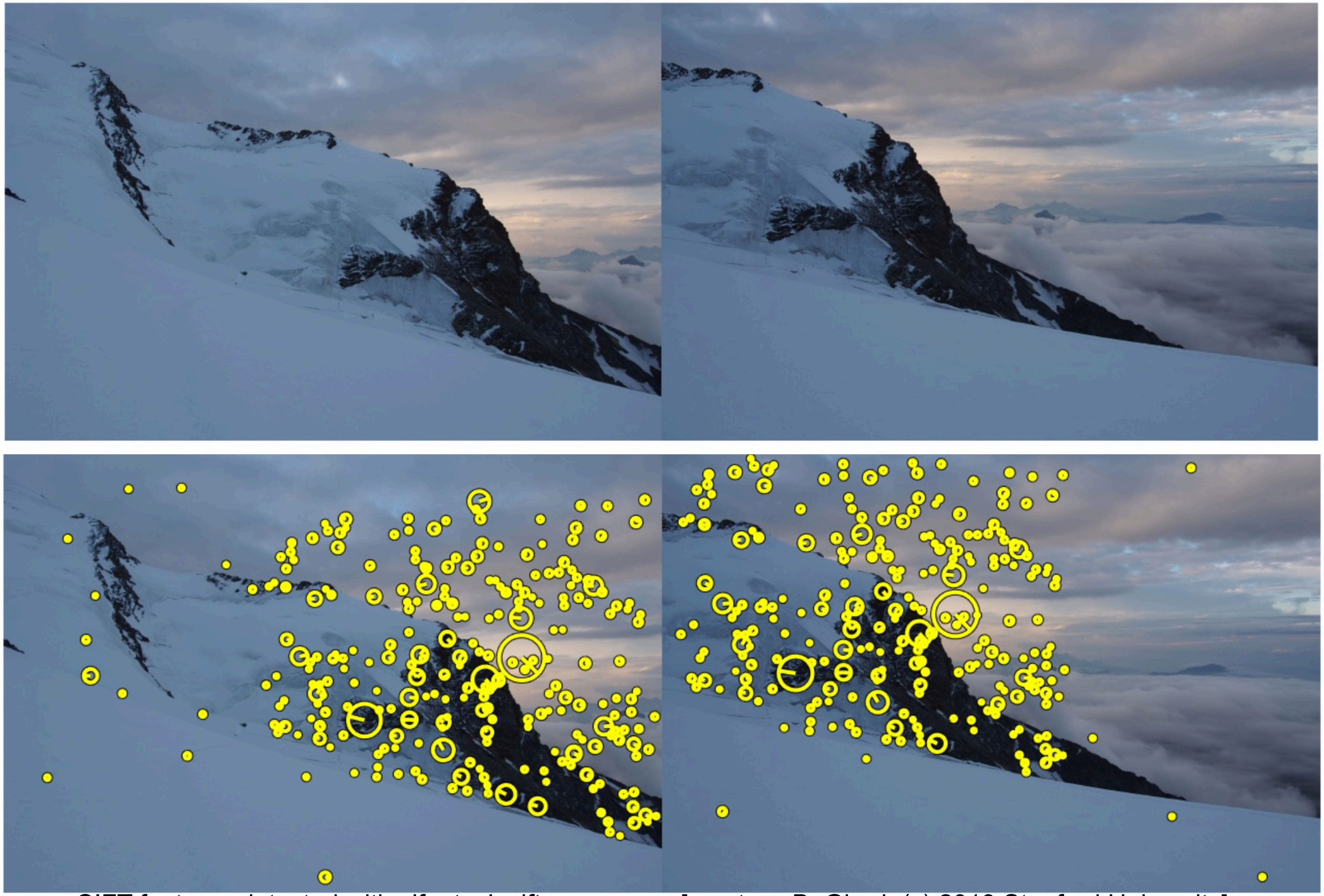
From http://web.stanford.edu/class/ee368/Handouts/Lectures/2016_Autumn/16-Panoramas_16x9_big.pdf

Some Practical Tips

- Shift the image coordinate so that the image center is (0,0)
- Scale the coordinates so that the average distance of all points to the center is sqrt(2)
- Will make the matrix A better conditioned -> more reliable results.

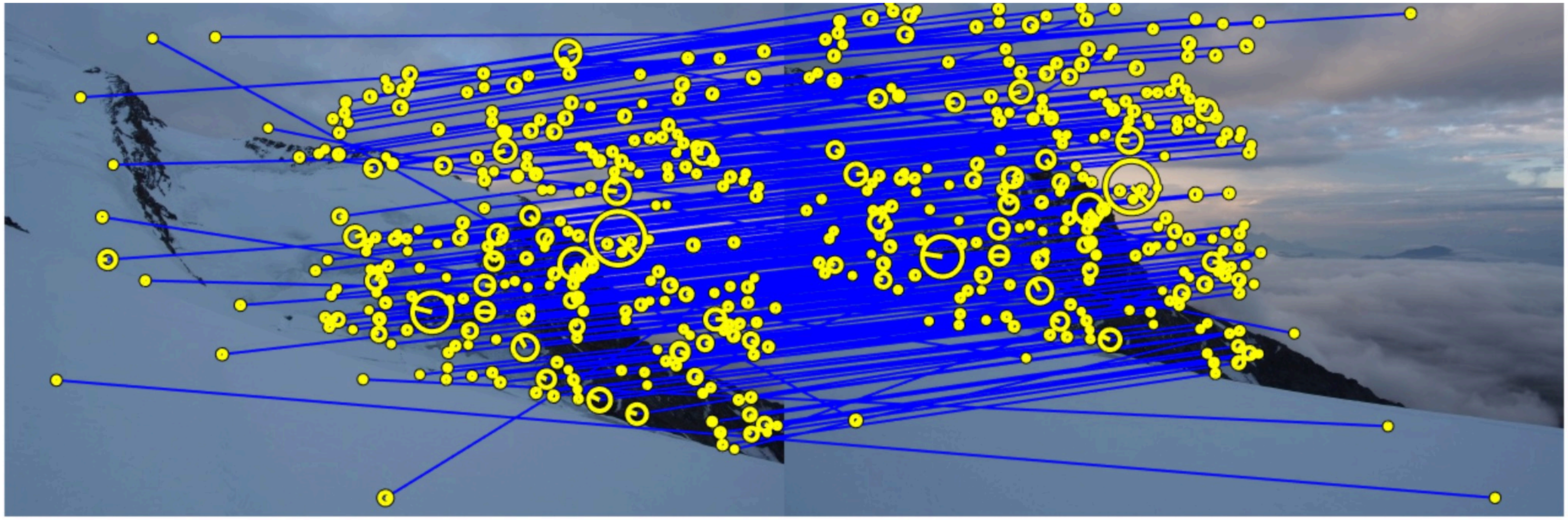
Problem with Least Squares

- Least squares gives equal weight to all candidate pairs.
- If some pairs are wrong matching (outlier), they could have adverse influence on the resulting solution.



SIFT features detected with vlfeat: vl_sift [courtesy B. Girod, (c) 2013 Stanford University]

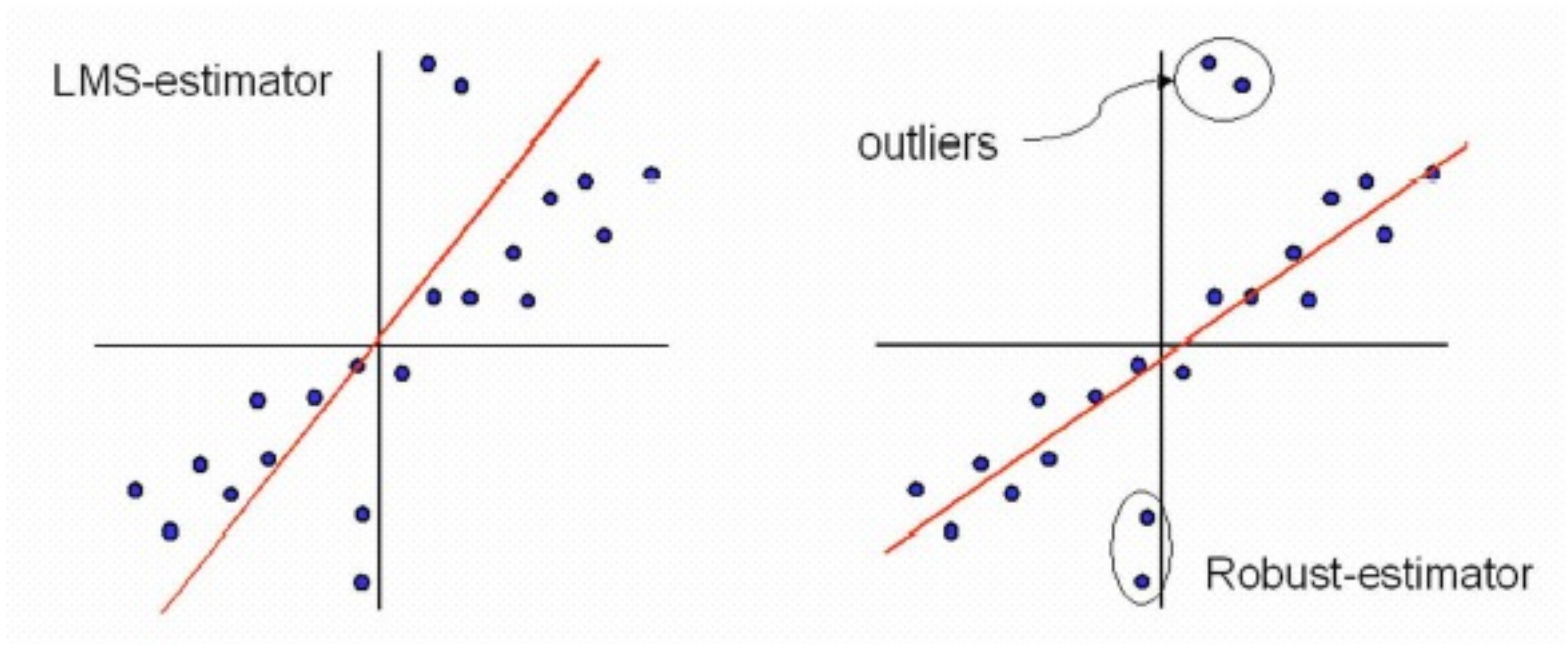
http://web.stanford.edu/class/ee368/Handouts/Lectures/2016_Autumn/16-Panoramas_16x9_big.pdf



Correspondence based on SIFT descriptor matching. [courtesy B. Girod, (c) 2013 Stanford University]
From http://web.stanford.edu/class/ee368/Handouts/Lectures/2016_Autumn/16-Panoramas_16x9_big.pdf

Robust Estimator: Line Fitting Example

- Try to recognize inliers (or outliers) and use only inliers for least square fitting

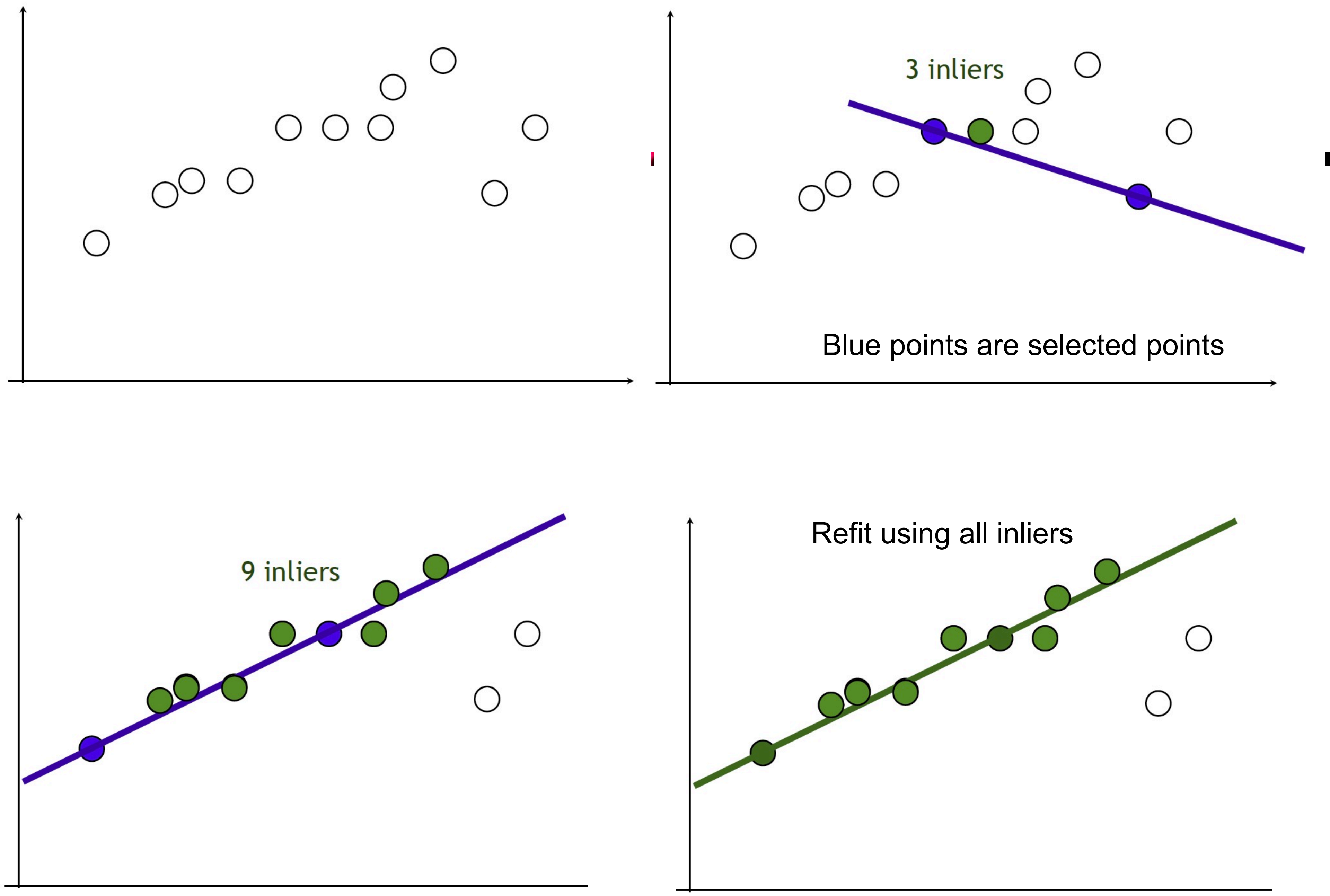


RANSAC (RANdom Sample Consensus): Basic Idea

- Choose sample points to derive the line
- See how other points fit with this line
  - # inlier = # points within a certain distance to this line
- Repeat many times
- Use the line that has the largest inlier set

RANSAC for line fitting

- Want to find points related with a line equation $y=ax+b$
- Given a set of points (x_l, y_l), $l=1,2,\ldots,L$
- Randomly select (or sample) 2 points from all candidate points
- Compute the line parameters (a,b) using selected points
- Apply resulting line model to each candidate point and evaluate the error (Euclidean distance between actual point and the line)
  - If $E_n>T$ (inlier threshold), consider this point an outlier
- Count the number of inliers for this trial
- Repeat the process S times
- Return the line model with the largest inlier count
- Refit the model using least square fit using all inlier points



From: http://courses.cs.washington.edu/courses/cse576/16sp/Slides/10_ImageStitching.pdf

RANSAC for Finding Geometric Mapping Based on Feature Points

- Given a set of candidate matching pairs ($\mathbf{u}_l$, $\mathbf{x}_l$), l=1,2,…,L
- Randomly select (or sample) K pairs from all candidate pairs
  - 2K >= N (number of parameters of geometric mapping)
- Compute the geometric mapping parameters **a** using selected pairs
- Apply resulting mapping **h**(**u**,**a**) to all candidate pairs and evaluate the mapping error (Euclidean distance) at each pair
  - $E_n = ||\mathbf{x}_n - \mathbf{h}(\mathbf{u}_n, \mathbf{a})||_2$
  - If $E_n > T$ (inlier threshold), consider pair n outlier
  - T is the distance in pixels that is considered significant, e.g. 1~3 pixels
- Count the number of inliers for this trial
- Repeat the process S times
- Return the mapping with the largest inlier count
- Refit the mapping using least square fit using all inlier pairs

How many trials are needed?

- q: probability that a randomly chosen pair (a sample) is correct
- K: the number of pairs taken in each trial to determine a transformation
- What is the probability that you need at least S sampling to get 1 set of K pairs that are all true correspondences?

How many trials are needed?

- q: probability that a randomly chosen pair (a sample) is correct (depending on how good initial candidates are!)
- K: the number of pairs taken in each trial to determine a transformation
- What is the probability that you need at least S sampling to get 1 set of K pairs that are all true correspondences?
- Probability that all K pairs are correct: q^K
- Probability that at least 1 pair is wrong match: $1-q^K$
- Probability that in S trials there are at least one trial where all k pairs are correct: P

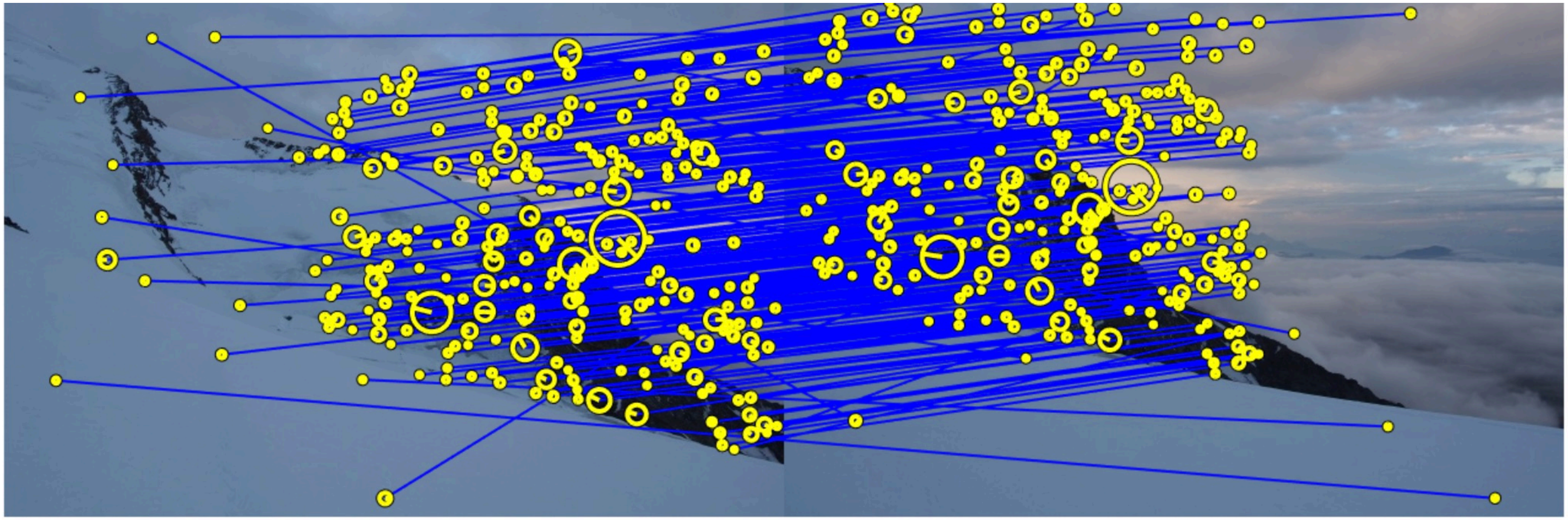
$$1-P=\left(1-q^K\right)^S \Rightarrow S=\frac{\log(1-P)}{\log\left(1-q^K\right)}$$

- Given P (a desired target, close to 1), S increases with K.
- So we want K to be as small as possible: K=N/2
- Ex. For homography, N=8 -> K=4, If q=0.3, P=0.99, S=566
- But, If q=0.5, S=72! (if we can preselect possible matching pairs to increase q, it reduces S substantially)

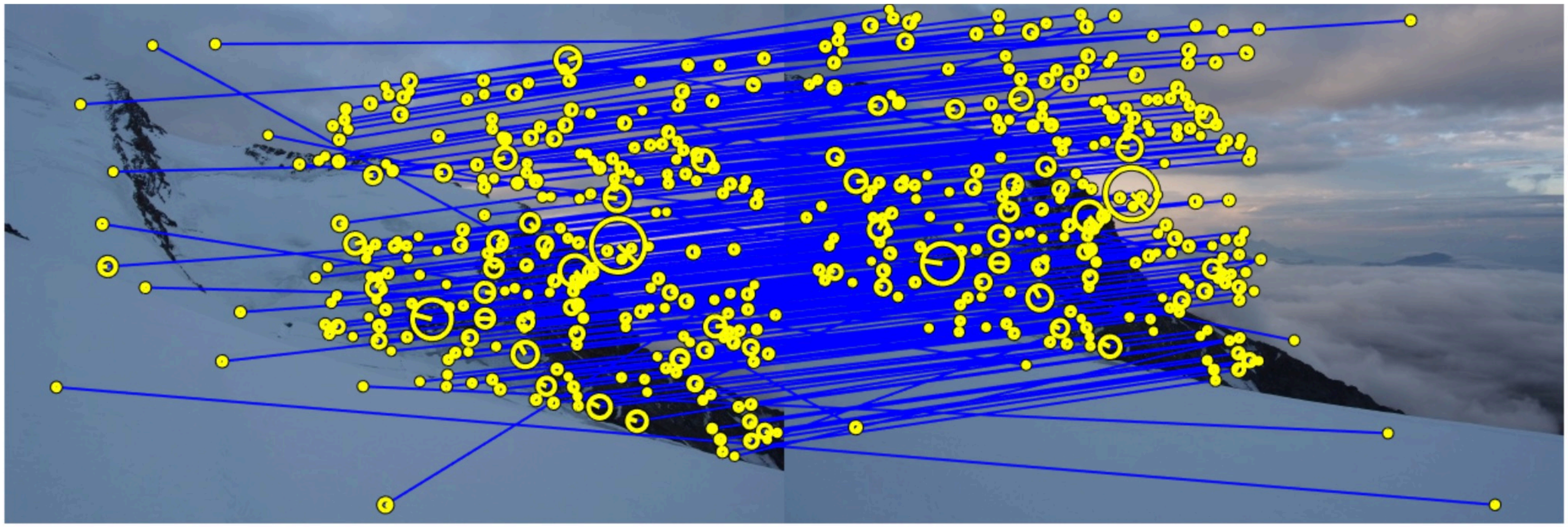
Selecting k, S, T

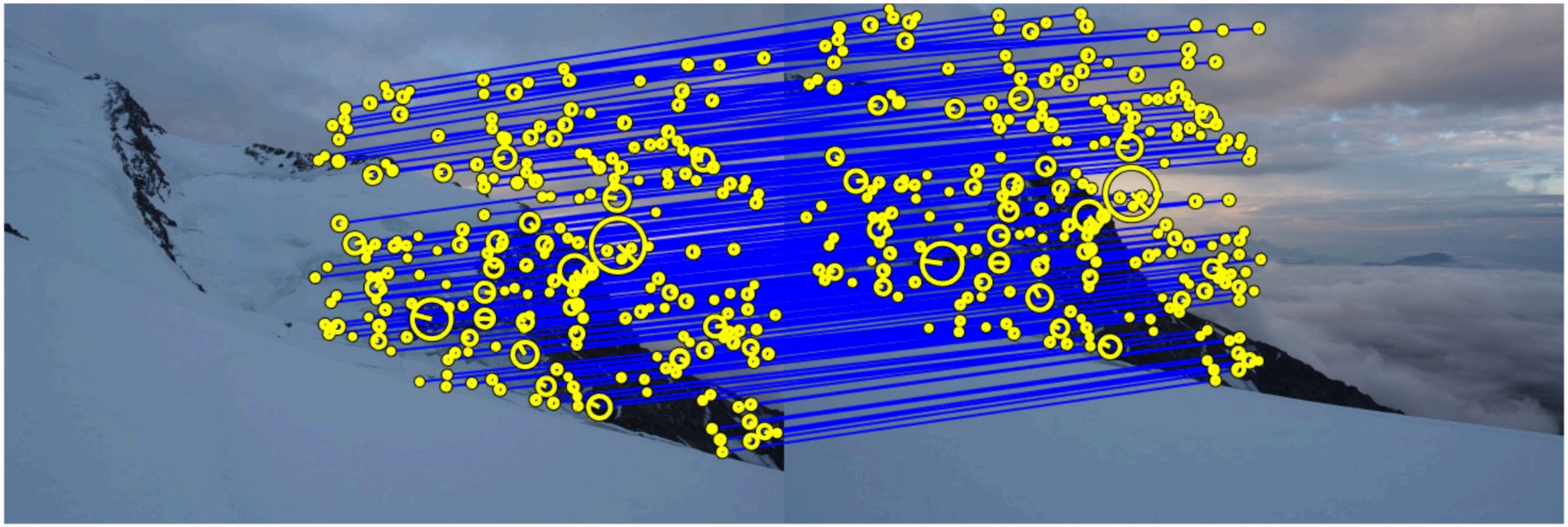
- To minimize S, we want to use smallest K
  - 2K = number of mapping parameters
  - K=3: affine mapping
  - K=4: homography
- How to determine q?
  - Among all matching pairs found based on SIFT descriptors, what percentages are true matching pairs?
- Number of trials to run?

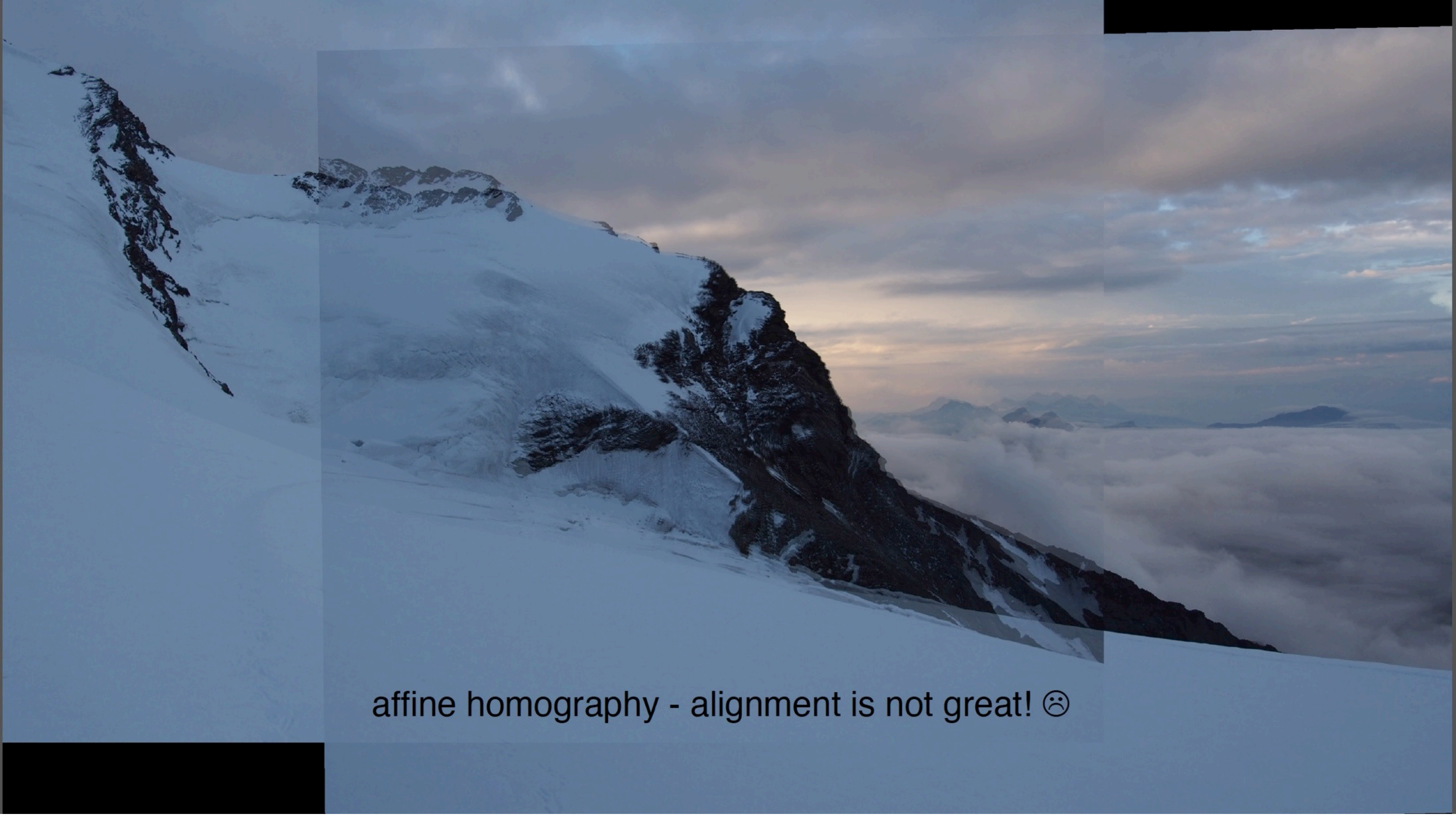
$$S = \frac{\log(1-P)}{\log\left(1-q^K\right)}$$



Correspondence based on SIFT descriptor matching. [courtesy B. Girod, (c) 2013 Stanford University]
From http://web.stanford.edu/class/ee368/Handouts/Lectures/2016_Autumn/16-Panoramas_16x9_big.pdf

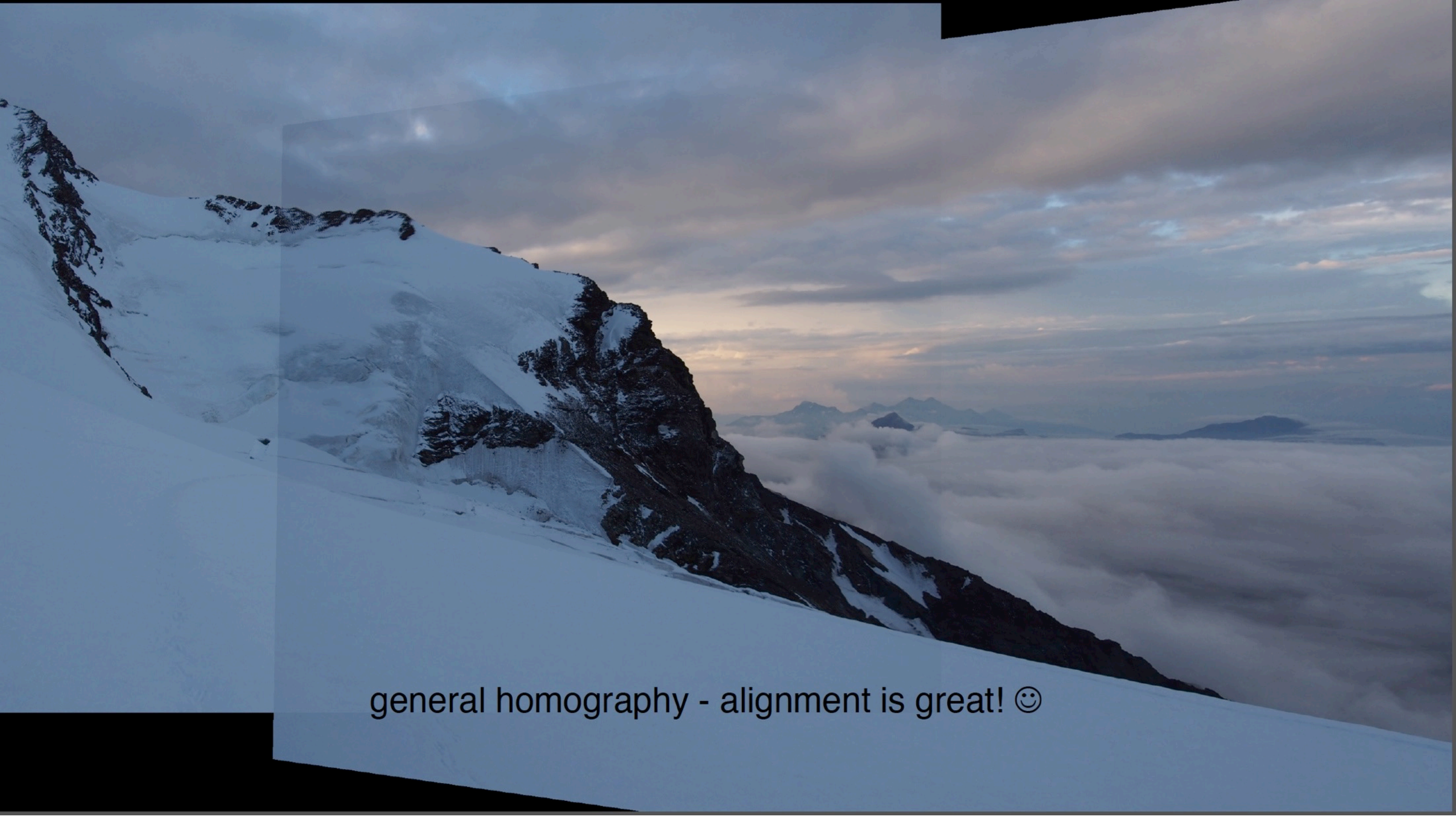

Correspondence based on SIFT descriptor matching.


Correspondence after RANSAC using homography model. [courtesy B. Girod, (c) 2013 Stanford University]



[courtesy B. Girod, (c) 2013 Stanford University]

http://web.stanford.edu/class/ee368/Handouts/Lectures/2016_Autumn/16-Panoramas_16x9_big.pdf



[courtesy B. Girod, (c) 2013 Stanford University]

http://web.stanford.edu/class/ee368/Handouts/Lectures/2016_Autumn/16-Panoramas_16x9_big.pdf

Pop Quiz

- What are problems with least squares?
- How does RANSAC work in principle?
- How many samples (pairs) to select in each trial to fit a model with N parameters?

Pop Quiz

- What are problems with least squares?
  - Not robust to outliers
- How does RANSAC work in principle?
  - Choose samples that minimize the number of outliers
- How many samples (pairs) to select in each trial to fit a model with N parameters?
  - K=N/2

Other ways to find the mapping that are less sensitive to the outliers

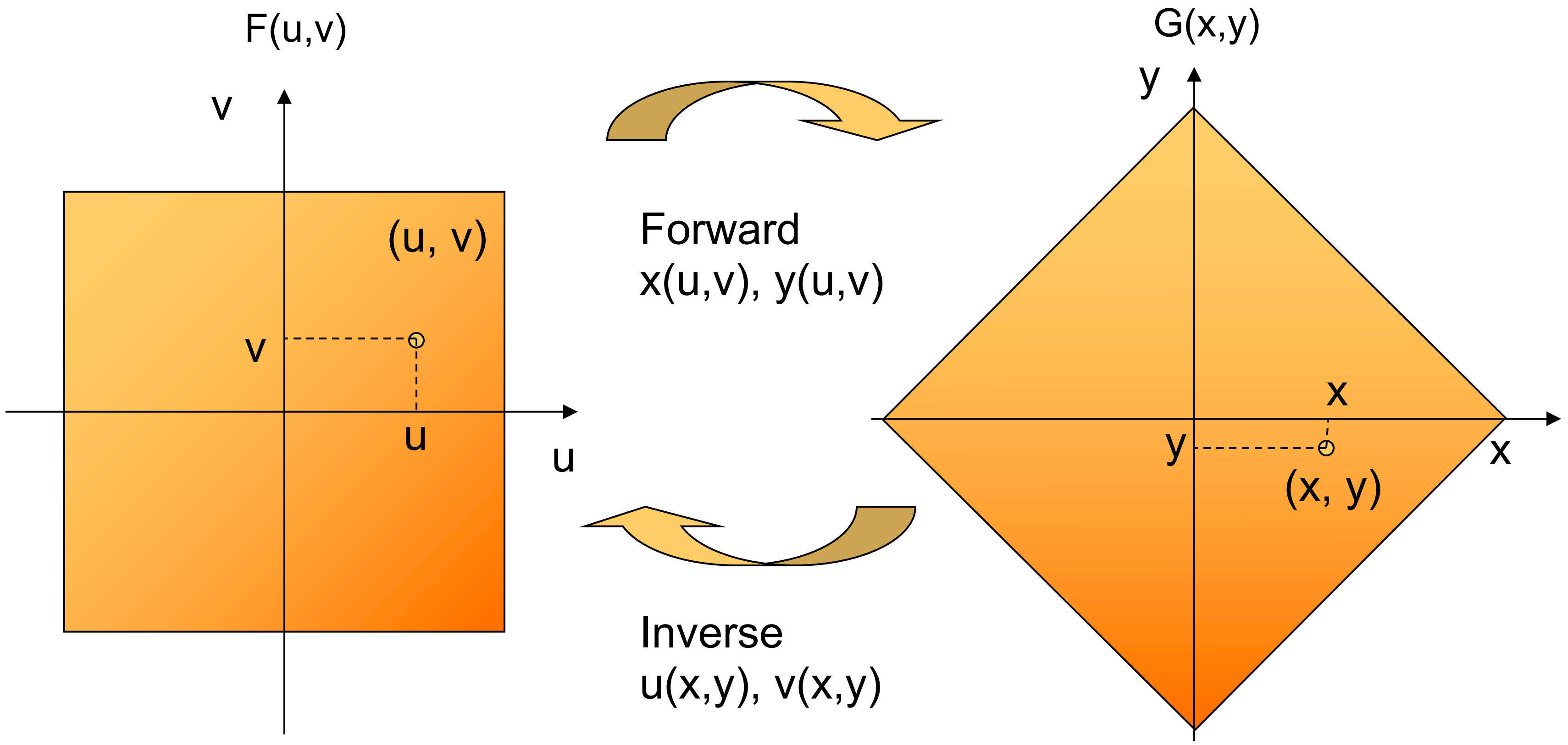
- Least squares method: $E(\mathbf{a}) = \sum_k (A_n\mathbf{a} - x_n)^2 = \|\mathbf{A}\mathbf{a} - \mathbf{x}\|^2 -> \min$
  - Error at an outlier is squared
- To minimize the impact of the outliner,
  - Don't square the error
  - Use L1 norm instead

$$E(\mathbf{a}) = \sum_k |A_n\mathbf{a} - x_n| = \|\mathbf{A}\mathbf{a} - \mathbf{x}\|_1 -> \min$$

Stitching of Two Images

- Given two images F(u,v) and G(x,y)
- Pick one image F as the reference
- Identify feature points in each image
- Find corresponding features between two images based on feature descriptors (initial candidate matchings)
- Derive the geometric transformation between G and F based on the correspondence (RANSAC)
  - Transformation model depends on the expected /observed view difference between two images
  - Homography is most often used
- Warp G to G' so that G' aligns with F (How?)
- Blend F and G' (How?)

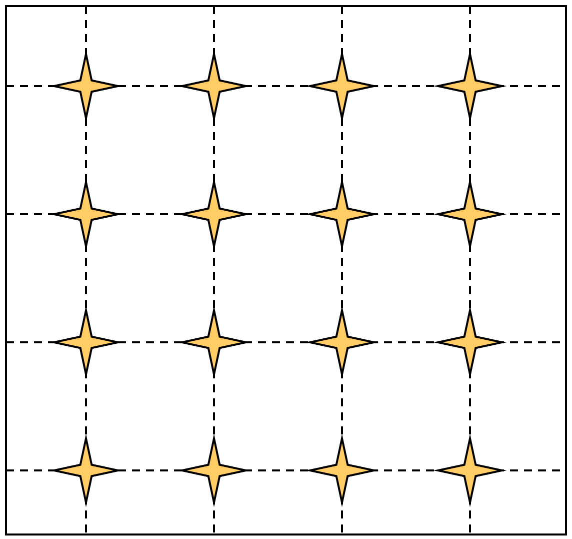
How to Warp One Image to Another?

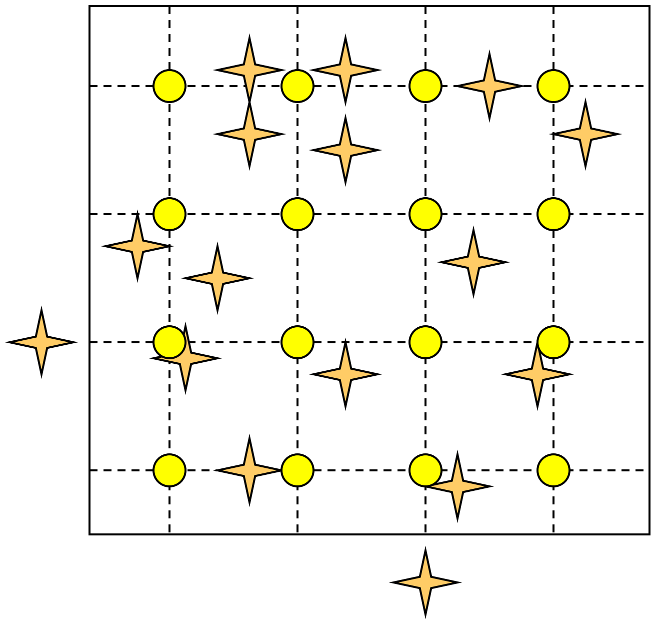


Inverse Mapping: G(x,y)=F(u(x,y), v(x,y)) or G(**x**)=F(**u**(**x**))

Image Warping by Forward Mapping

- Mapping image f(u, v) to g(x, y) based on a given mapping function: x(u, v), y(u, v).
- Forward Mapping
  - For each point (u, v) in the original image, find the corresponding position (x, y) in the deformed image by the forward mapping function, and let g(x,y)=f(u,v).
  - What if the mapped position (x,y) is not an integer sample in the desired image?



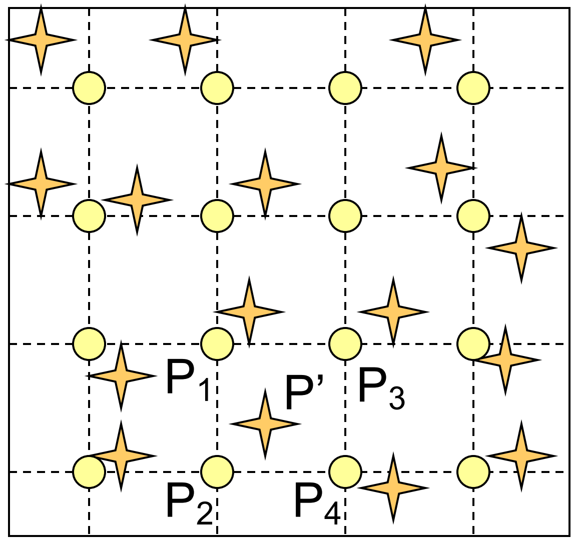


Warping points are often non-integer samples

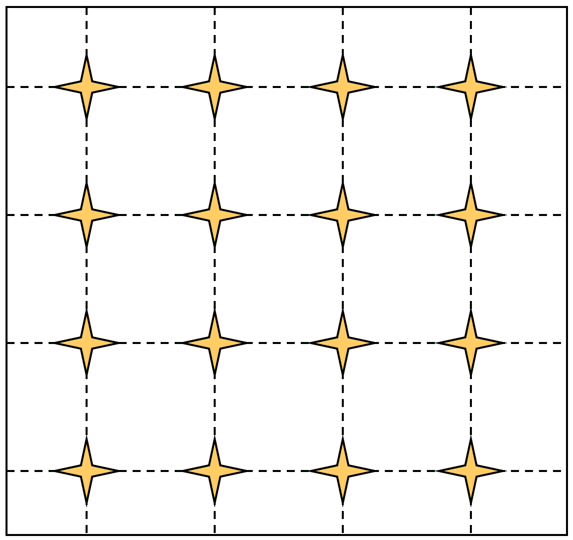
Many integer samples "o" are not assigned Values

Image Warping by Inverse Mapping

- For each point (x, y) in the image to be obtained, find its corresponding point (u, v) in the original image using the inverse mapping function, and let g(x, y) = f(u, v).
- What if the mapped point (u,v) is not an integer sample?
  - Interpolate from nearby integer samples!



P' will be interpolated from P_1, P_2, P_3, and P_4



Interpolation Method

- Nearest neighbor:
  - Round (u,v) to the nearest integer samples
- Bilinear interpolation:
  - Find four integer samples nearest to (u,v), apply bilinear interpolation
- Other higher order interpolation methods can also be used
  - Require more than 4 nearest integer samples!

Related Python Functions

- For Affine transform
  - M = cv2.getAffineTransform(srcPts, dstPts)
  - result = cv2.warpAffine(srcImg, M, dsize, flag)
- For perspective transform (homography)
  - M = cv2.getPerspectiveTransform(srcPts, dstPts)
    - %determine the transform using >=4 pairs of points
  - M= cv.FindHomography(srcPts, dstPts, method,ransacReprojThreshold, mask)
    - % allow the use of RANSAC
  - result = cv2.warpPerspective(srcImg, M, dsize, flag)
    - Flag allow selection of interpolation methods

- https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html
-

MATLAB function: interp2

- ZI = INTERP2(X,Y,Z,XI,YI, METHOD) interpolate known values on Z at the points defined by matrices X and Y, to find ZI, the values of the underlying 2-D function Z at the points defined by matrices XI and YI.
  - Matrices X and Y specify the points at which the data Z is given.
  - METHOD specifies interpolation filter
    - 'nearest' - nearest neighbor interpolation
    - 'linear' - bilinear interpolation
    - 'spline' - spline interpolation
    - 'cubic' - bicubic interpolation as long as the data is uniformly spaced, otherwise the same as 'spline'

Using ‘interp2’ to realize image warping

- Use inverse mapping
- Step 1: For all possible pixels in output image (x,y), find corresponding points in the input image (u,v)
    - (X,Y)=meshgrid(1:M,1:N)
    - Apply inverse mapping function to find corresponding (u,v), for every (x,y), store in (UI,VI)
        - Can use tforminv( ) function if you derived the transformation using maketform().
        - Or write your own code using the specified mapping
- Step 2: Use interp2 to interpret the value of the input image at (UI,VI) from their values at regularly sampled points (X,Y)
    - Outimg=interp2(X,Y,inimg,UI,VI,’linear’);

MATLAB Built-in Function for Warping

- B = IMTRANSFORM(A,TFORM, INTERP) transforms the image A according to the 2-D spatial transformation defined by TFORMB; INTERP specifies the interpolation filter
- Example 1
-     ---------
-     Apply a horizontal shear to an intensity image.
-
-         I = imread('cameraman.tif');
-         tform = maketform('affine',[1 0 0; .5 1 0; 0 0 1]);
-         J = imtransform(I,tform);
-         figure, imshow(I), figure, imshow(J)

Stitching of Two Images

- Given two images F(u,v) and G(x,y)
- Pick one image F as the reference
- Identify feature points in each image
- Find corresponding features between two images based on feature descriptors
- Derive the geometric transformation between G and F based on the correspondence (RANSAC)
  - Transformation model depends on the expected /observed view difference between two images
  - Homography is most often used
- Warp G to F -> G'
- Blend F and G'

Should we find mapping from F to G, or G to F?

We want to warp G to align with F!
To use inverse warping, for every pixel (x,y) in F, find their location (u,v) in G:
$u=h_x(x,y)$, $v=h_y(x,y)$, $G'(x,y)=G(h_x(x,y), h_y(x,y))$

Pop Quiz

- Why is inverse mapping better for image warping?
- Why is interpolation needed?

Pop Quiz

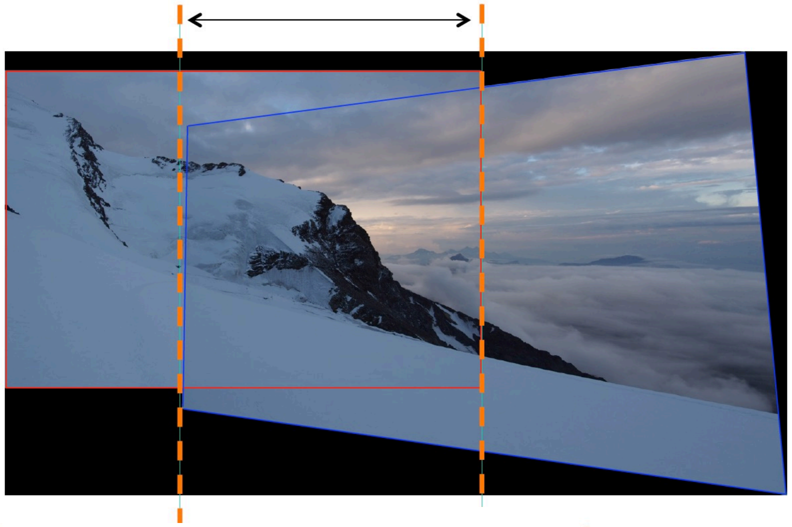
- Why is inverse mapping better for image warping?
  - To avoid holes
  - To avoid multiple pixels warped to the same pixel

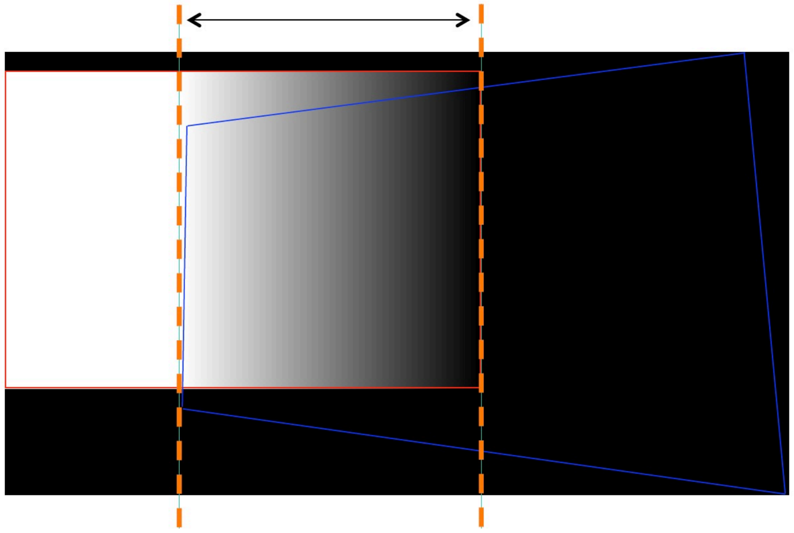
- Why is interpolation needed?
  - The corresponding position may not have integer coordinate
  - Need to interpolate from nearby pixels with integer coordinates

Image Blending

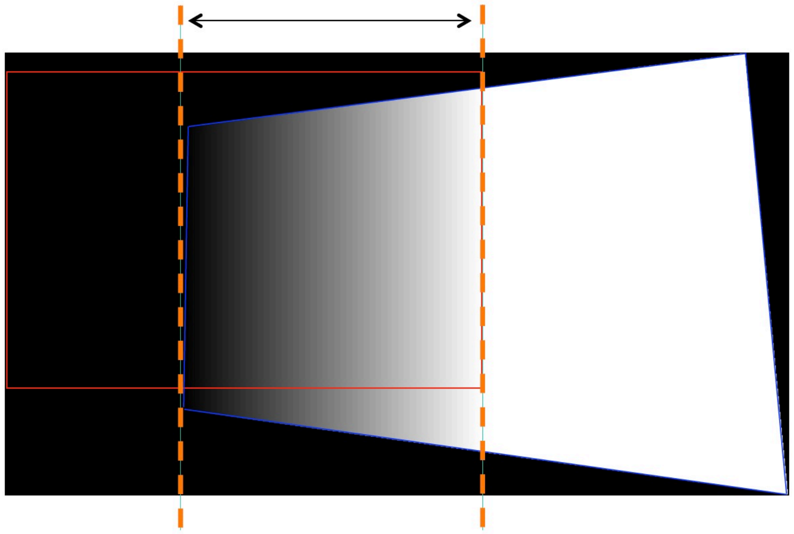
- Need to determine the size of the stitched image
  - width = 2*maximum width of F,G'; height=2*maximum of height of F,G'
- Need to find area of overlapping between F and G'
- How to blend in areas of overlap?
  - Simple approach
    - Use reference image value
    - Use average of the two images
    - Can lead to visible boundary
  - Distance based weighting (still not good enough)
  - Blending using pyramid representation!
- What if there are contrast difference between F and G?
  - Gain normalization

Linear Blending





Weight map for left image



Weight map for right image

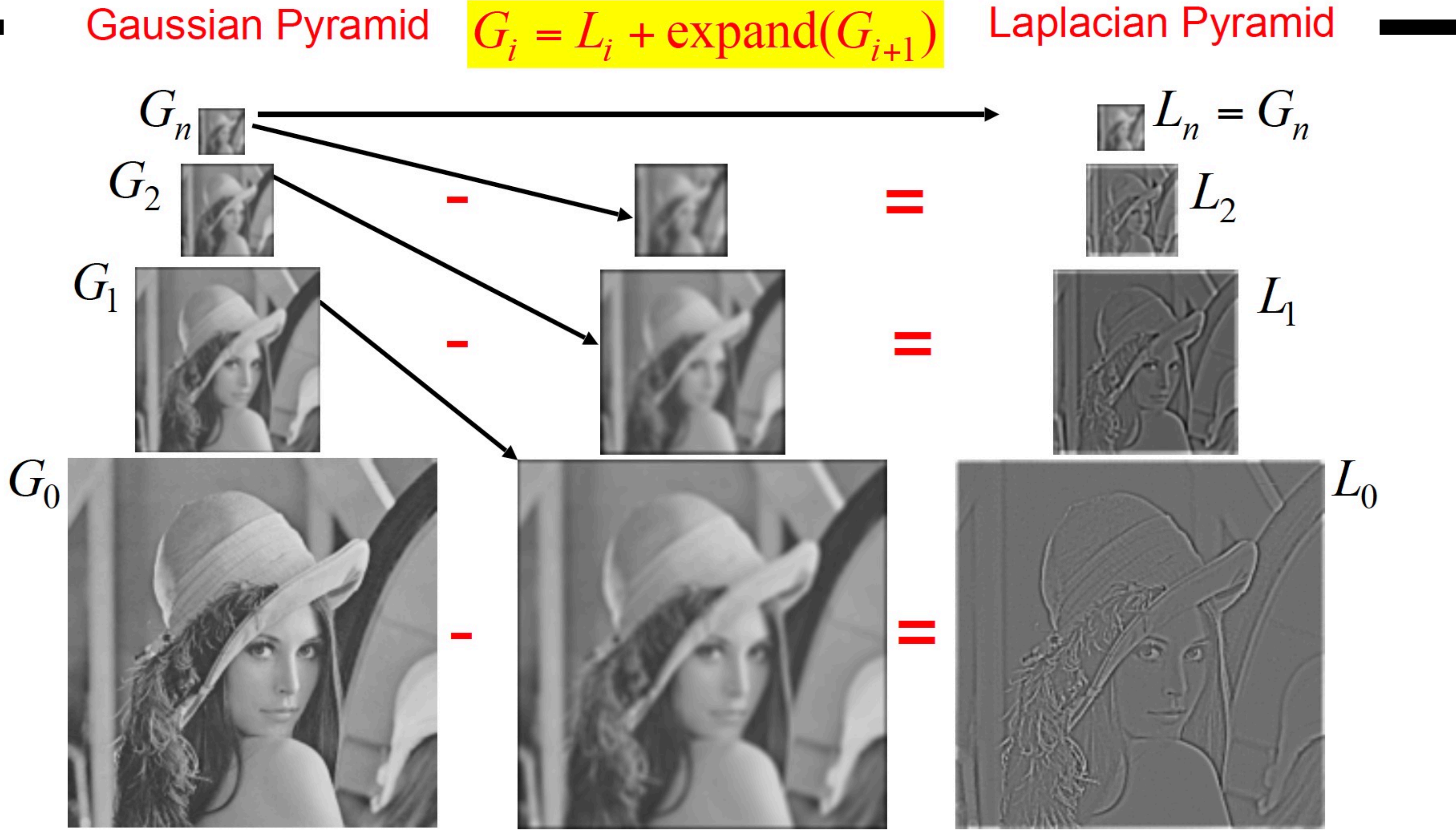
[courtesy B. Girod, (c) 2013 Stanford University]

From http://web.stanford.edu/class/ee368/Handouts/Lectures/2016_Autumn/16-Panoramas_16x9_big.pdf

The Laplacian Pyramid

$$L_i = G_i - \text{expand}(G_{i+1})$$

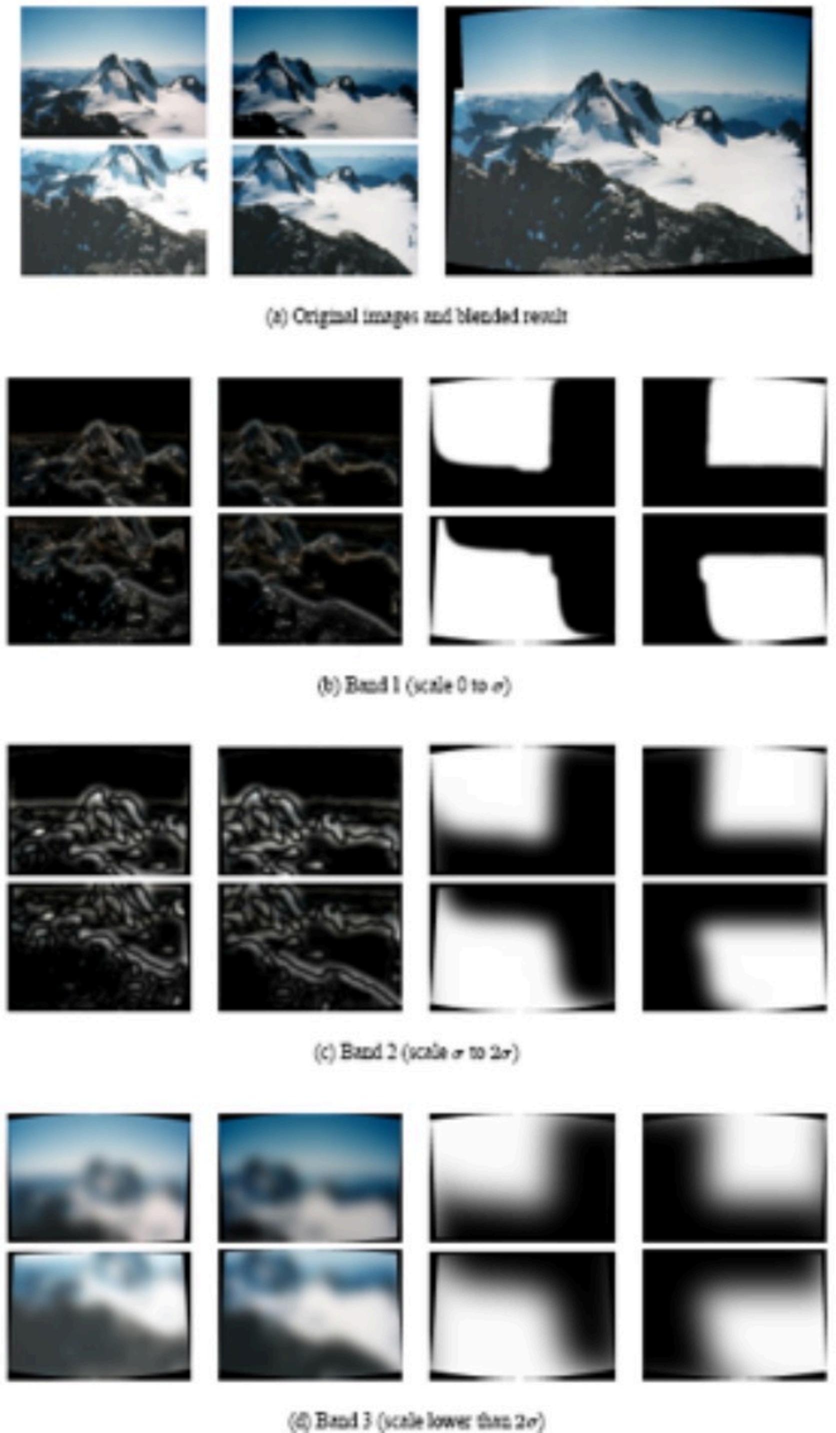
$$G_i = L_i + \text{expand}(G_{i+1})$$



From: http://courses.cs.washington.edu/courses/cse576/16sp/Slides/10_ImageStitching.pdf

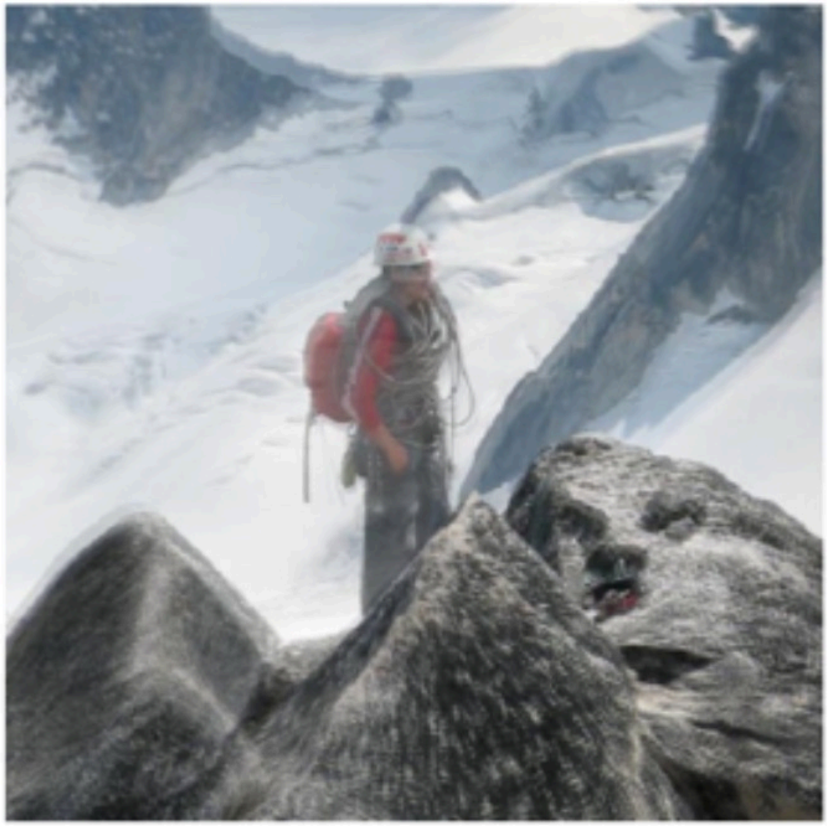
Blending Using Pyramid

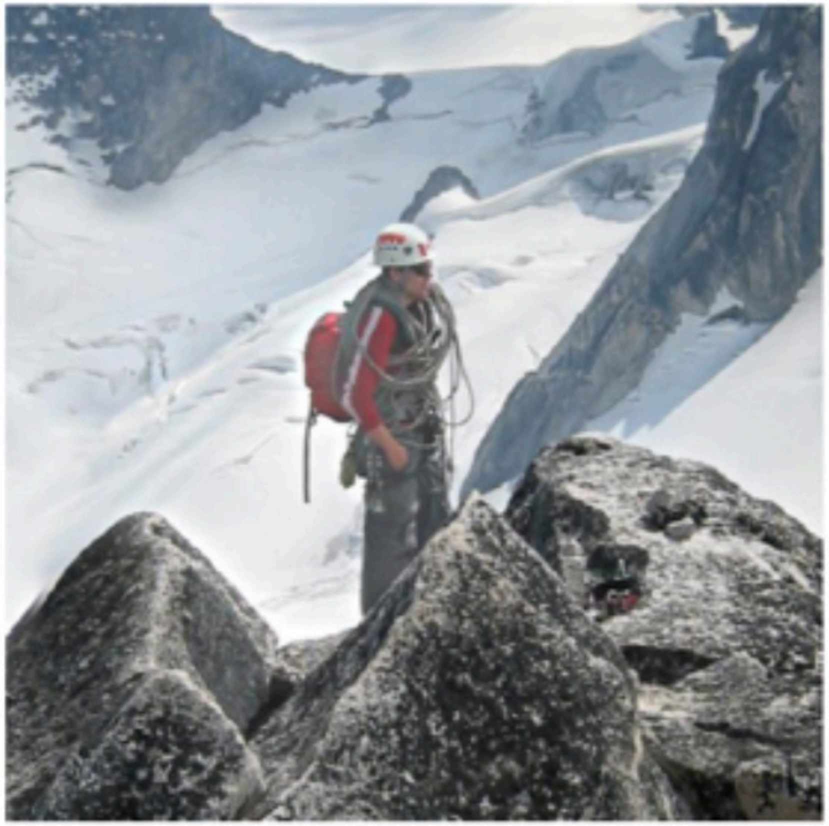
- Based on overlap between each image and the target mosaic image, create a mask image
- Create Laplacian pyramid of each image, and Gaussian pyramid of its mask image
- Averaging Laplacian images at each pyramid level, using Gaussian images of the masks as weights (normalized)
- Reconstruct the blended images from the blended Laplacian pyramid


(a) Original images and blended result
(b) Band 1 (scale 0 to σ)
(c) Band 2 (scale σ to 2σ)
(d) Band 3 (scale lower than 2σ)

From: http://courses.cs.washington.edu/courses/cse576/16sp/Slides/10_ImageStitching.pdf

Blending comparison (IJCV 2007)

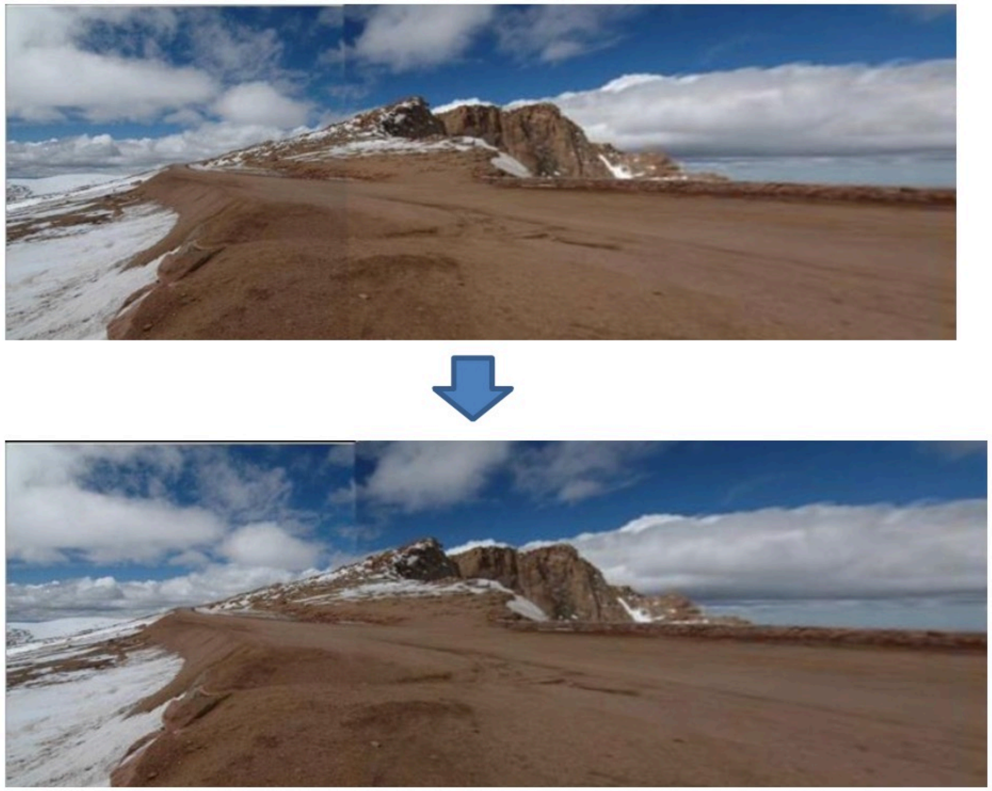

(a) Linear blending

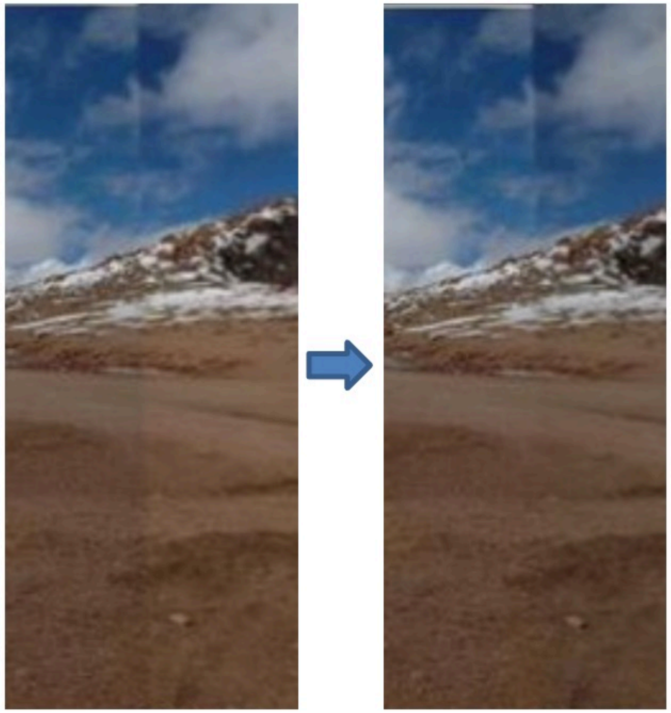

(b) Multi-band blending

From: http://courses.cs.washington.edu/courses/cse576/16sp/Slides/10_ImageStitching.pdf

Gain compensation

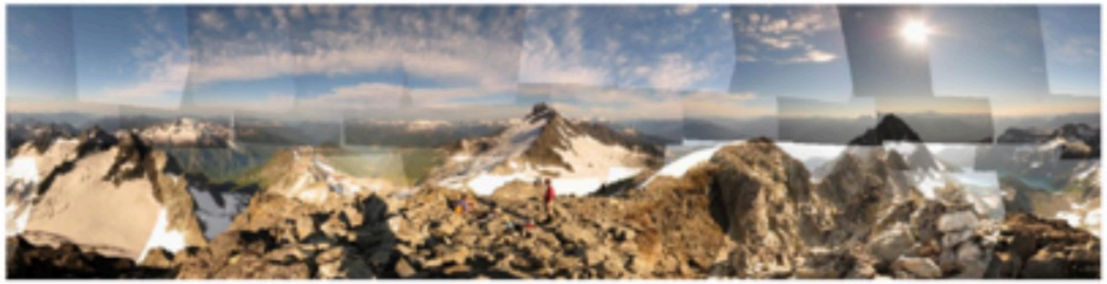
- Simple gain adjustment
  - Compute average RGB intensity of each image in overlapping region
  - Normalize intensities by ratio of averages



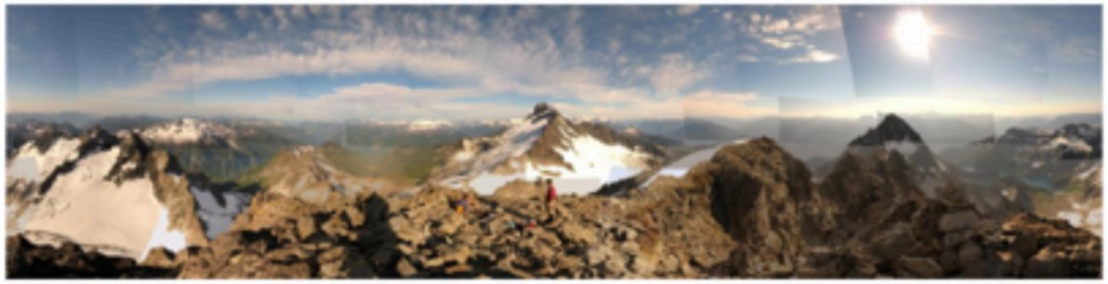


From: http://courses.cs.washington.edu/courses/cse576/16sp/Slides/10_ImageStitching.pdf

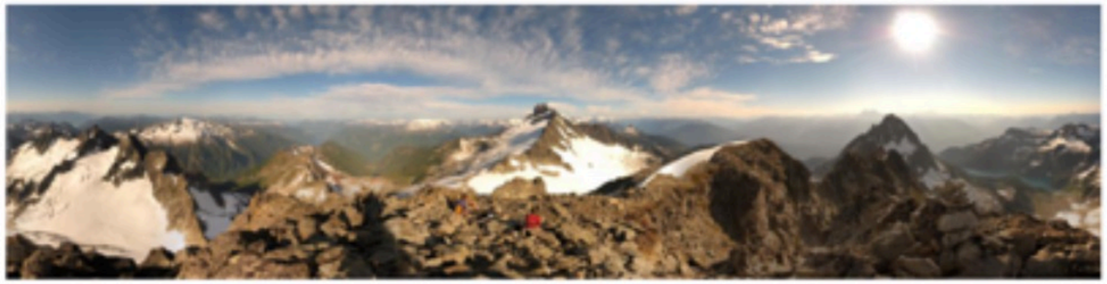
Blending Comparison



(b) Without gain compensation



(c) With gain compensation



(d) With gain compensation and multi-band blending

From: http://courses.cs.washington.edu/courses/cse576/16sp/Slides/10_ImageStitching.pdf

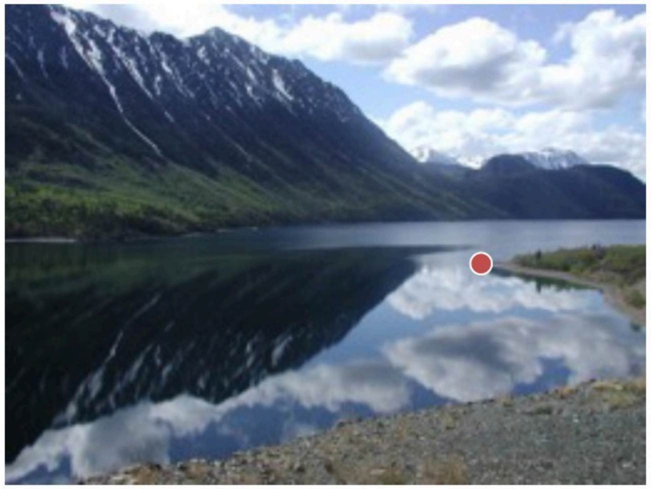
How to Stitch Multiple Images?

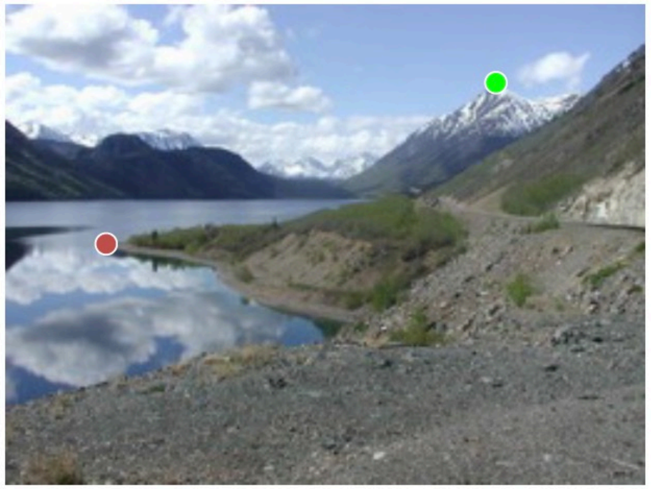
- Pick one as reference
- Pick an image nearest to the reference, warp to the reference, stitch the two images
- Use the above stitched image as new reference, repeat for the next image
- May not work well when cameras have significant rotations in between

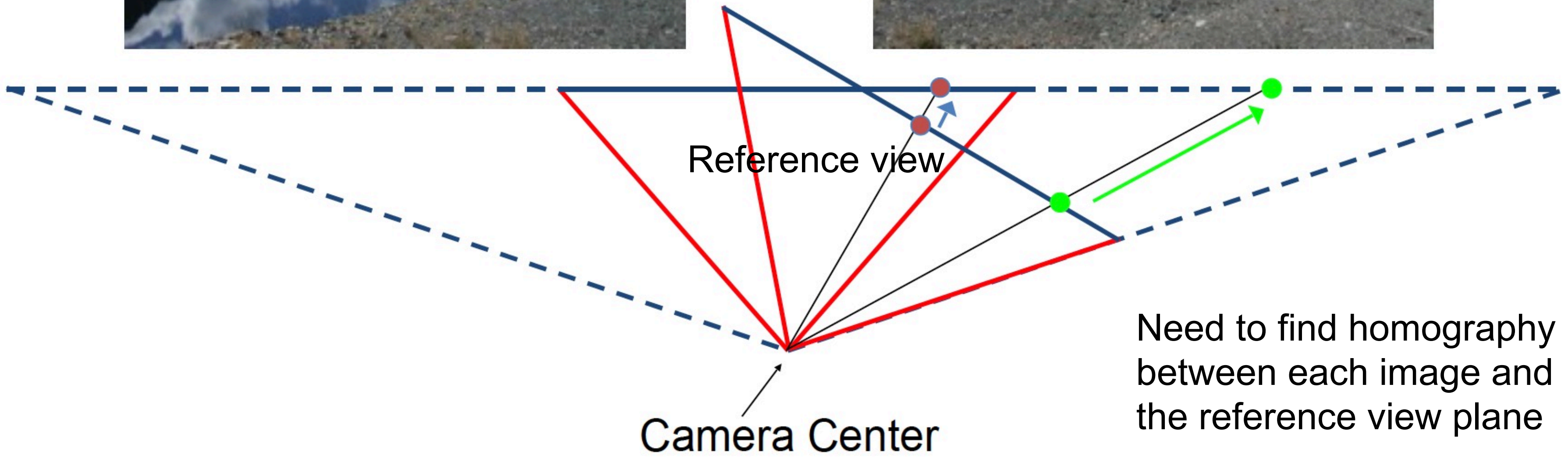
Pop Quiz

- How do you determine the size of the stitched image?
- How do you determine the overlap region?
- How do you do pyramid blending?
- Why is gain compensation needed?

Reprojection for Planar Panorama





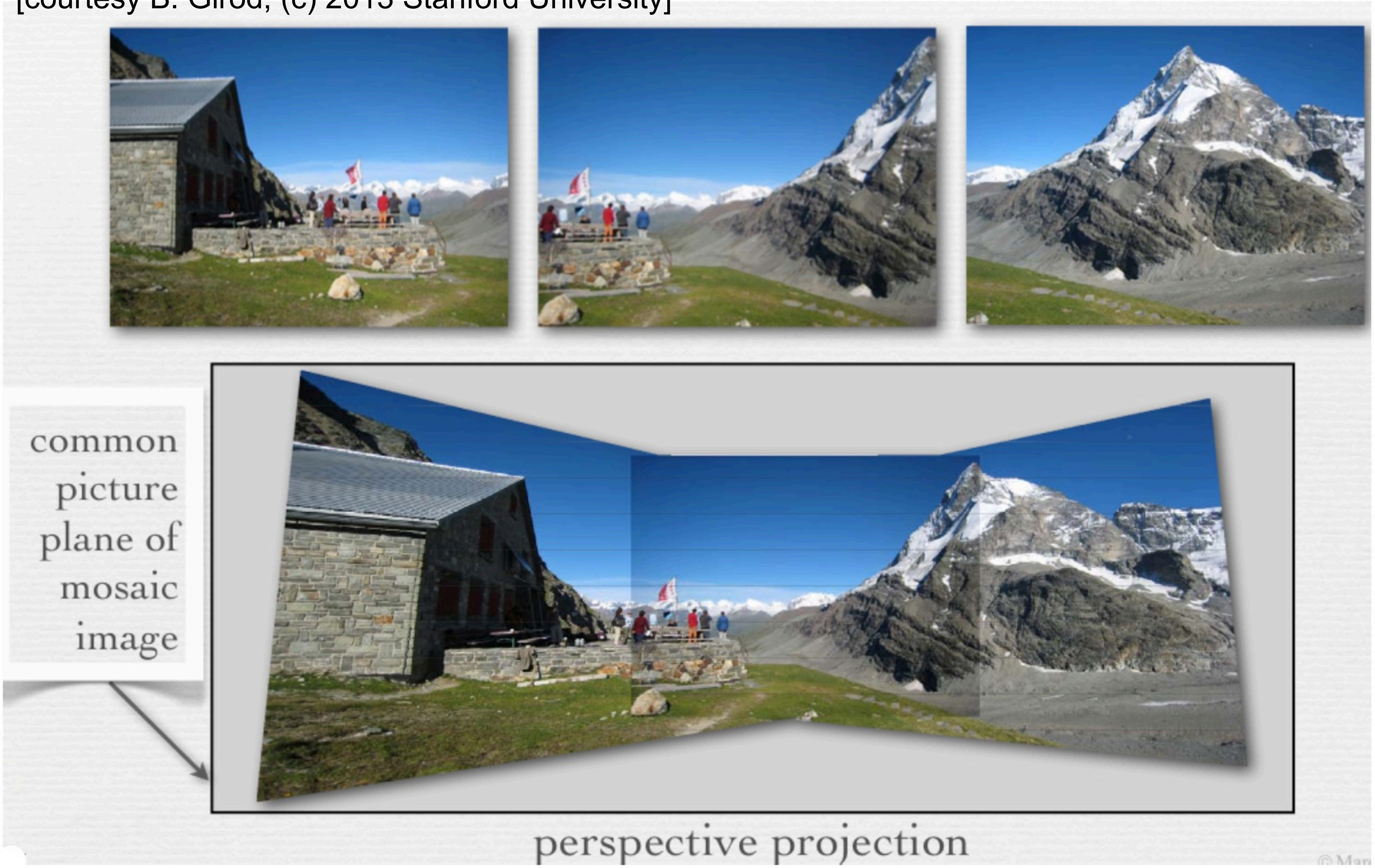


Need to find homography between each image and the reference view plane

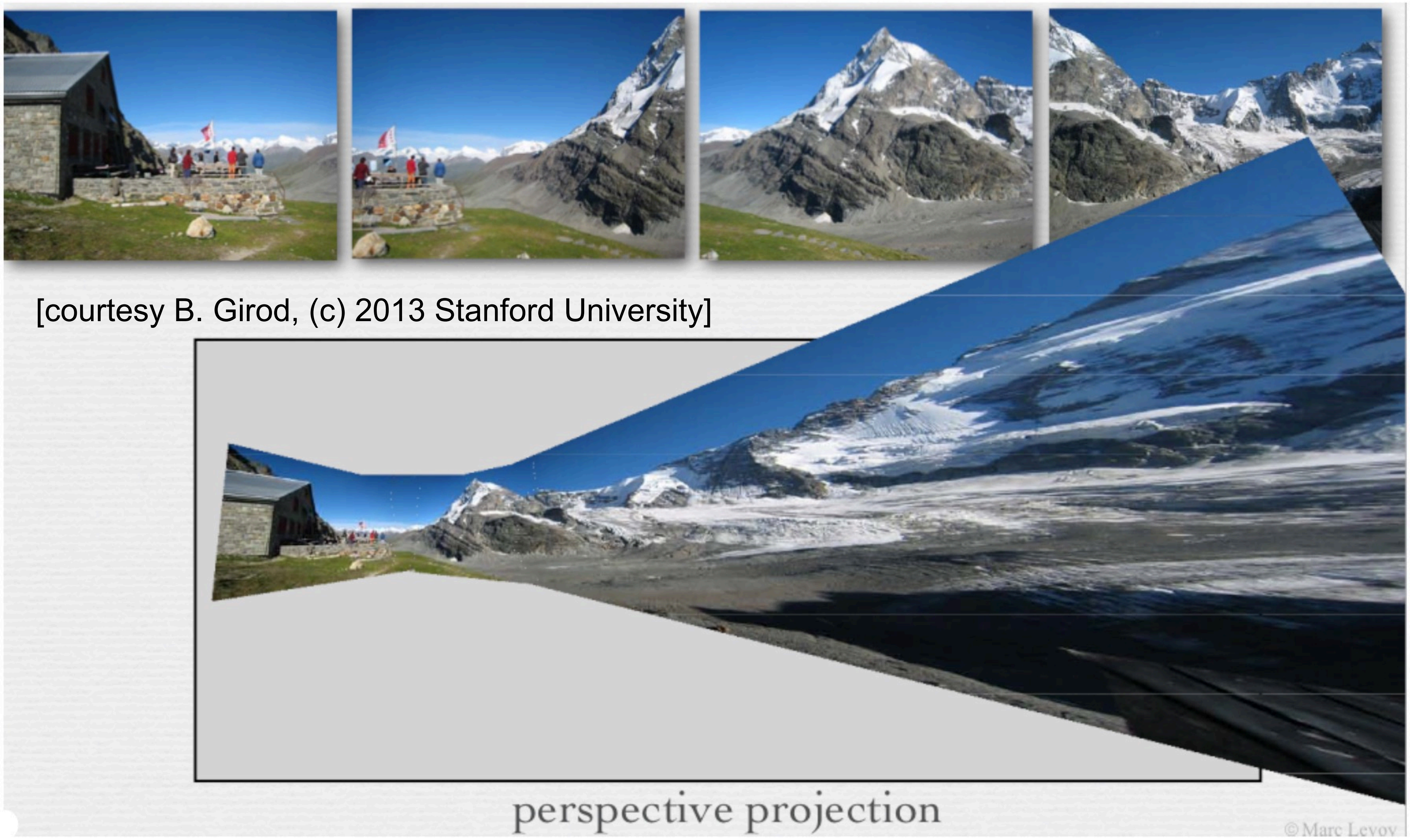
From: http://courses.cs.washington.edu/courses/cse576/16sp/Slides/10_ImageStitching.pdf

Example: Stitching 3 Images

[courtesy B. Girod, (c) 2013 Stanford University]



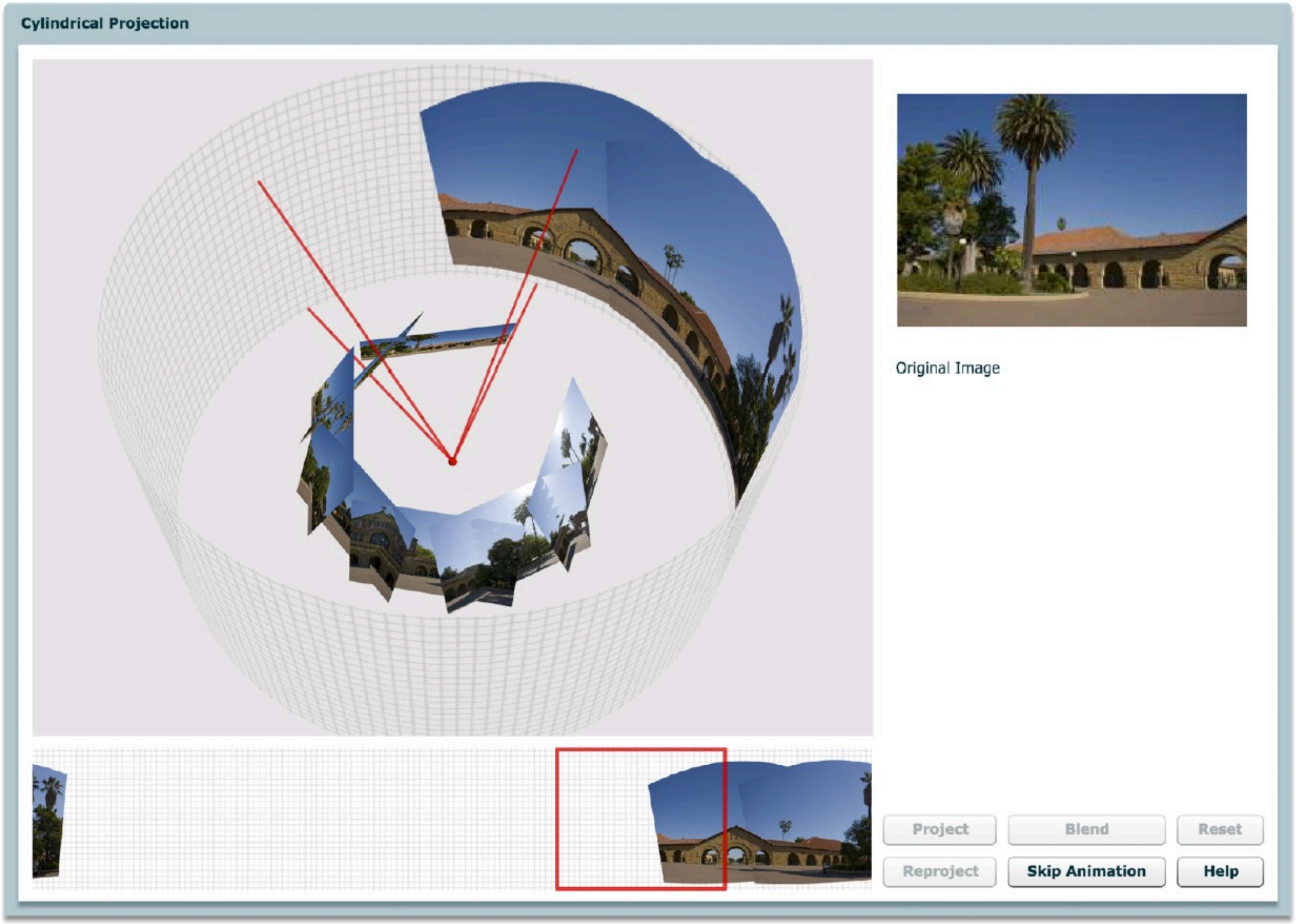
Example: Stitching 4 Images



[courtesy B. Girod, (c) 2013 Stanford University]

How to avoid this stretching? Don’t restrict project surface to a plane!

Cylindrical Panorama (not required)

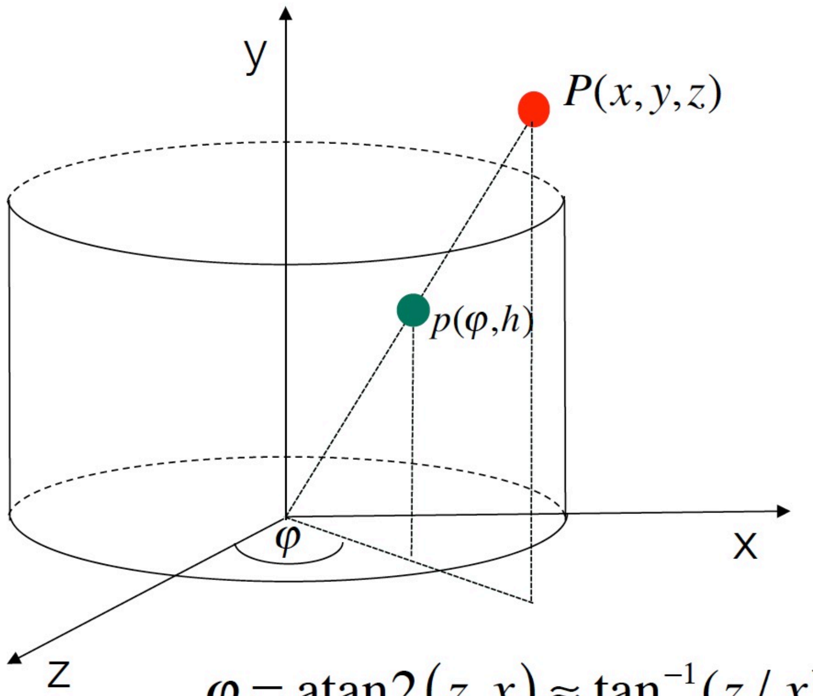


https://graphics.stanford.edu/courses/cs178/applets/projection.html

[courtesy B. Girod, (c) 2013 Stanford University]

Cylindrical Panorama

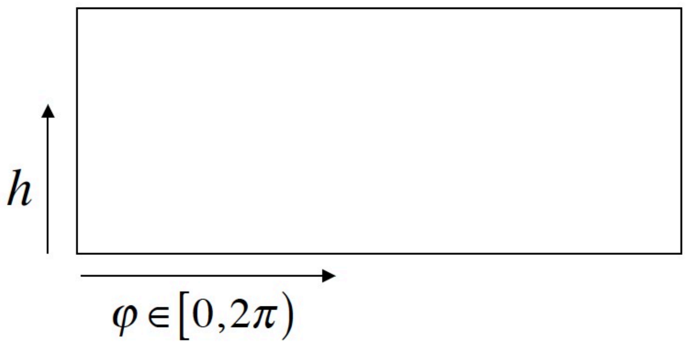
Assume cylinder of unit radius



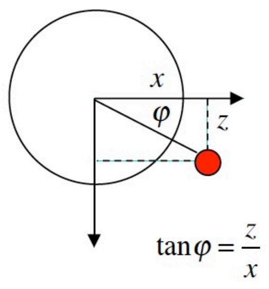
$$\varphi = \operatorname{atan2}(z,x) \approx \tan^{-1}(z/x)$$

$$h = y/\sqrt{x^2+z^2}$$

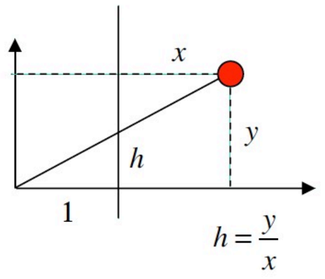
Unwrapped Cylinder



top view



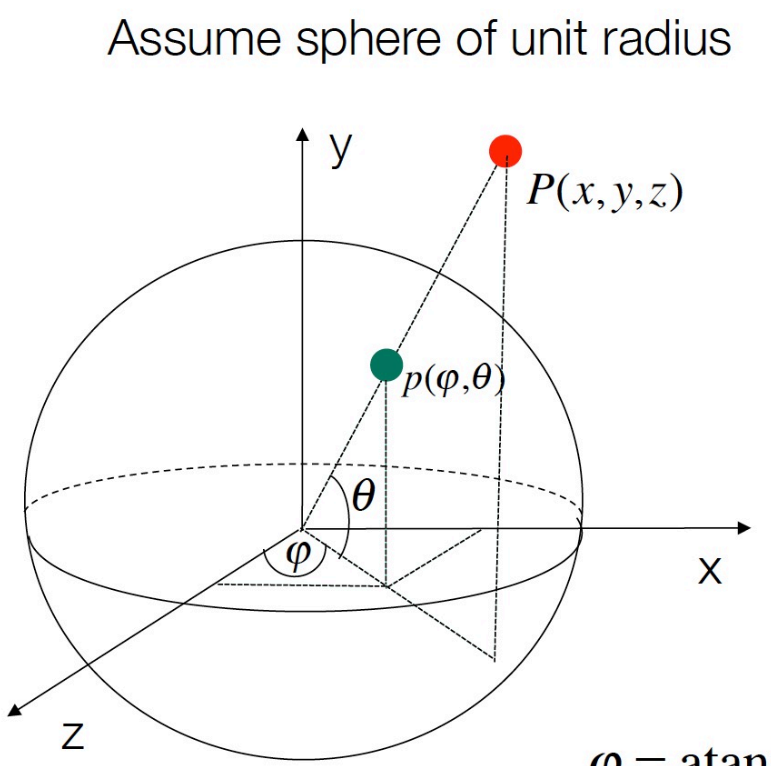
side view (1D)

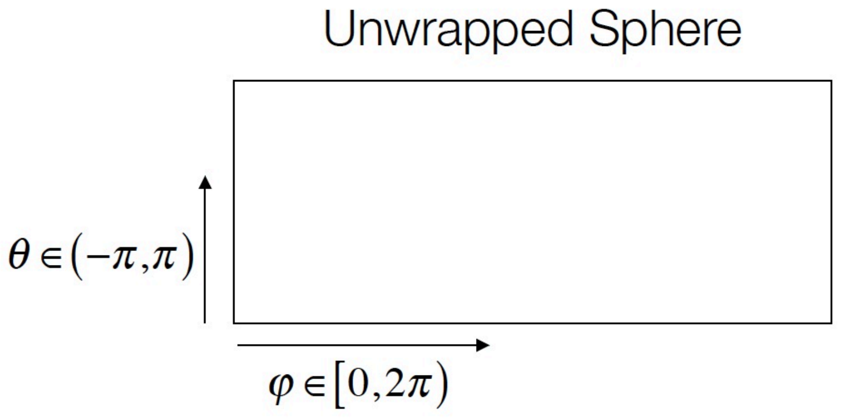


[courtesy B. Girod, (c) 2013 Stanford University]

P(x,y.z) derived from known camera geometry: pixel (x,y) is projected to (x,y,z) for an assumed z for one camera position, other camera positions have known 3D rotations and shifts with the initial camera position, so that (x,y) is projected to [R][x,y,z]^T+t.

Spherical Panorama (not required)





$$\varphi = \operatorname{atan2}(z,x)$$
$$\theta = \operatorname{atan2}(y,x)$$

[courtesy B. Girod, (c) 2013 Stanford University]

Use multiple cameras looking into different positions to cover the entire sphere. Each camera view produces images with a particular set of points (x,y,z) determined from their image coordinates.

Some 360 Cameras

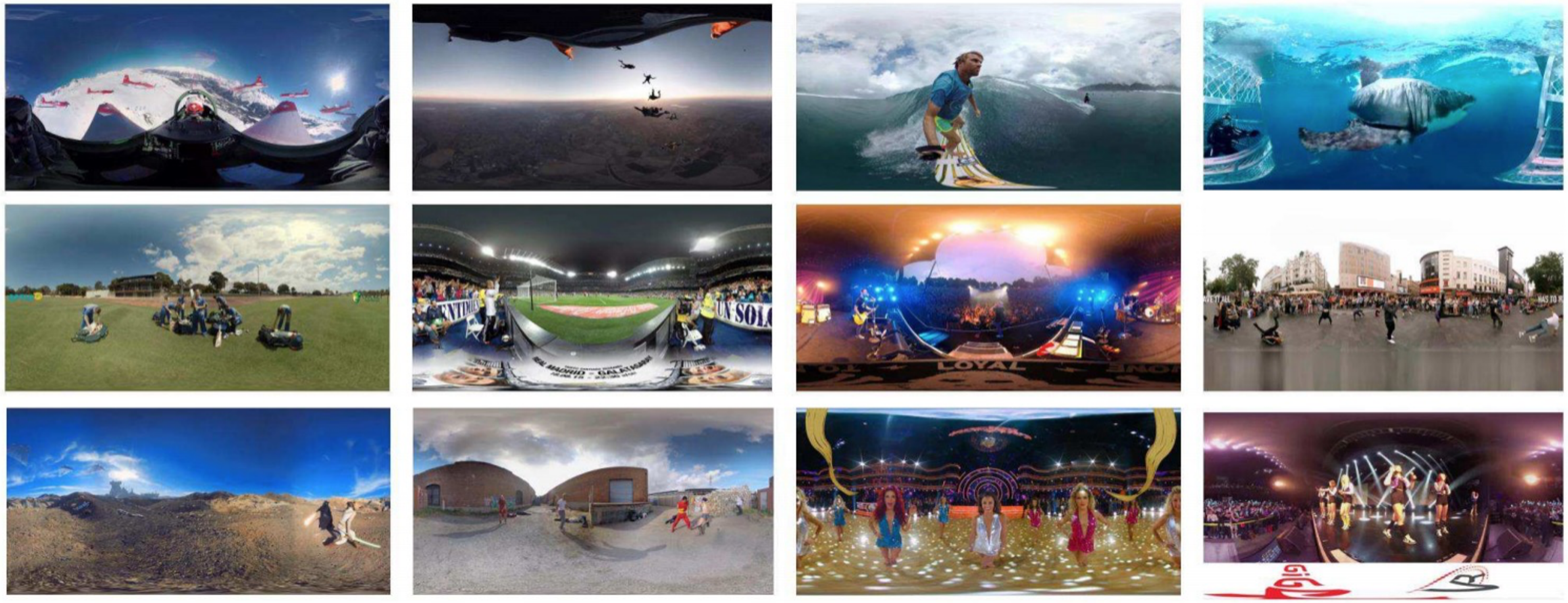
Go pro omni

Facebook surround 360

Nokia Ozo

Samsung Gear 360

Example of 360^o Videos (equirectangular format)



Example video: https://www.youtube.com/watch?v=EbUHKw8r5xA

Summary

- Relation between images from different camera view points
  - Geometric mapping: planar homography vs affine mapping
  - Intensity (color) mapping
- How to determine the geometric mapping parameters
  - Feature correspondence based on local descriptors (SIFT)
  - Regression of mapping based on feature correspondences
    - Least squares
    - RANSAC (remove outliers from the initial feature correspondence)
- Image warping
  - Forward vs. inverse mapping
- Image blending
  - Pyramid blending, gain compensation
- Panoramic view stitching
  - Stitching onto one planar surface, cylindrical surface (not required), or a spherical surface (not required)

Reading Assignment

- [Szeliski2021] Richard Szeliski, Computer Vision: Algorithms and Applications. 2021. Sec. 2.1, 3.5.5, 3.6.1, 7.1.3, 8.1, 8.2

- Optional:
- R. Szeliski, Image alignment and stitching: A tutorial, Journal of Foundations and Trends in Computer Graphics and Vision, Volume 2 Issue 1, January 2006, Pages 1 – 104, http://research.microsoft.com/pubs/75695/Szeliski-FnT06.pdf
- R. Szeliski and H. Y. Shum, "**Creating full view panoramic image mosaics and environment maps", SIGGRAPH**'97
- Hartley, Richard, and Andrew Zisserman. Multiple view geometry in computer vision. Cambridge university press, 2003.
- Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM , 24(6):381–395.
- Stewart, C. V. (1999). Robust parameter estimation in computer vision. SIAM Reviews ,41(3):513–537.

Written Homework

- Show that the image coordinates of the same 3D point taken by two cameras with the same focus related by a rotation and translation can be described by

$$x_1 = F\frac{(r_1x_0 + r_2y_0 + r_3F)Z + T_xF}{(r_7x_0 + r_8y_0 + r_9F)Z + T_zF}, y_1 = F\frac{(r_4x_0 + r_5y_0 + r_6F)Z + T_yF}{(r_7x_0 + r_8y_0 + r_9F)Z + T_zF}$$

- Show that for 3D points on the same plane, the above mapping function reduces to the planar homography.
- Suppose two images are related by a planar homography. Describe the steps needed to determine the homography parameters. Call this Algorithm 1.
- Suppose you want to generate a panoramic image from three images taken by panning a camera. Describes the steps needed to generate the image. You can call Algorithm 1.

MATLAB Exercises (Optional)

- Suppose you were given N pairs of corresponding feature points in two images, with image coordinates given as (u_k,v_k) and (x_k,y_k),k=1,2,…,N. You want to approximate the geometric mapping between the two image using an affine mapping. Write a MATLAB code to solve the problem when you have only N=3 pairs of matching points, and when you have N>3 with least squares fitting.
- Repeat above for using planar homography model. Write a MATLAB code to solve the problem when you have only N=4 pairs of matching points, and when you have N>4 with least squares fitting. Use the DLT formulation.
- Write a matlab program that implements the following steps:i) Read in two images that are taken of the same scene but at slightly different angles; ii) select corresponding feature points (more than 3) in these two images (You can use cpselect() function; iii) determine the affine transform between the two images; iii) apply affine transform to one image so that it is aligned with the other image. Please note that you should not use the 'cp2tform' or 'imtransform()' function in MATLAB. You should write your own program, which can call "interp2()". Compare your results with that obtained using the 'cp2tform' and 'imtransform()'.