Image stitching

Digital Visual Effects

Yung-Yu Chuang

with slides by Richard Szeliski, Steve Seitz, Matthew Brown and Vaclav Hlavac

Image stitching

- Stitching = alignment + blending

geometrical registration

photometric registration

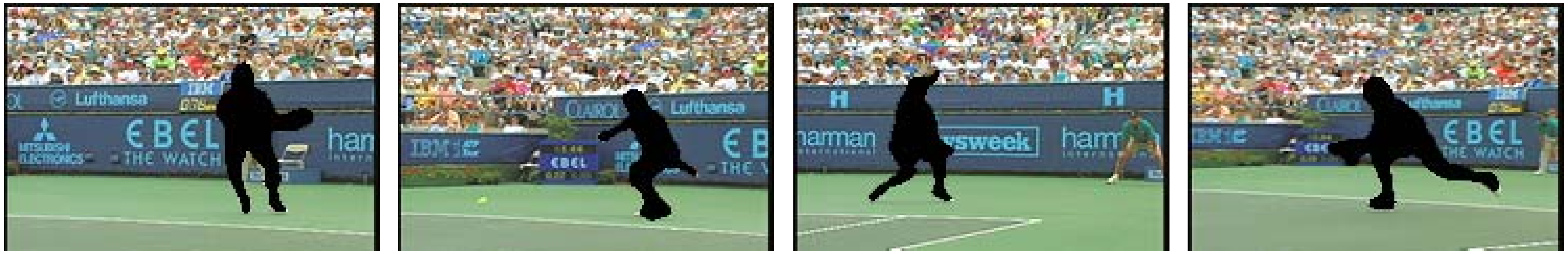
Applications of image stitching

- Video stabilization
- Video summarization
- Video compression
- Video matting
- Panorama creation

Video summarization

DigiVFX

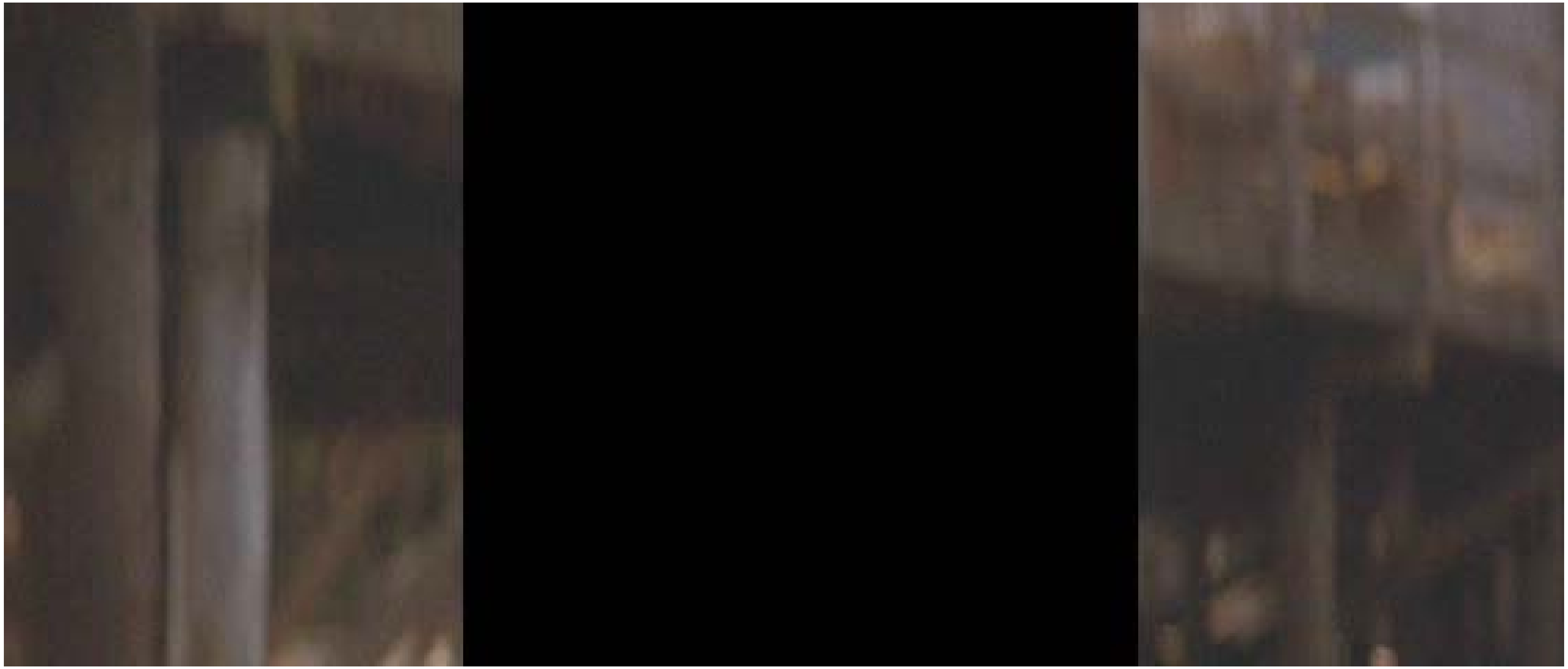
Video compression



Object removal

input video

Object removal



remove foreground

Object removal

estimate background

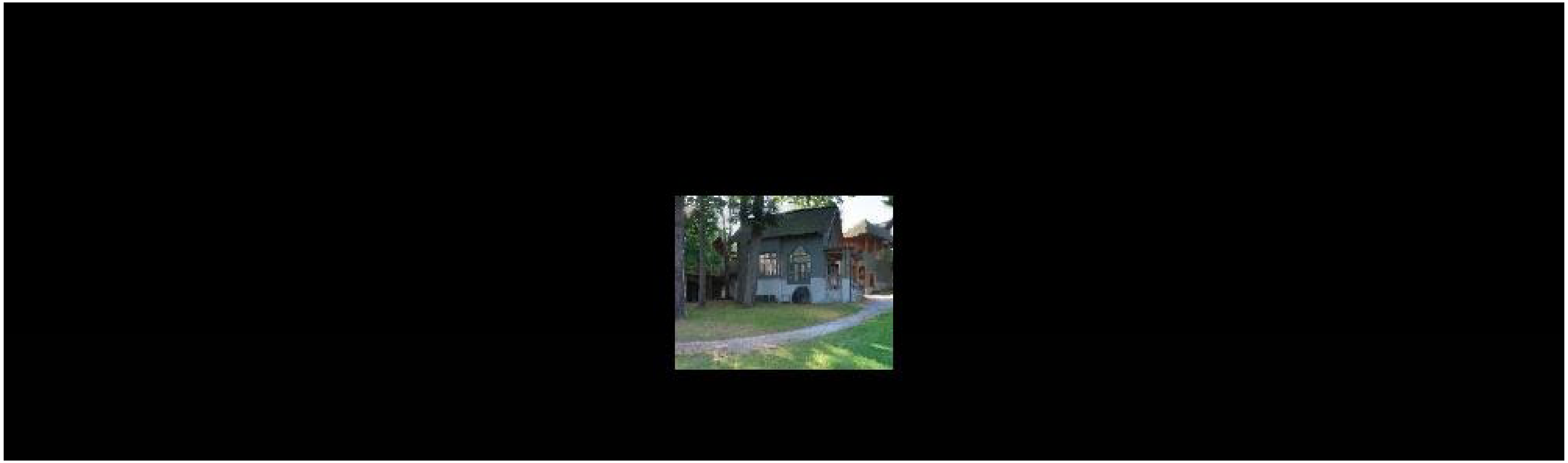
Object removal

background estimation

Panorama creation

Why panorama?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°



Why panorama?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°

Why panorama?

- Are you getting the whole picture?
  - Compact Camera FOV = 50 x 35°
  - Human FOV = 200 x 135°
  - Panoramic Mosaic = 360 x 180°

Panorama examples

- Like HDR, it is a topic of computational photography, seeking ways to build a better camera mostly in software.
- Most consumer cameras have a panorama mode
- Mars:

http://www.panoramas.dk/fullscreen3/f2_mars97.html

- Earth:

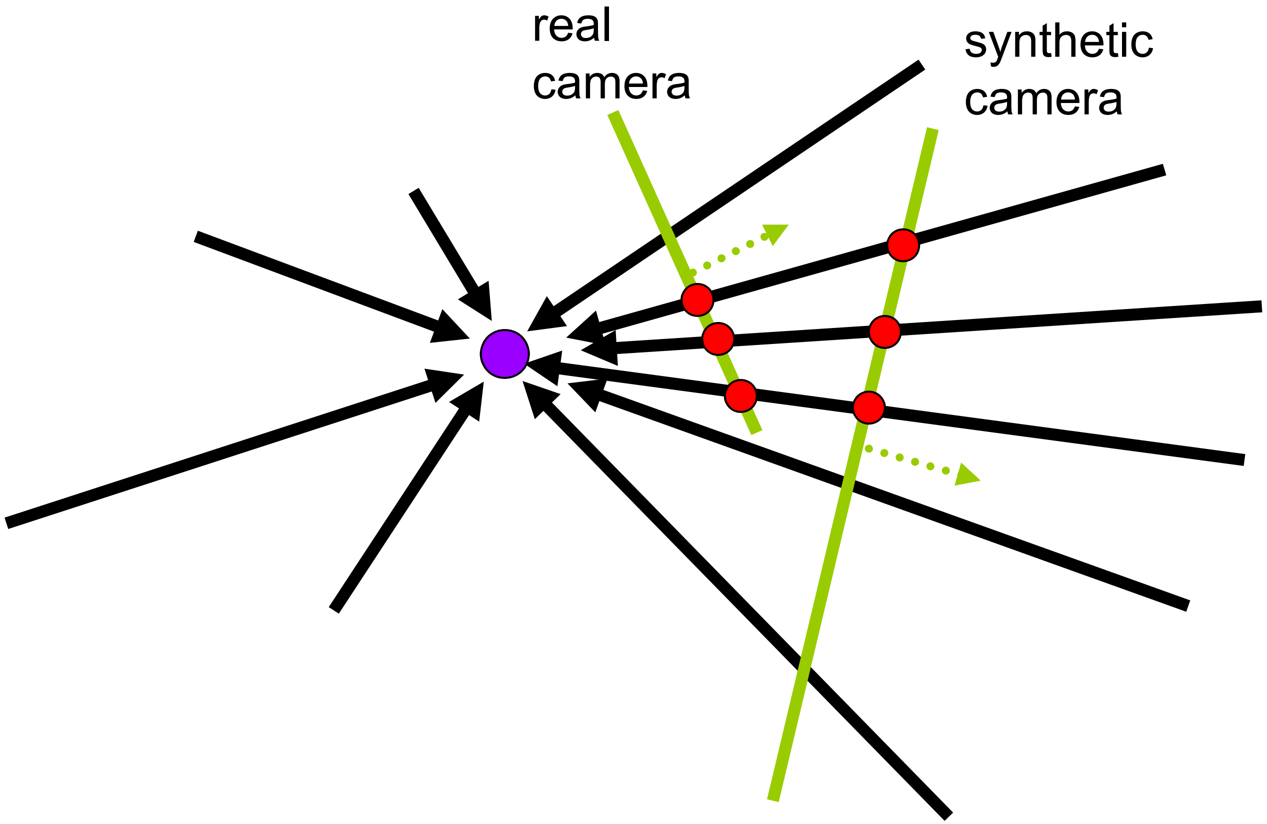
http://www.panoramas.dk/new-year-2006/taipei.html

http://www.360cities.net/

What can be globally aligned?

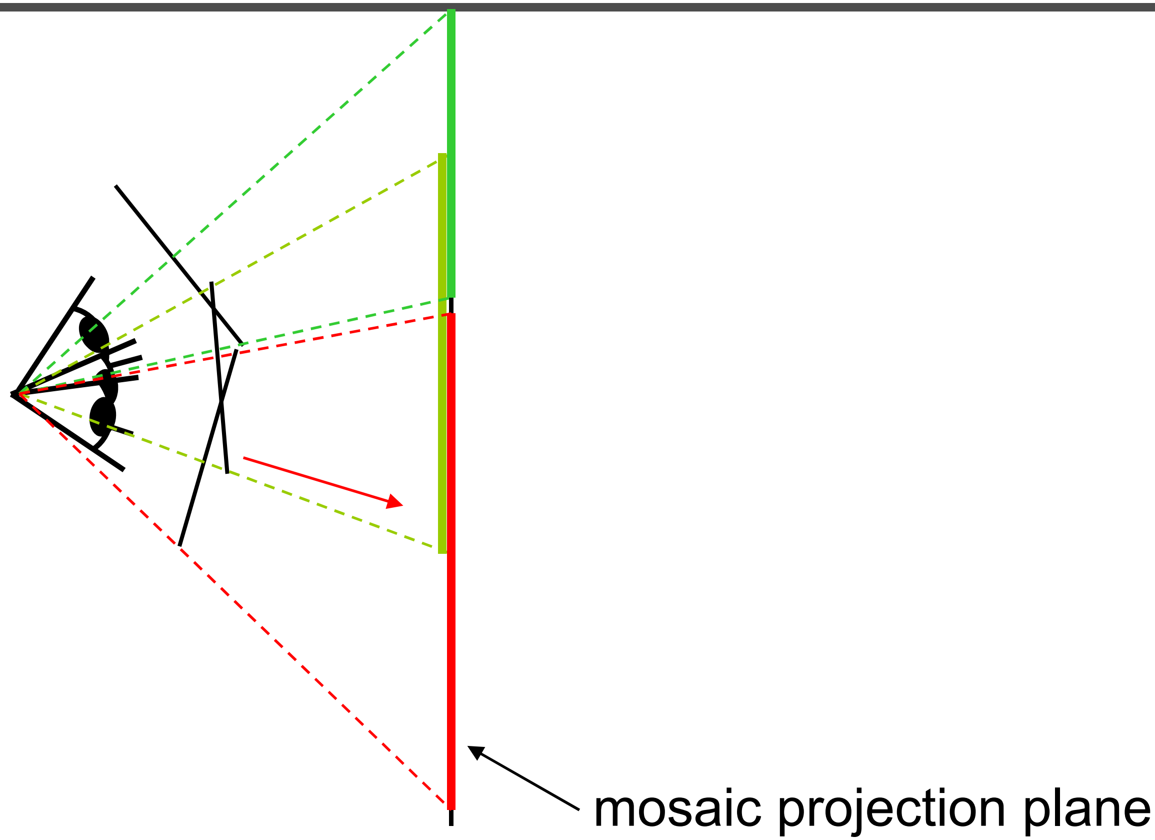
- In image stitching, we seek for a matrix to globally warp one image into another. Are any two images of the same scene can be aligned this way?
  - Images captured with the same center of projection
  - A planar scene or far-away scene

A pencil of rays contains all views



Can generate any synthetic camera view
as long as it has **the same center of projection**!

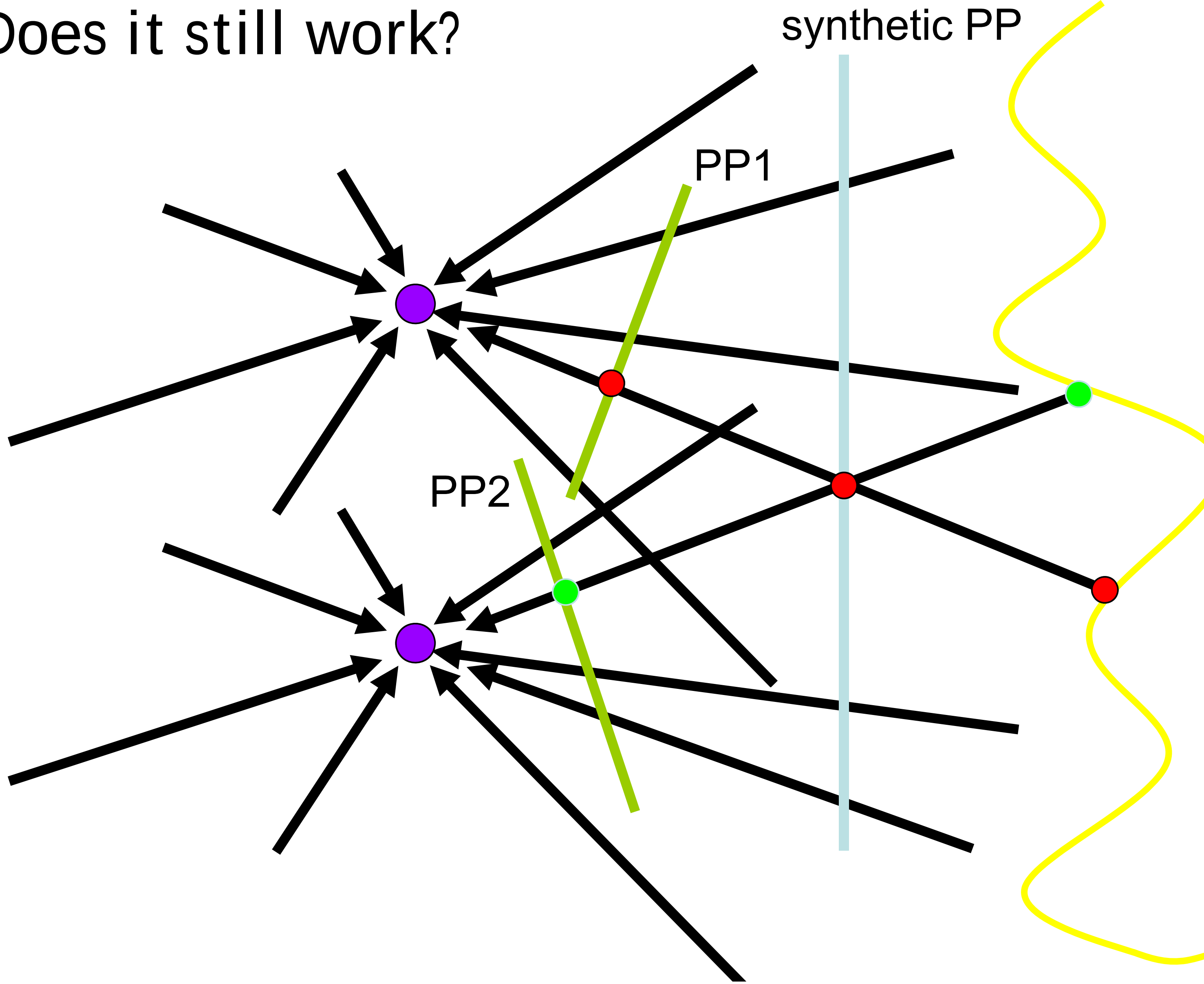
Mosaic as an image reprojection



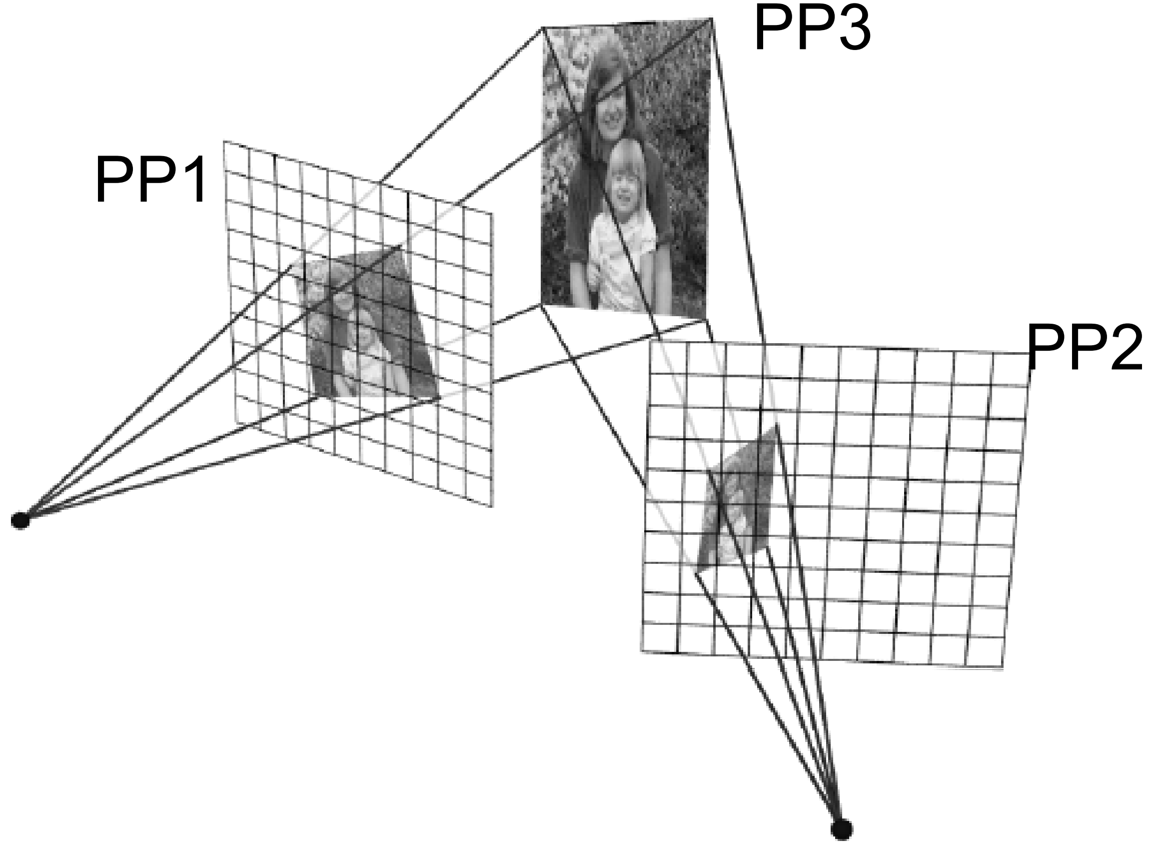
- The images are reprojected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a *synthetic wide-angle camera*

Changing camera center

- Does it still work?



Planar scene (or a faraway one)

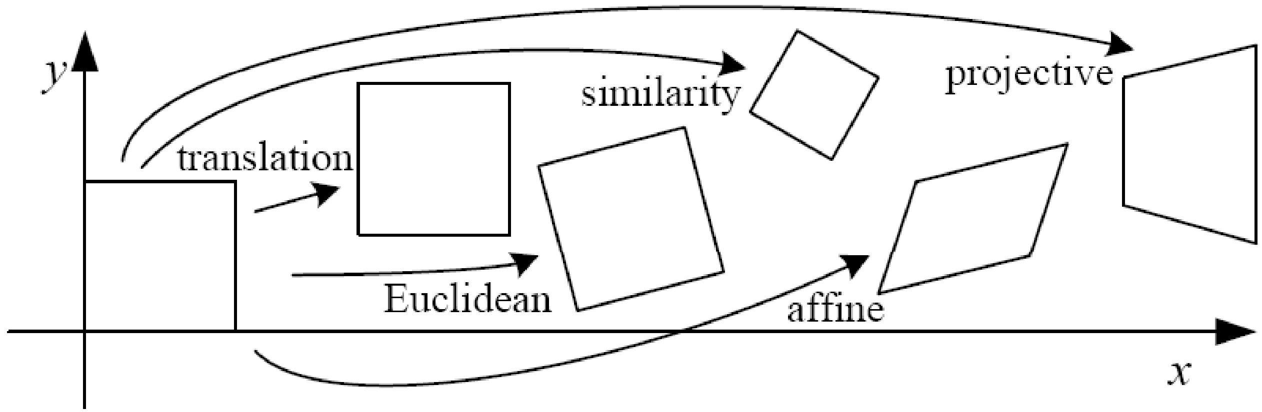


- PP3 is a projection plane of both centers of projection, so we are OK!
- This is how big aerial photographs are made

Motion models

- Parametric models as the assumptions on the relation between two images.

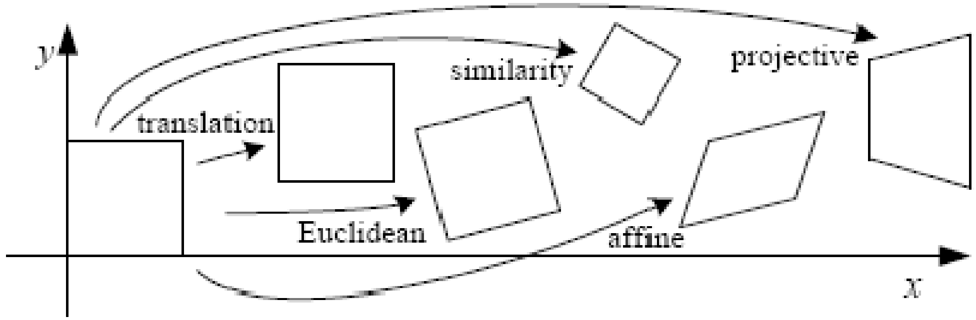
2D Motion models



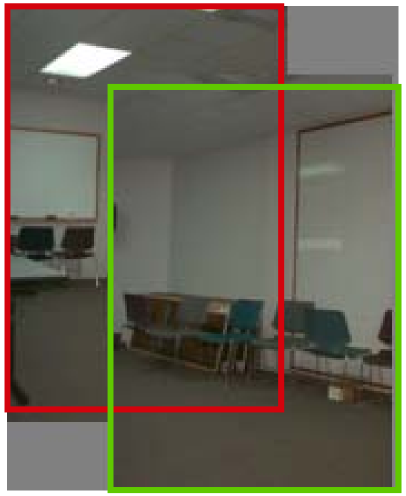
Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+ \cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+ \cdots$	
similarity	$\begin{bmatrix}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

Motion models

DigiVFX

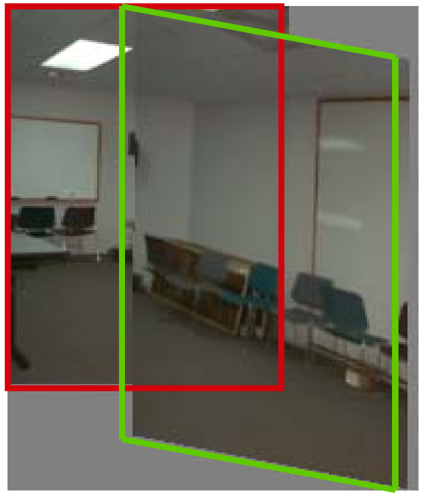


Translation



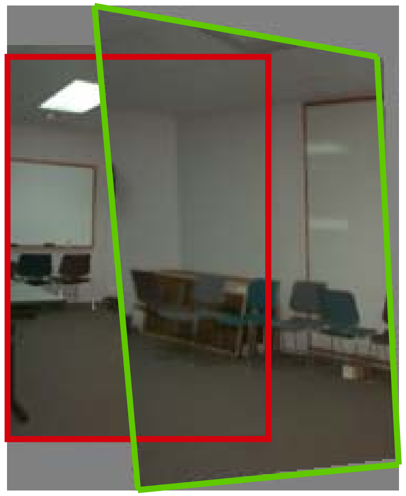
2 unknowns

Affine



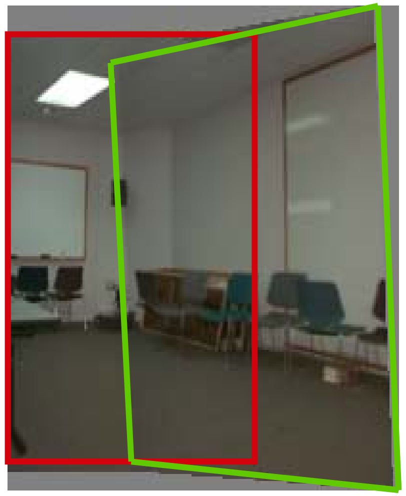
6 unknowns

Perspective



8 unknowns

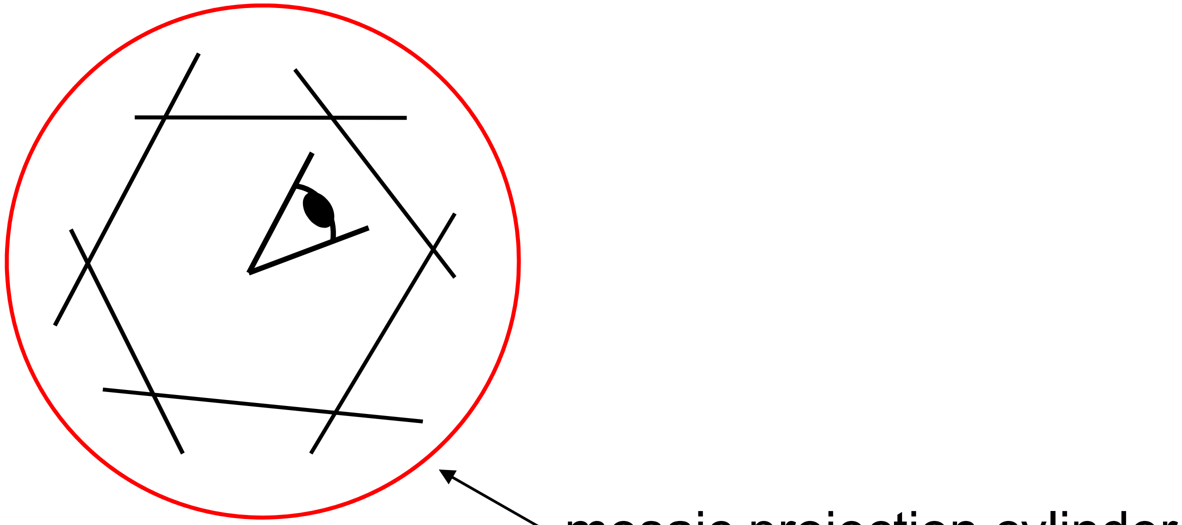
3D rotation



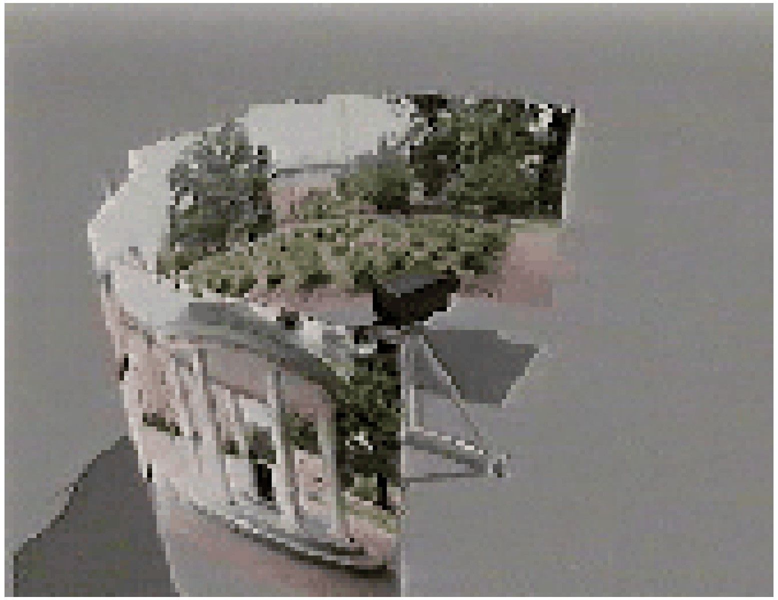
3 unknowns

A case study: cylindrical panorama

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



Cylindrical panoramas



- Steps
  - Reproject each image onto a cylinder
  - Blend
  - Output the resulting mosaic

Cylindrical panorama

1. Take pictures on a tripod (or handheld)
2. Warp to cylindrical coordinate
3. Compute pairwise alignments
4. Fix up the end-to-end alignment
5. Blending
6. Crop the result and import into a viewer

It is required to do radial distortion correction for better stitching results!

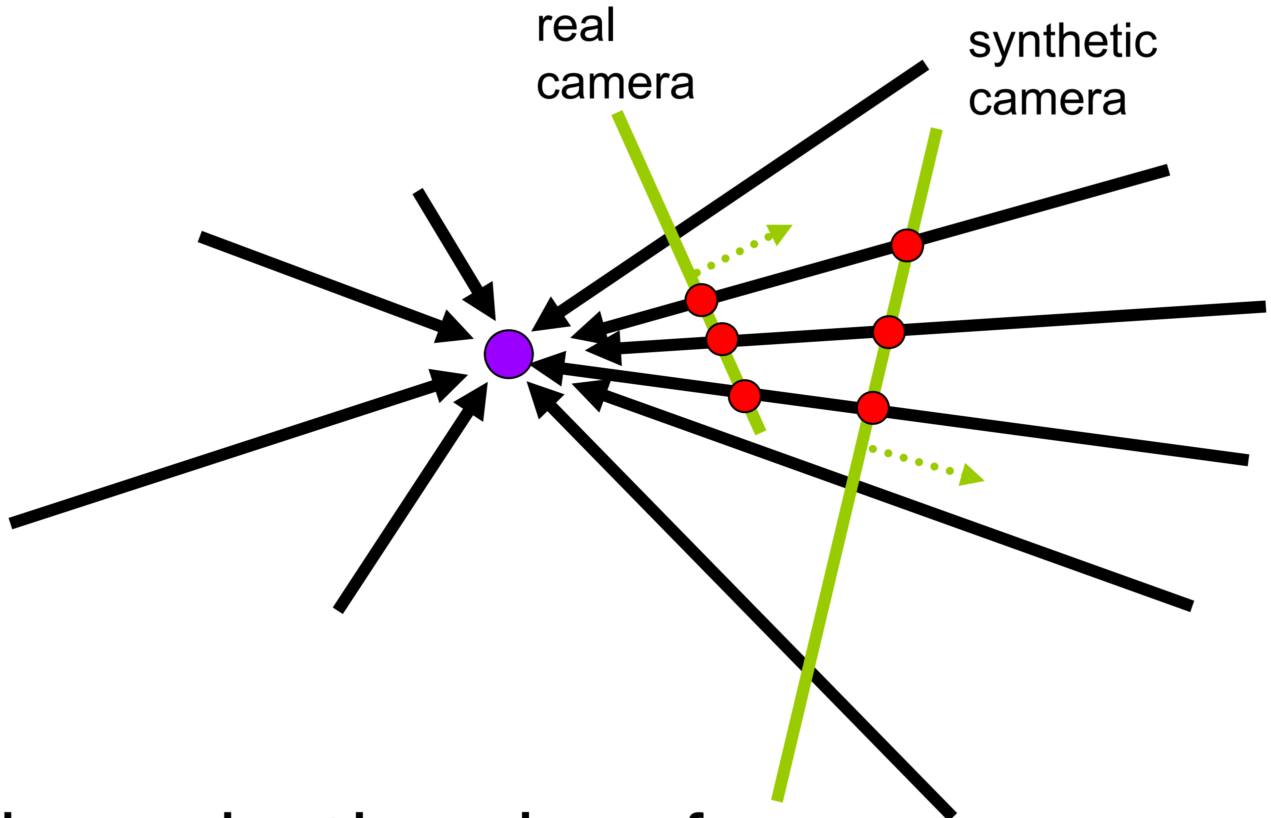
Taking pictures

Kaidan panoramic tripod head

Translation model

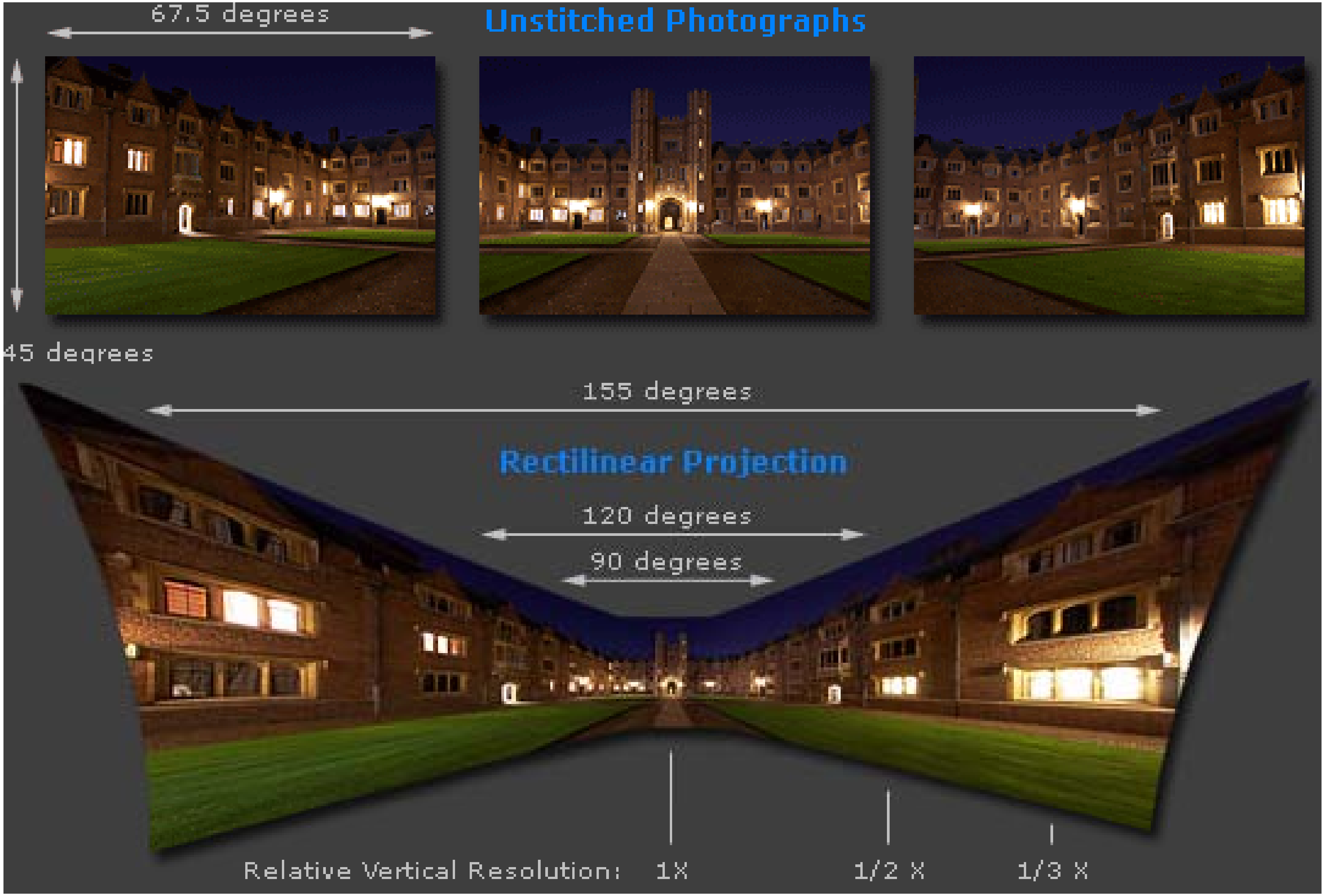
Try to align this in PaintShop Pro

Where should the synthetic camera be



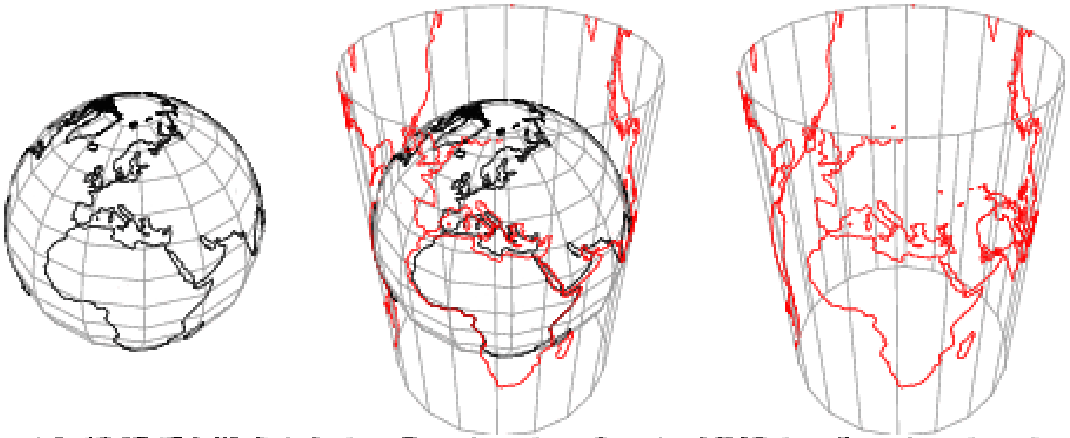
- The projection plan of some camera
- Onto a cylinder

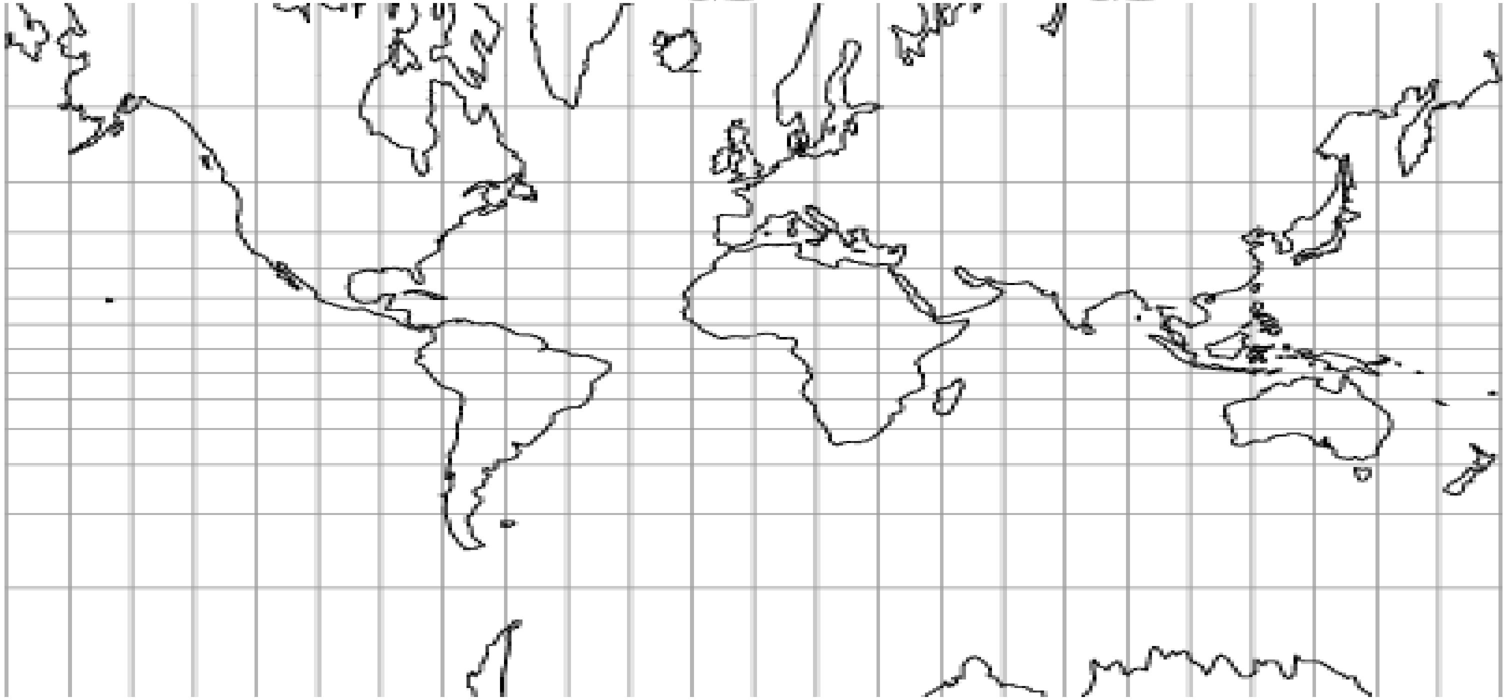
Cylindrical projection



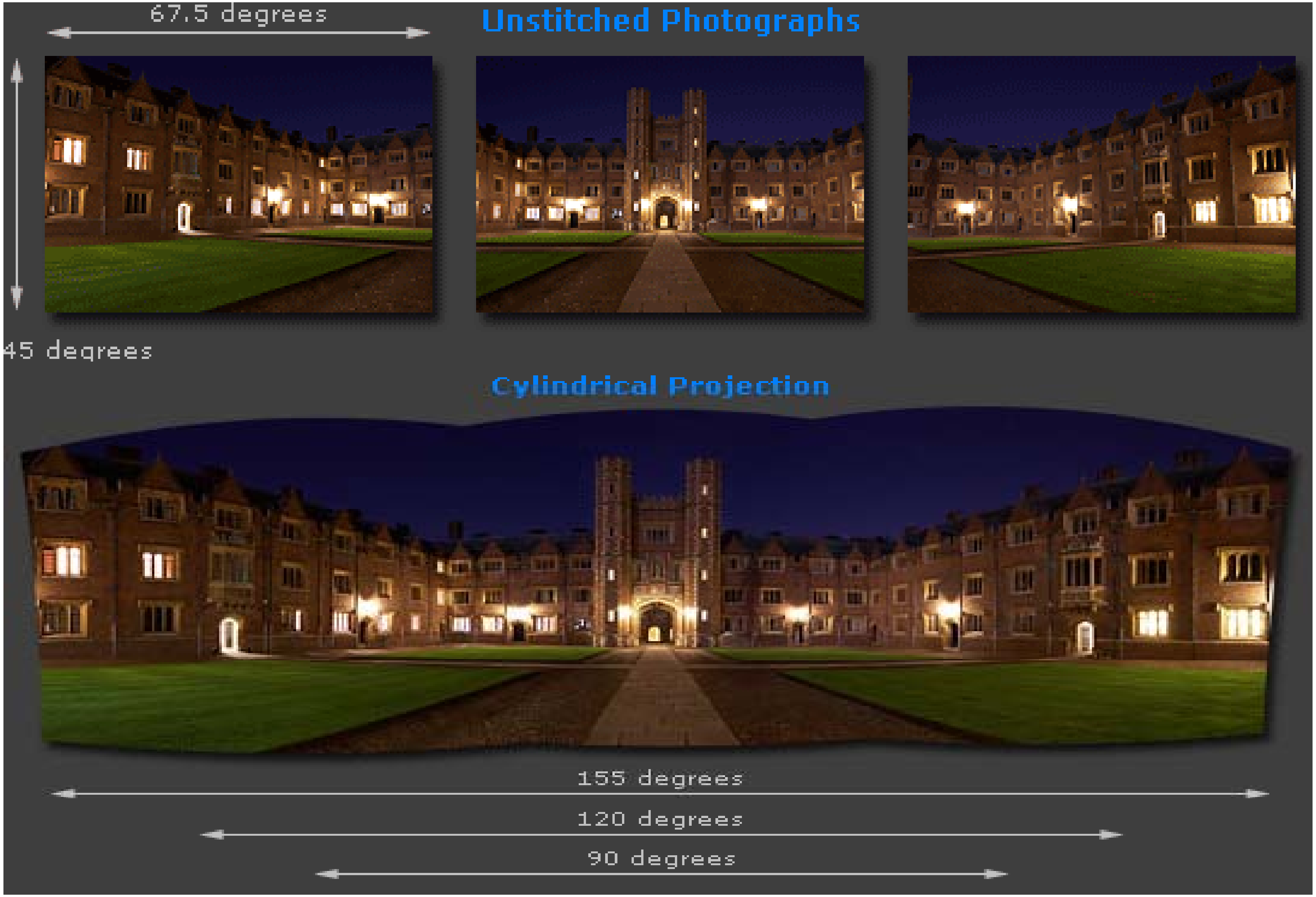
Adopted from http://www.cambridgeincolour.com/tutorials/image-projections.htm

Cylindrical projection



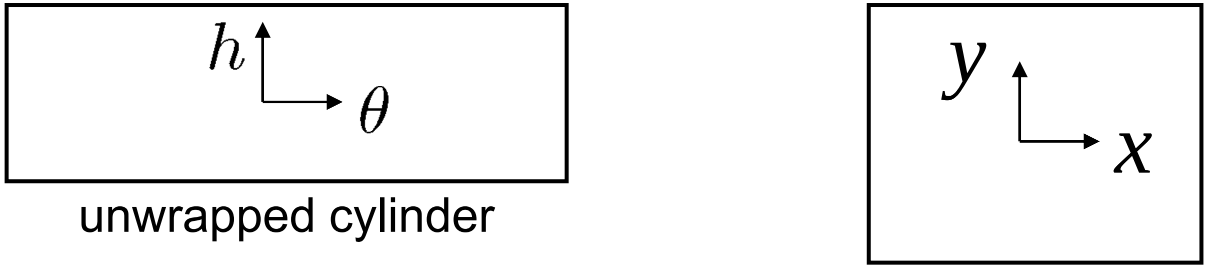


Cylindrical projection

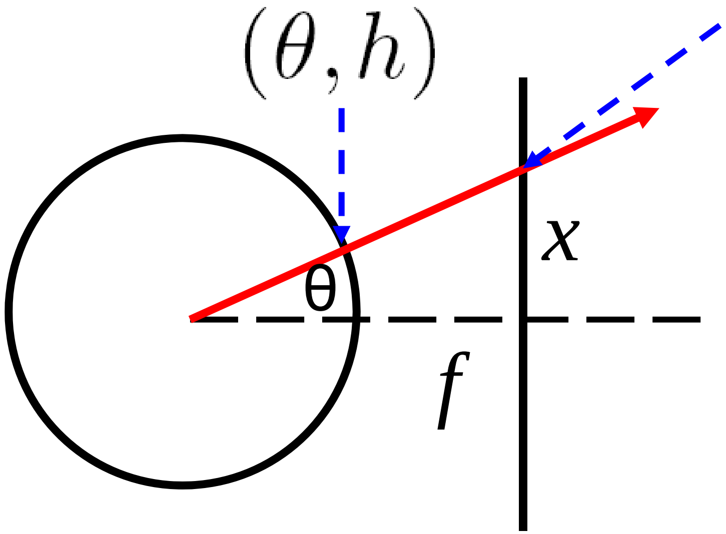


Adopted from http://www.cambridgeincolour.com/tutorials/image-projections.htm

Cylindrical projection

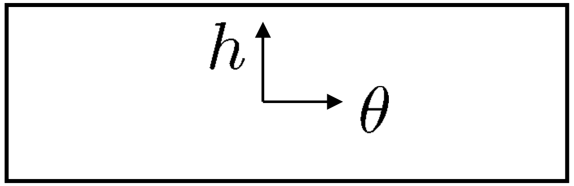


$$(\sin\theta, h, \cos\theta) \propto (x, y, f)$$

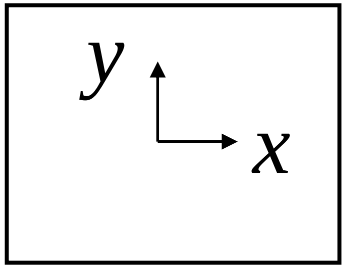


$$\theta = \tan^{-1}\frac{x}{f}$$

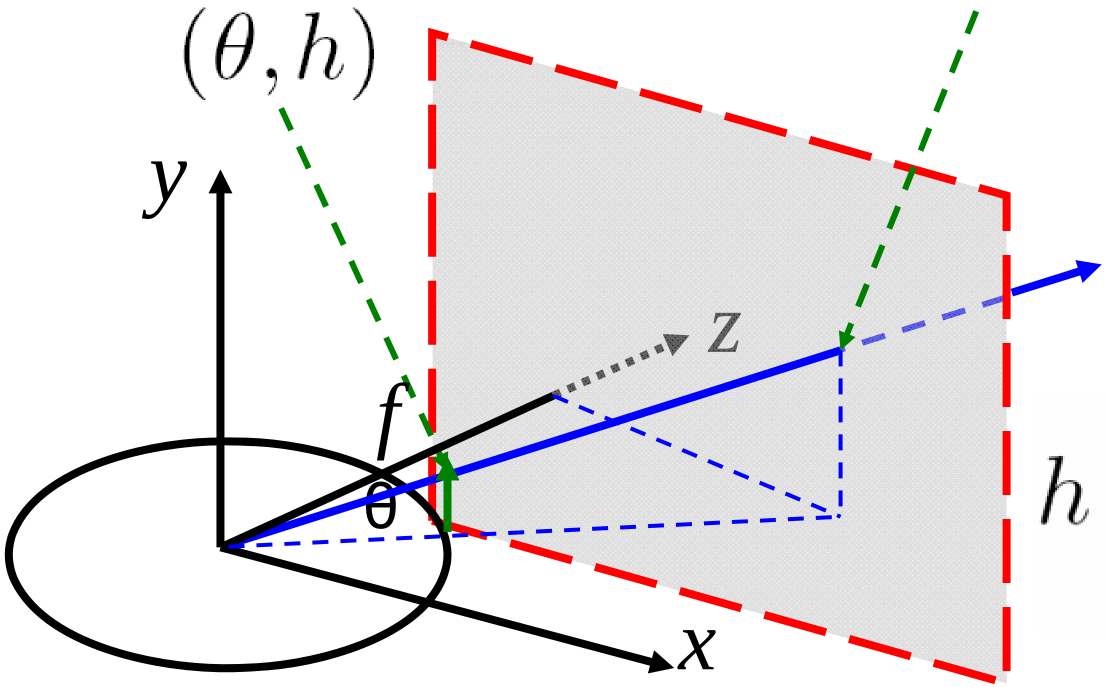
Cylindrical projection



unwrapped cylinder

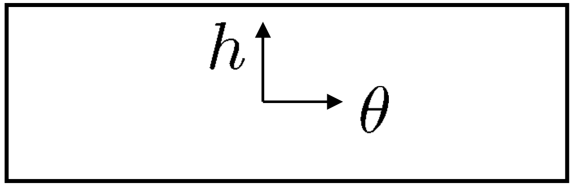


$$(\sin\theta, h, \cos\theta) \propto (x, y, f)$$

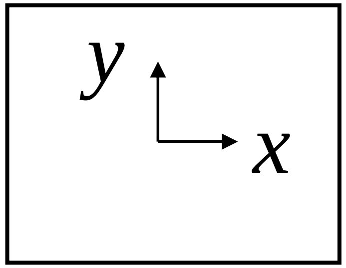


$$h = \frac{y}{\sqrt{x^2 + f^2}}$$

Cylindrical projection

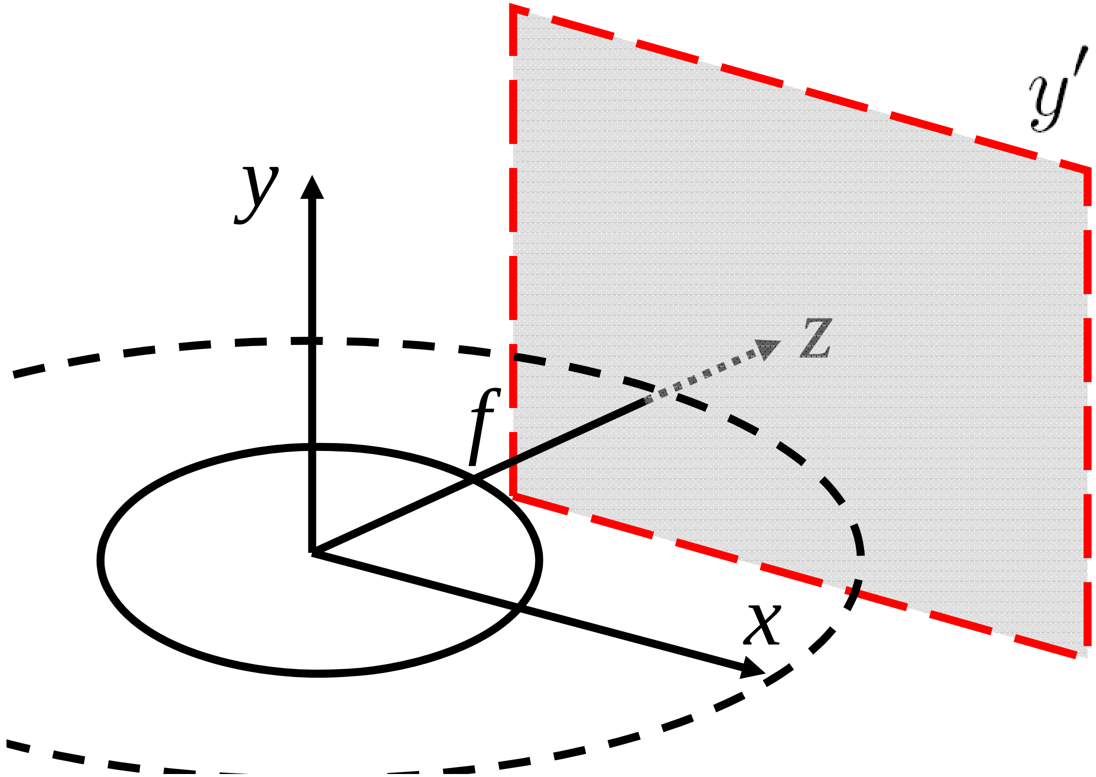


unwrapped cylinder



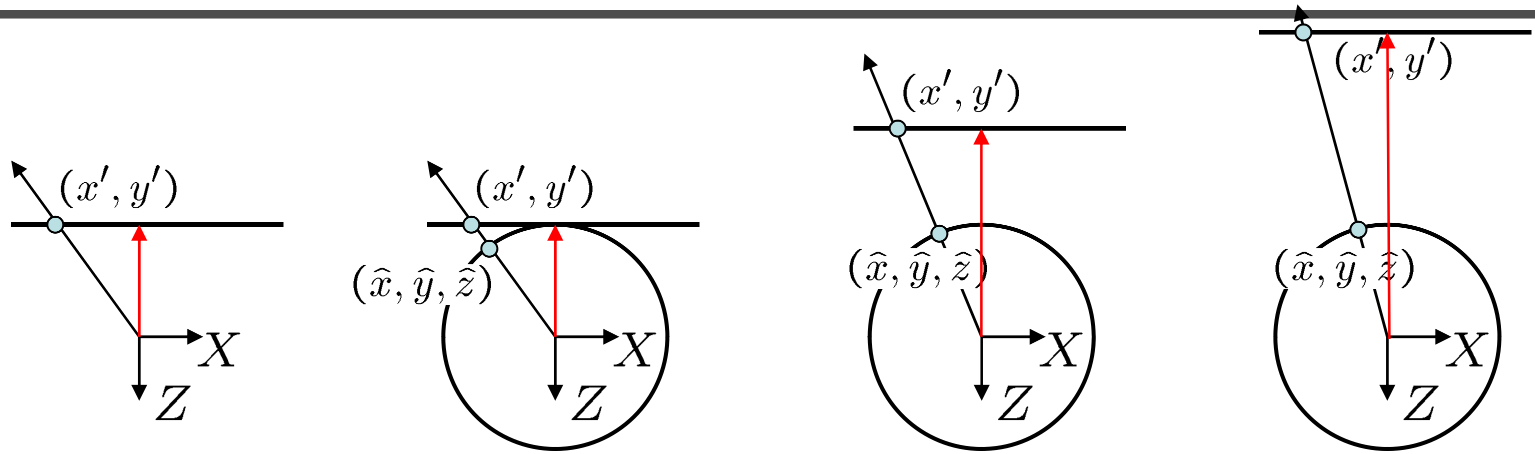
$$x' = s\theta = s \tan^{-1} \frac{x}{f}$$

$$y' = sh = s \frac{y}{\sqrt{x^2 + f^2}}$$



s=*f* gives less distortion

Cylindrical reprojection

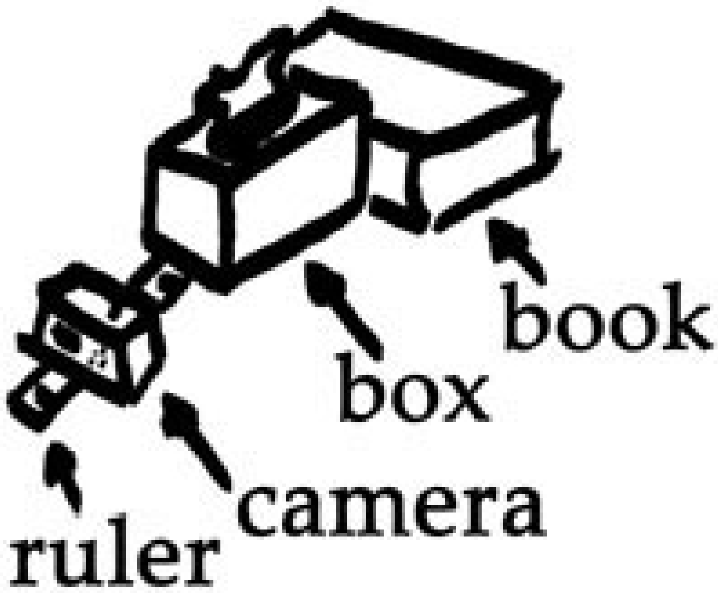


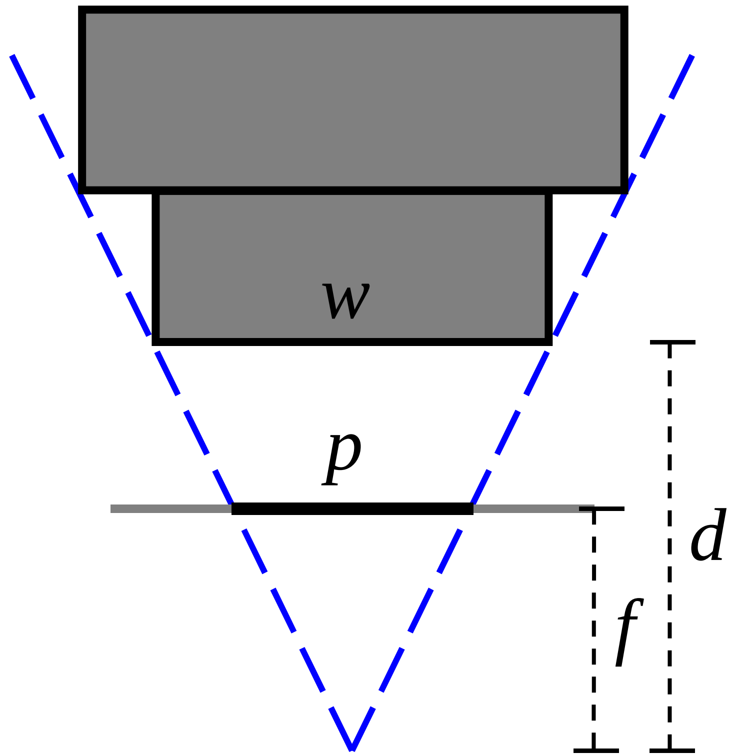
top-down view

Focal length – the dirty secret…

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

A simple method for estimating f





Or, you can use other software, such as AutoStich, to help.

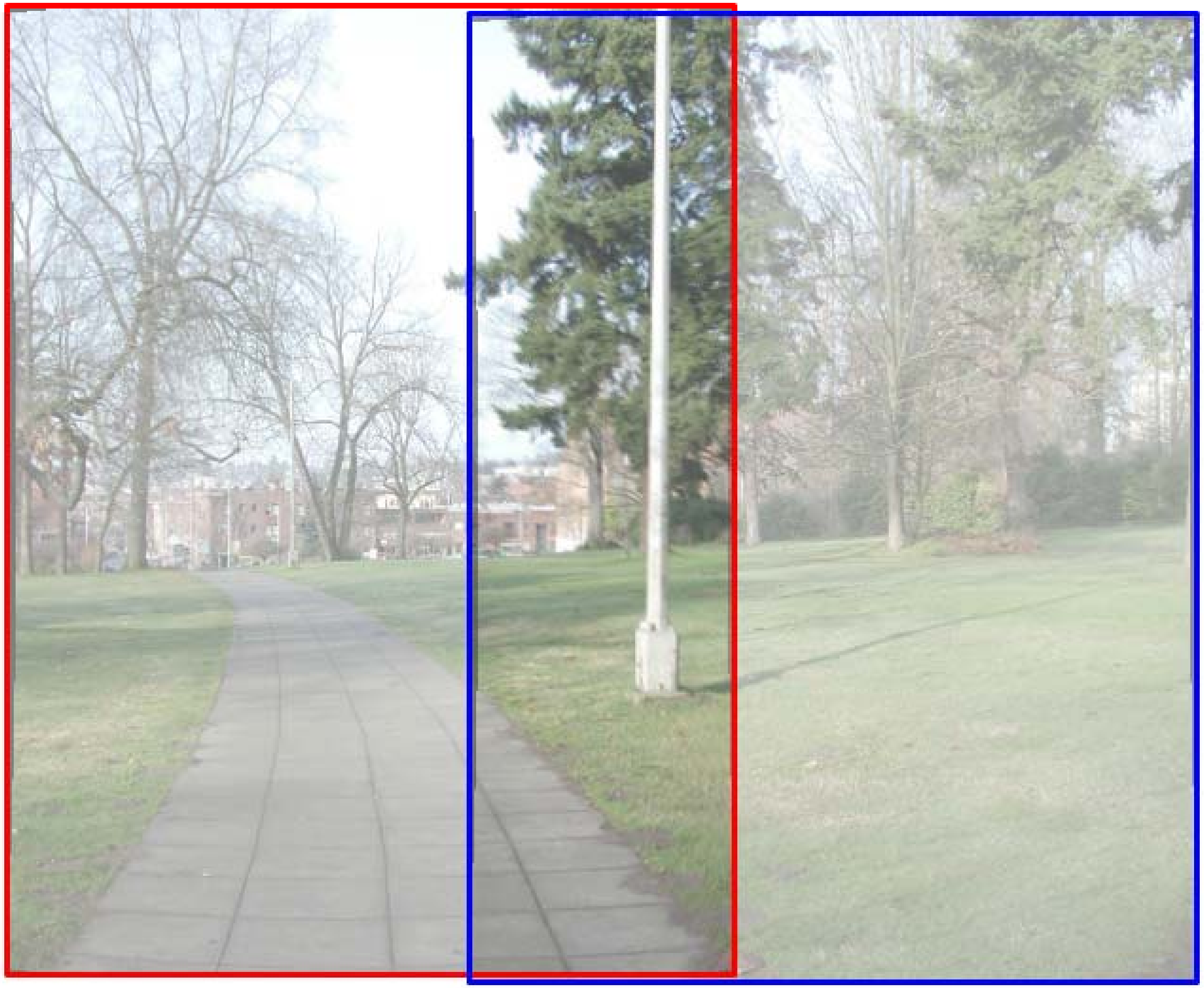
Input images

Cylindrical warping

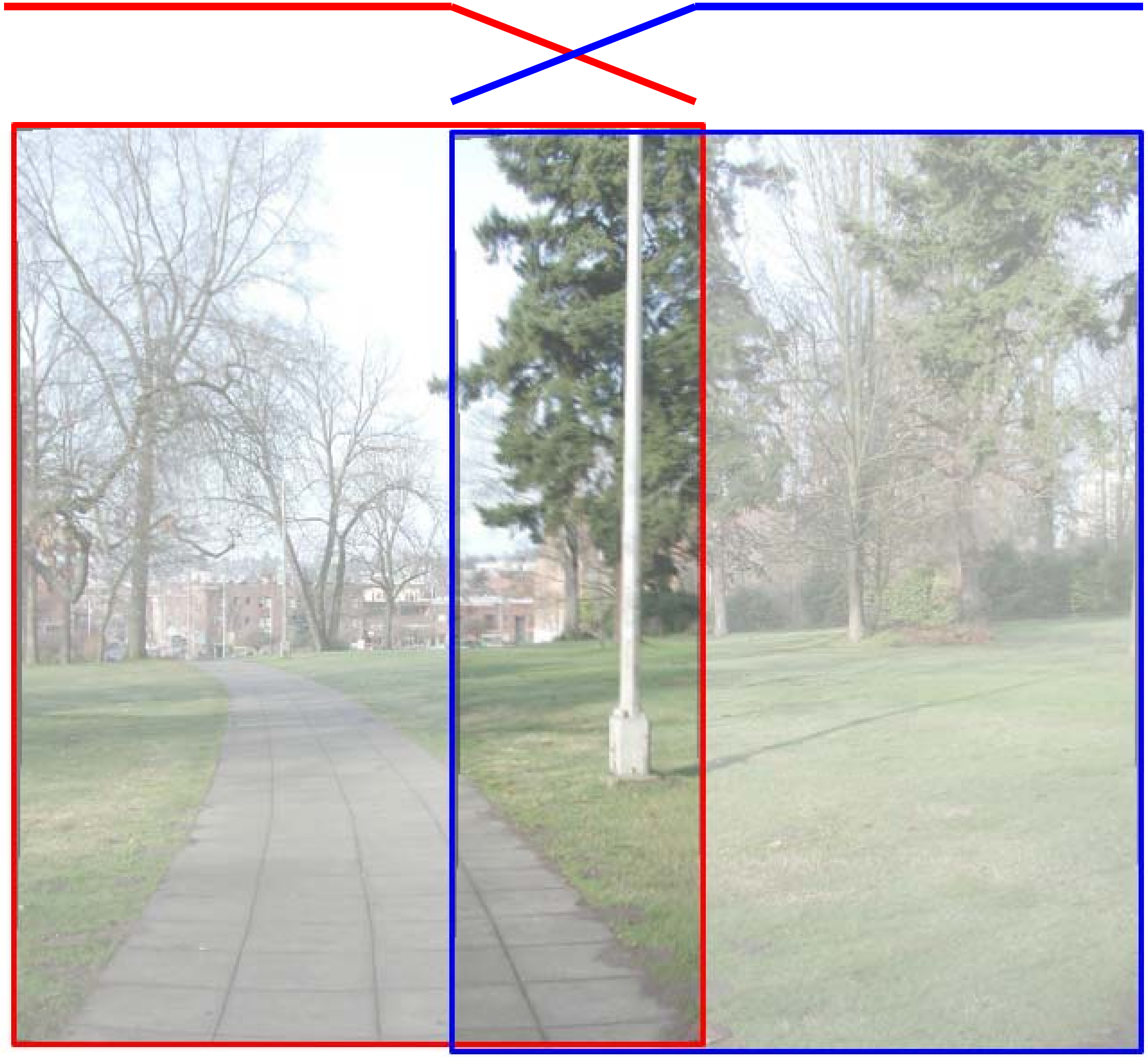
Blending

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

Blending

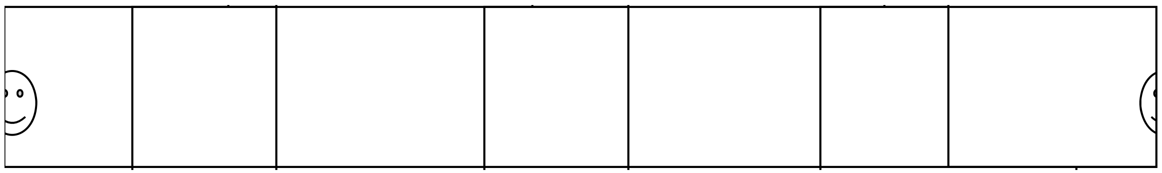


Blending



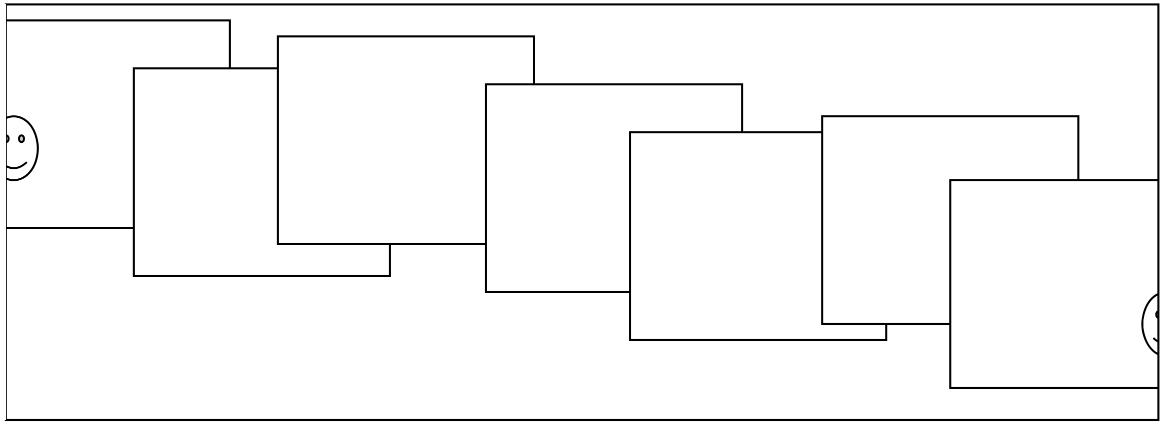
Blending

Assembling the panorama



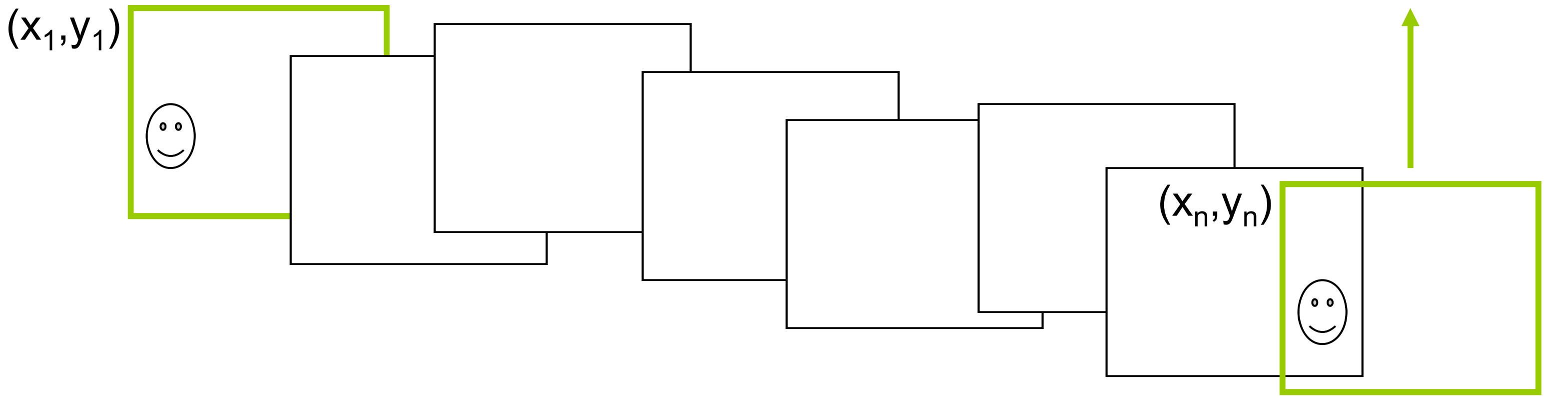
- Stitch pairs together, blend, then crop

Problem: Drift



- Error accumulation
  - small errors accumulate over time

Problem: Drift

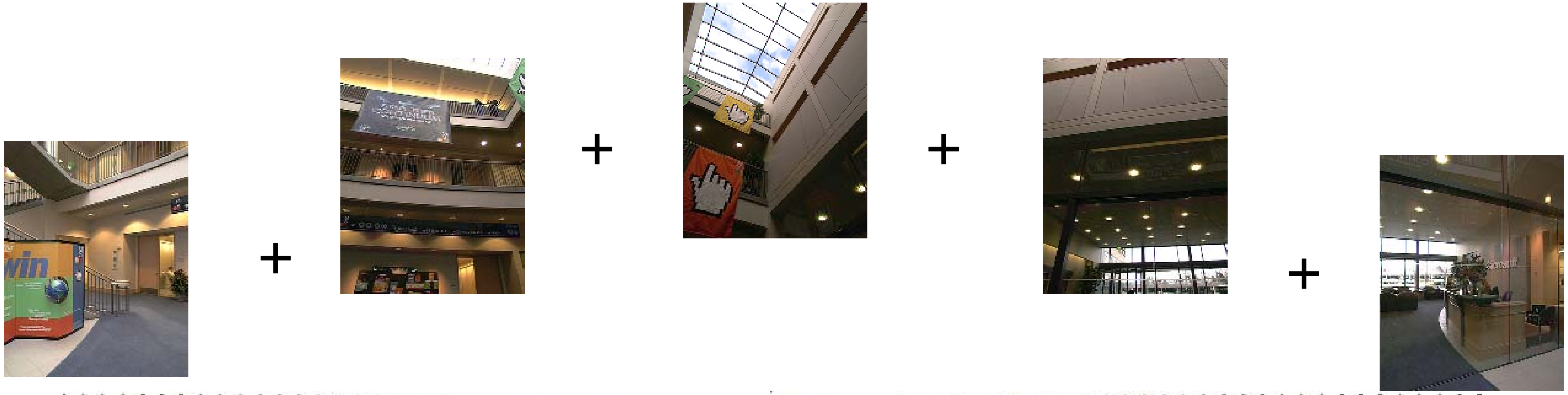


- copy of first image

- 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

Viewer: panorama



example: http://www.cs.washington.edu/education/courses/cse590ss/01wi/projects/project1/students/dougz/index.html

Viewer: texture mapped model

example: http://www.panoramas.dk/

Cylindrical panorama

1. Take pictures on a tripod (or handheld)
2. Warp to cylindrical coordinate
3. Compute pairwise alignments
4. Fix up the end-to-end alignment
5. Blending
6. Crop the result and import into a viewer

Determine pairwise alignment?

- Feature-based methods: only use feature points to estimate parameters
- We will study the "Recognising panorama" paper published in ICCV 2003
- Run SIFT (or other feature algorithms) for each image, find feature matches.

Determine pairwise alignment

- p'=Mp, where M is a transformation matrix, p and p' are feature matches
- It is possible to use more complicated models such as affine or perspective
- For example, assume M is a 2x2 matrix

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- Find M with the least square error

$$\sum_{i=1}^{n} (Mp - p')^2$$

Determine pairwise alignment

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$x_1 m_{11} + y_1 m_{12} = x_1'$$
$$x_1 m_{21} + y_1 m_{22} = y_1'$$

- Overdetermined system

$$\begin{pmatrix} x_1 & y_1 & 0 & 0 \\ 0 & 0 & x_1 & y_1 \\ x_2 & y_2 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 0 & 0 \\ 0 & 0 & x_n & y_n \end{pmatrix} \begin{pmatrix} m_{11} \\ m_{12} \\ m_{21} \\ m_{22} \end{pmatrix} = \begin{pmatrix} x_1' \\ y_1' \\ x_2' \\ \vdots \\ x_n' \\ y_n' \end{pmatrix}$$

Normal equation

Given an overdetermined system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

the normal equation is that which minimizes the sum of the square differences between left and right sides

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Why?

Normal equation

$$E(\mathbf{x}) = (\mathbf{A}\mathbf{x} - \mathbf{b})^2$$

$$\begin{bmatrix} a_{11} & \dots & a_{1m} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

nxm, n equations, m variables

Normal equation

$$\mathbf{Ax}-\mathbf{b}=\begin{bmatrix}\sum_{j=1}^{m}a_{1j}x_j\\ \vdots\\ \sum_{j=1}^{m}a_{ij}x_j\\ \vdots\\ \sum_{j=1}^{m}a_{nj}x_j\end{bmatrix}-\begin{bmatrix}b_1\\ \vdots\\ b_i\\ \vdots\\ b_n\end{bmatrix}=\begin{bmatrix}\left(\sum_{j=1}^{m}a_{1j}x_j\right)-b_1\\ \vdots\\ \left(\sum_{j=1}^{m}a_{ij}x_j\right)-b_i\\ \vdots\\ \left(\sum_{j=1}^{m}a_{nj}x_j\right)-b_n\end{bmatrix}$$

$$E(\mathbf{x})=(\mathbf{Ax}-\mathbf{b})^2=\sum_{i=1}^{n}\left[\left(\sum_{j=1}^{m}a_{ij}x_j\right)-b_i\right]^2$$

Normal equation

$$E(\mathbf{x}) = (\mathbf{A}\mathbf{x}-\mathbf{b})^2 = \sum_{i=1}^{n}\left[\left(\sum_{j=1}^{m} a_{ij}x_j\right)-b_i\right]^2$$

$$0 = \frac{\partial E}{\partial x_1} = \sum_{i=1}^{n} 2\left[\left(\sum_{j=1}^{m} a_{ij}x_j\right)-b_i\right]a_{i1}$$

$$= 2\sum_{i=1}^{n} a_{i1}\sum_{j=1}^{m} a_{ij}x_j - 2\sum_{i=1}^{n} a_{i1}b_i$$

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

Normal equation

$$(\mathbf{Ax}-\mathbf{b})^2$$

$$=(\mathbf{Ax}-\mathbf{b})^T(\mathbf{Ax}-\mathbf{b})$$

$$=\left((\mathbf{Ax})^T-\mathbf{b}^T\right)(\mathbf{Ax}-\mathbf{b})$$

$$=\left(\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}-\mathbf{b}^{\mathrm{T}}\right)(\mathbf{Ax}-\mathbf{b})$$

$$=\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{Ax}-\mathbf{b}^{\mathrm{T}}\mathbf{Ax}-\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{b}+\mathbf{b}^{\mathrm{T}}\mathbf{b}$$

$$=\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{Ax}-\left(\mathbf{A}^{\mathrm{T}}\mathbf{b}\right)^{\mathrm{T}}\mathbf{x}-\left(\mathbf{A}^{\mathrm{T}}\mathbf{b}\right)^{\mathrm{T}}\mathbf{x}+\mathbf{b}^{\mathrm{T}}\mathbf{b}$$

$$\frac{\partial E}{\partial \mathbf{x}}=2\mathbf{A}^{\mathrm{T}}\mathbf{Ax}-2\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Determine pairwise alignment

- p'=Mp, where M is a transformation matrix, p and p' are feature matches
- For translation model, it is easier.

$$E = \sum_{i=1}^{n} \left[\left(m_1 + x_i - x_i^{'} \right)^2 + \left(m_2 + y_i - y_i^{'} \right)^2 \right]$$

$$0 = \frac{\partial E}{\partial m_1}$$

- What if the match is false? Avoid impact of outliers.

RANSAC

- RANSAC = Random Sample Consensus
- An algorithm for robust fitting of models in the presence of many data outliers
- Compare to robust statistics

- Given N data points x_i, assume that mjority of them are generated from a model with parameters Θ, try to recover Θ.

RANSAC algorithm

Run k times:

(1) draw n samples randomly

(2) fit parameters Θ with these n samples

(3) for each of other N-n points, calculate its distance to the fitted model, count the number of inlier points, c

Output Θ with the largest c

How many times?

How big?
Smaller is better

How to define?
Depends on the problem.

How to determine k

p: probability of real inliers

P: probability of success after k trials

$$P = 1 - (1 - p^n)^k$$

$\underbrace{p^n}$ n samples are all inliers

$\underbrace{1-p^n}$ a failure

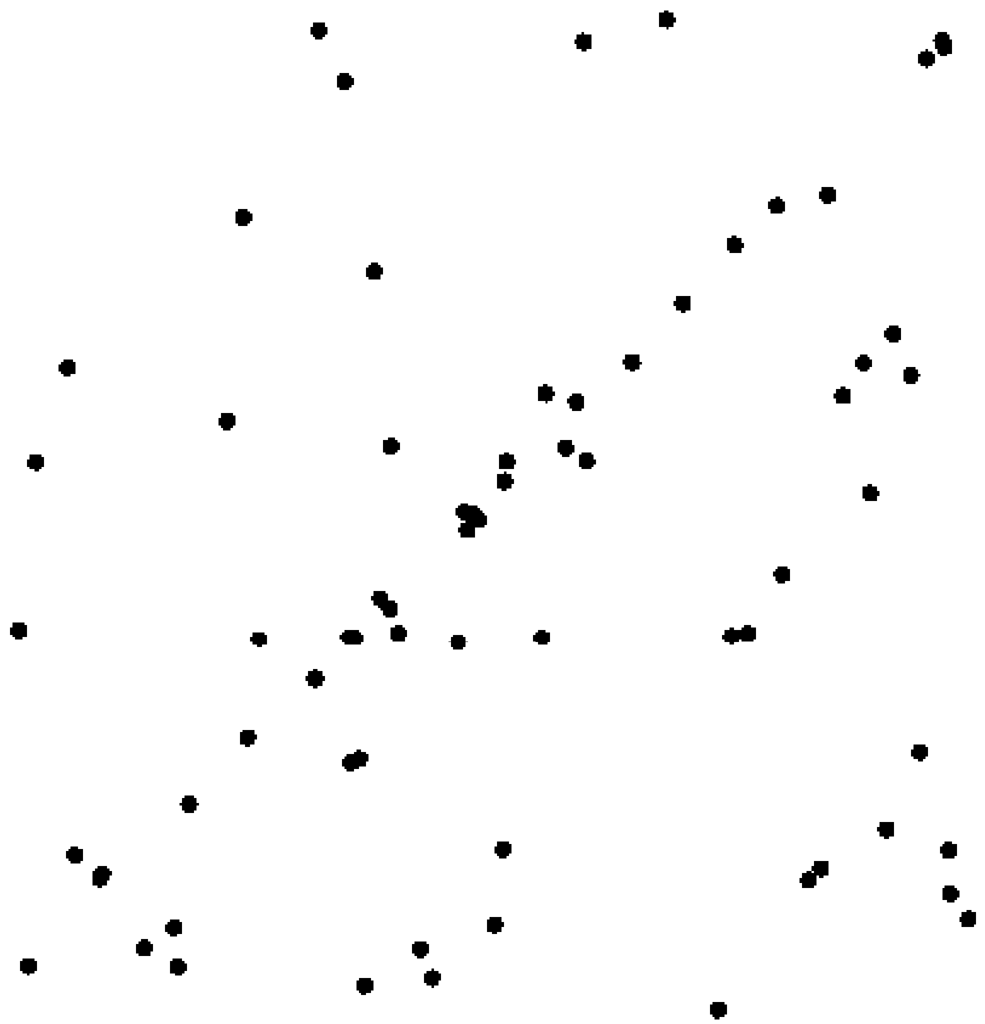
$\underbrace{(1-p^n)^k}$ failure after k trials

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

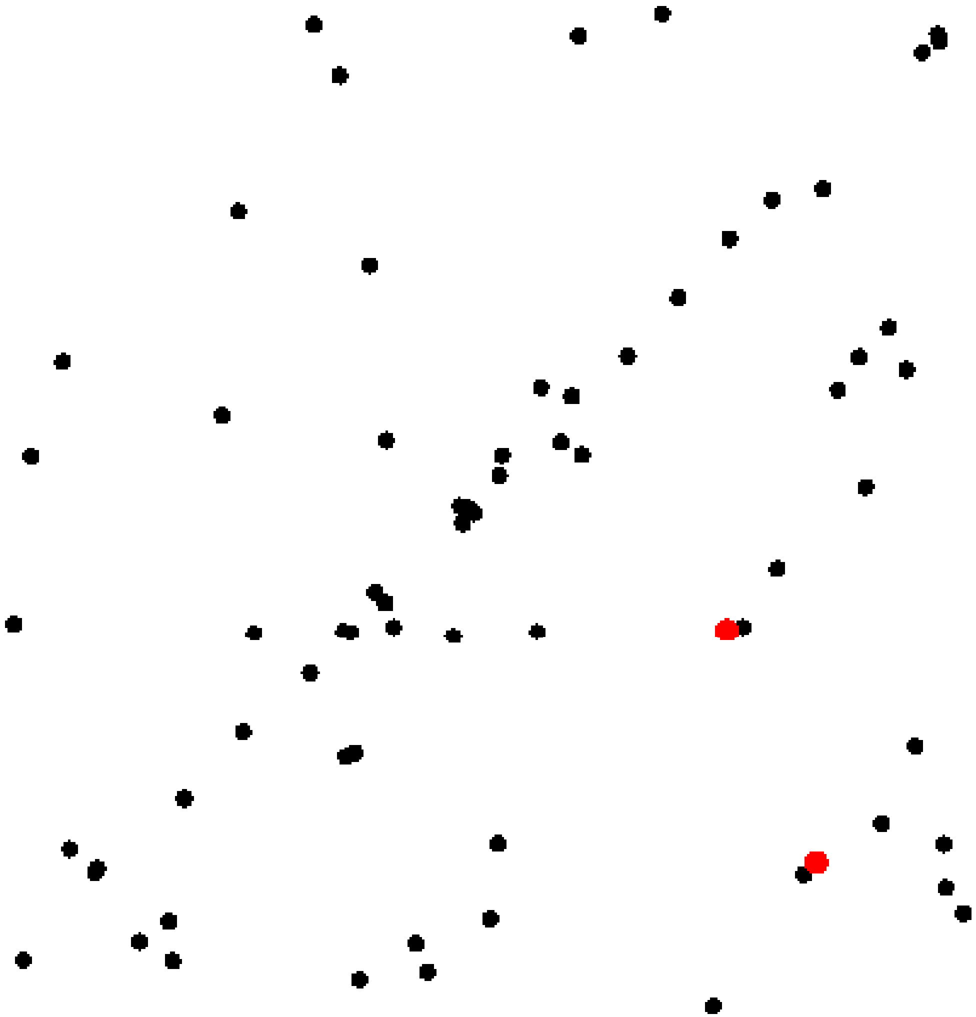
for $P=0.99$

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

Example: line fitting

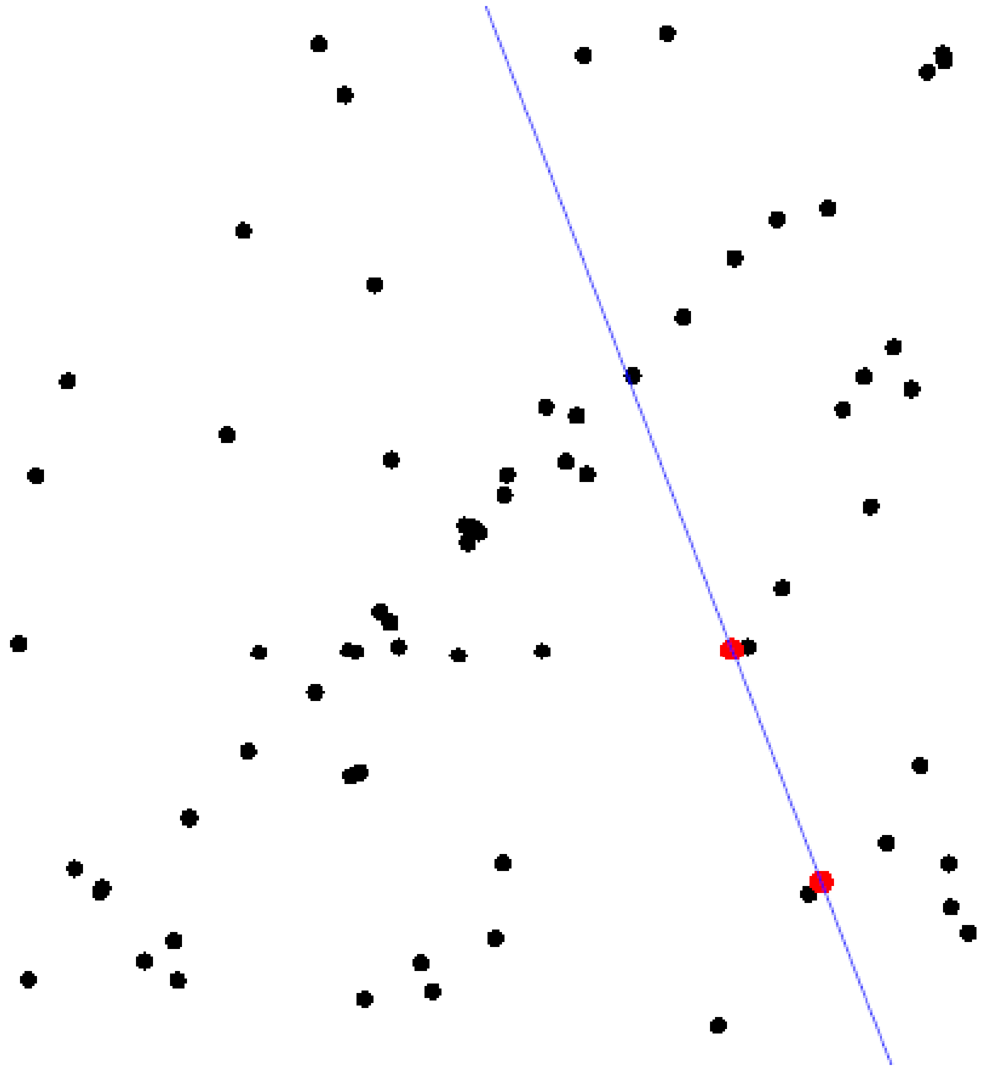


Example: line fitting

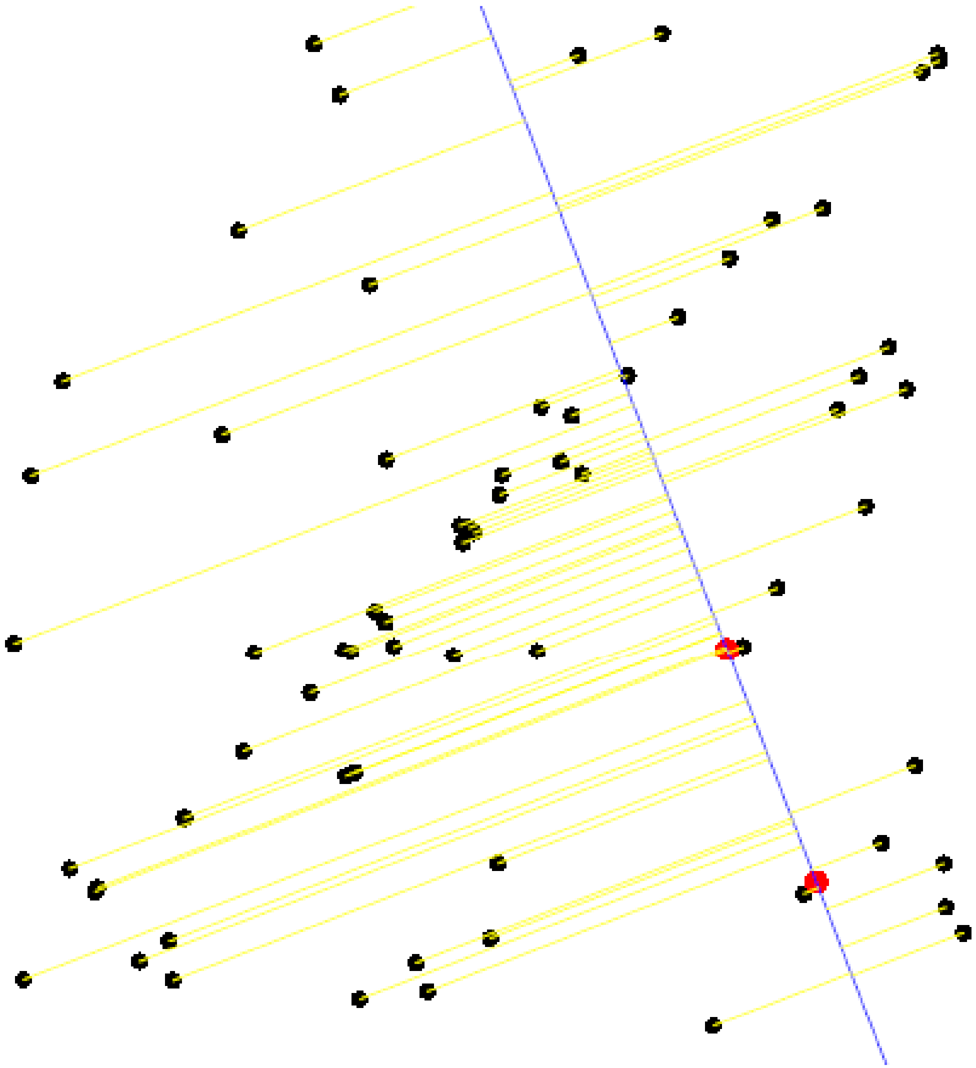


n=2

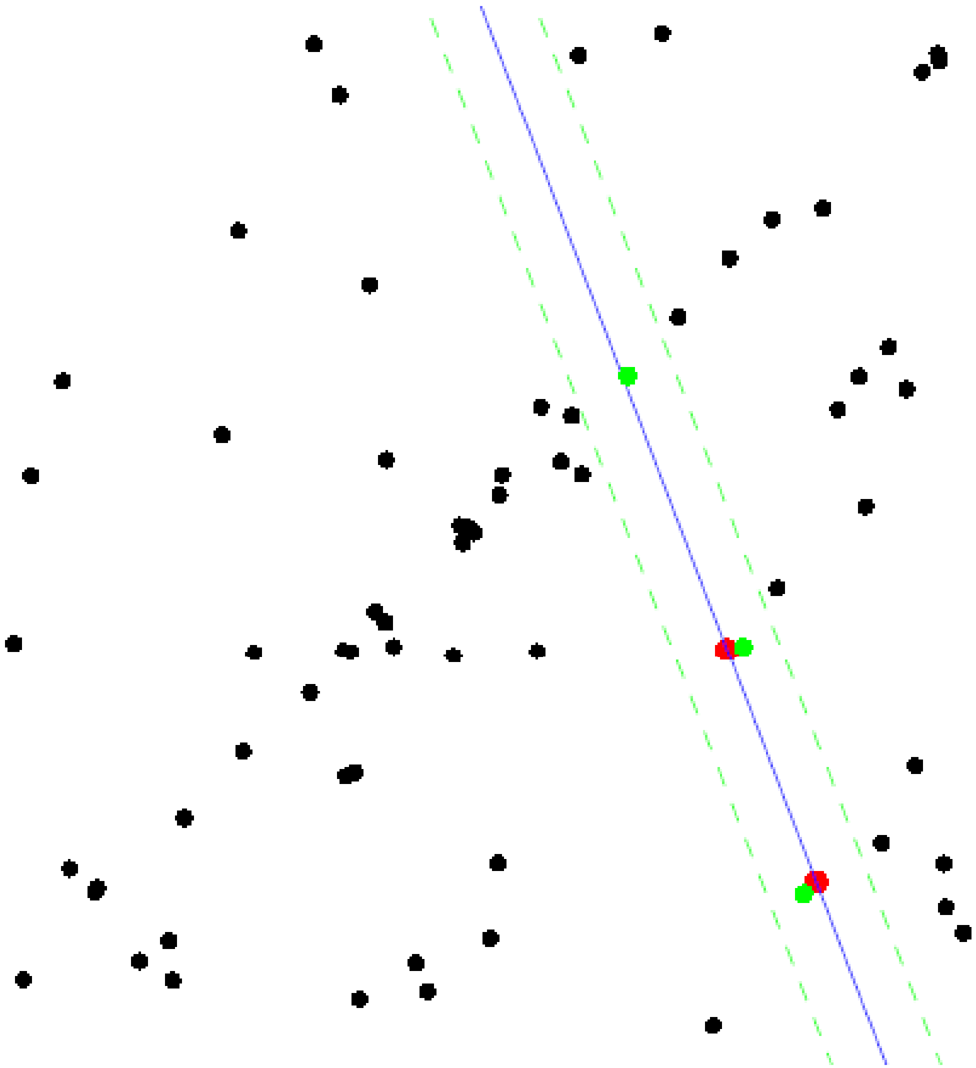
Model fitting



Measure distances

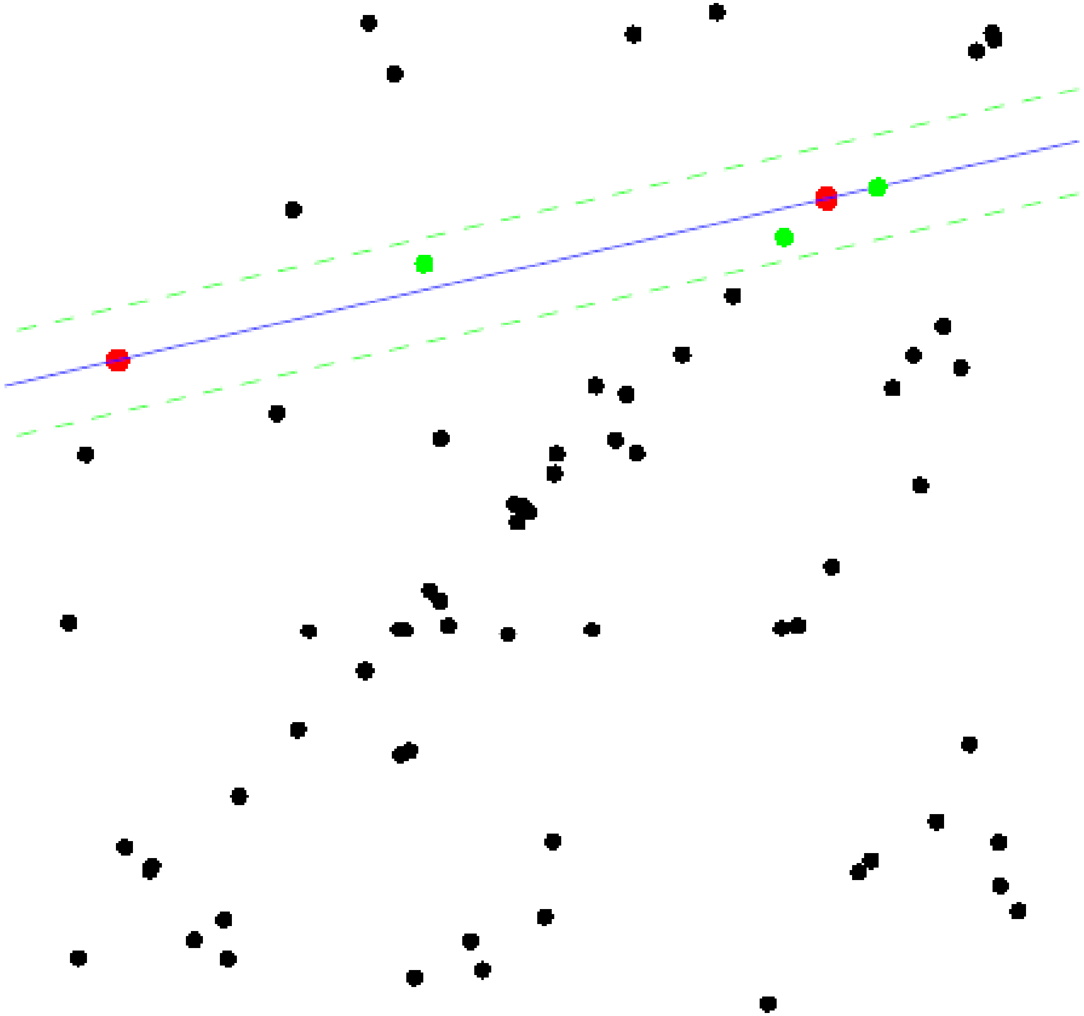


Count inliers



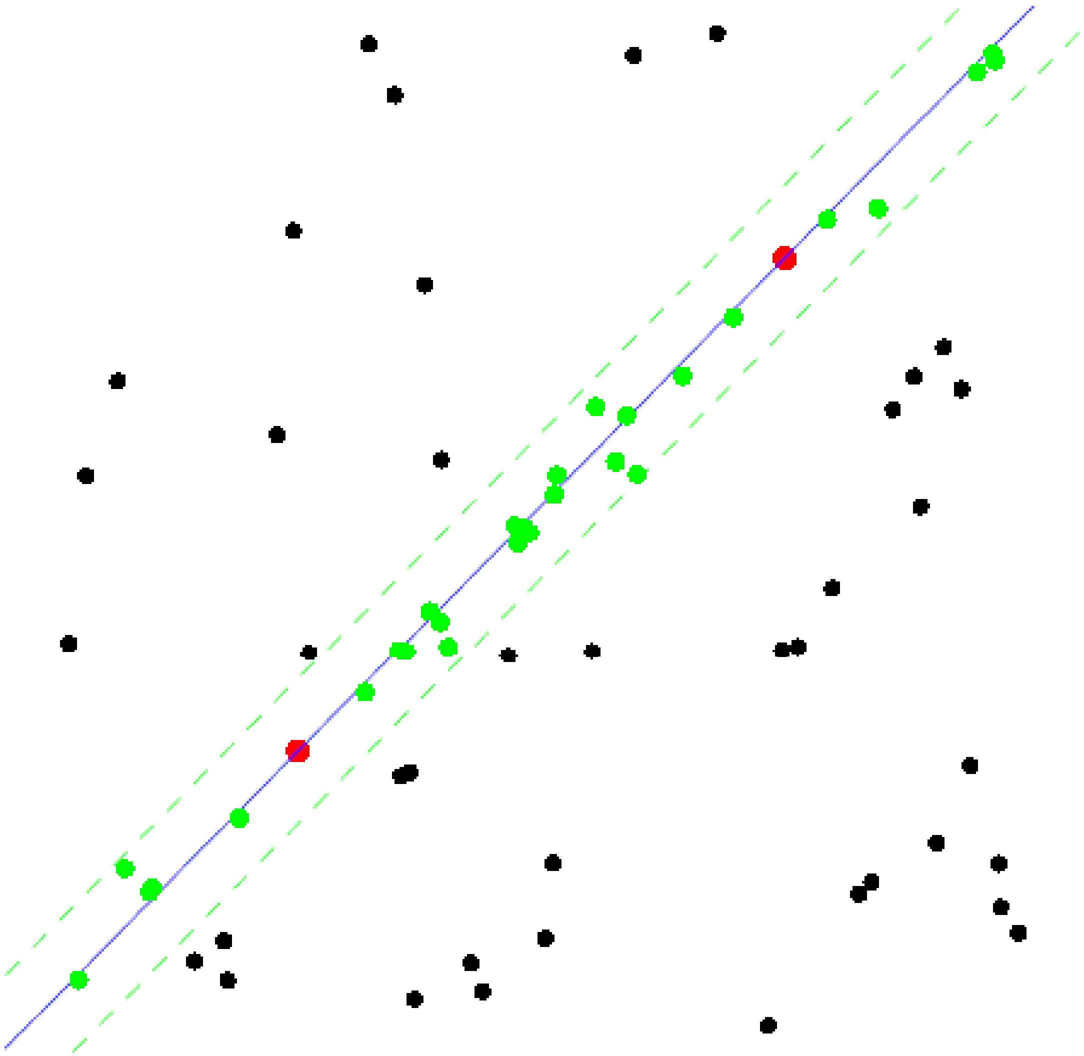
$c=3$

Another trial



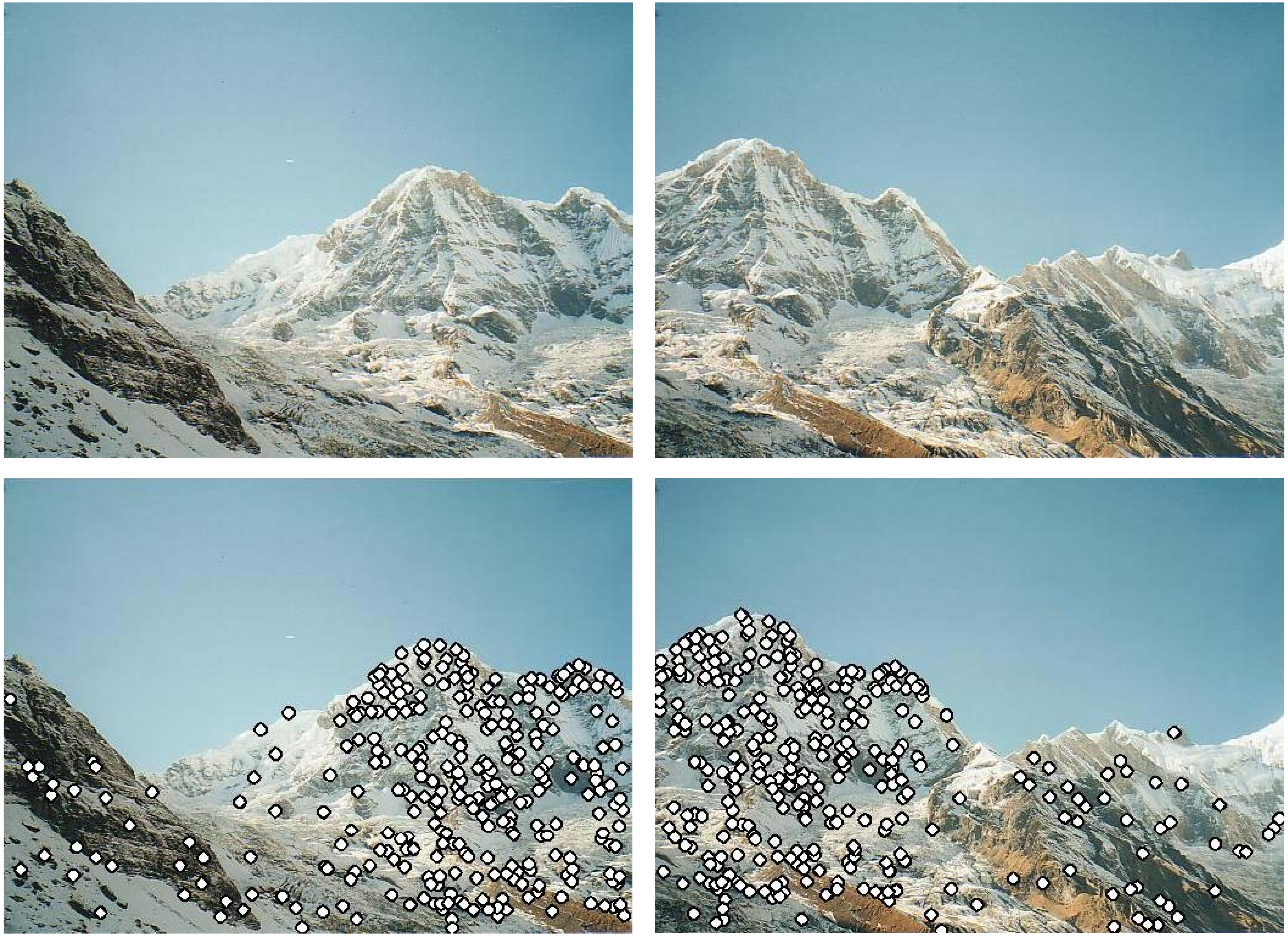
$c=3$

The best model

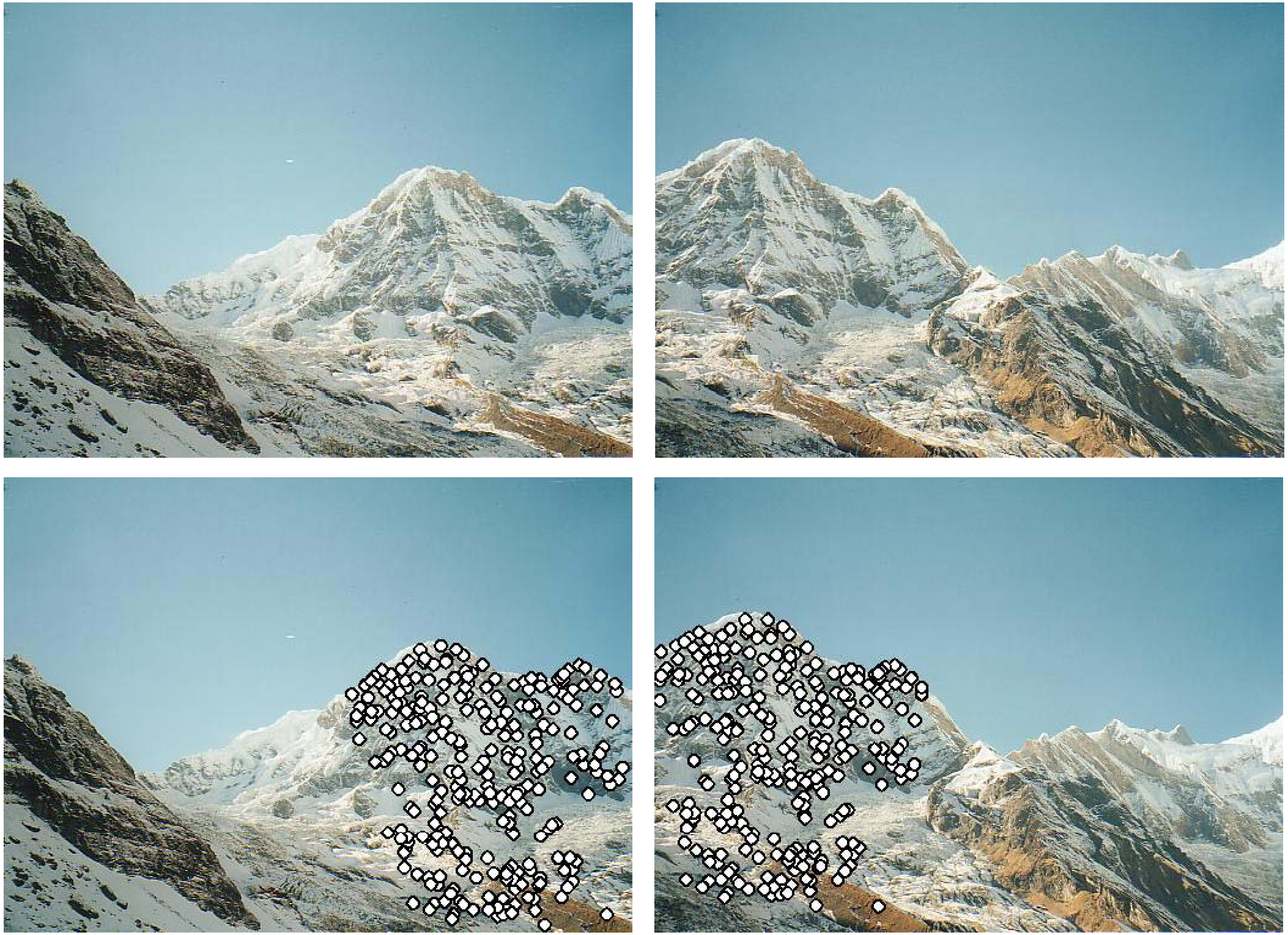


$c=15$

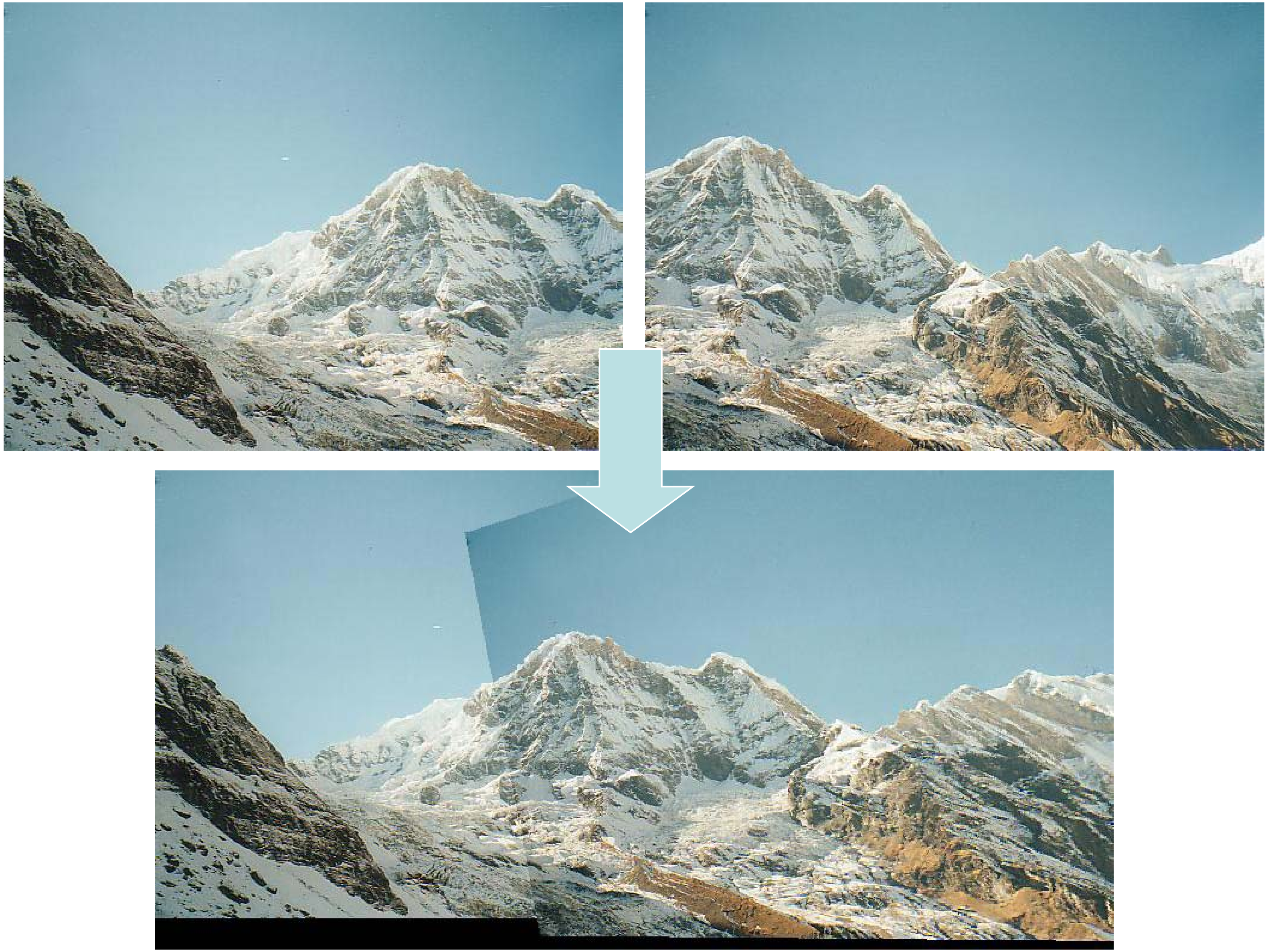
RANSAC for Homography



RANSAC for Homography



RANSAC for Homography



Applications of panorama in VFX

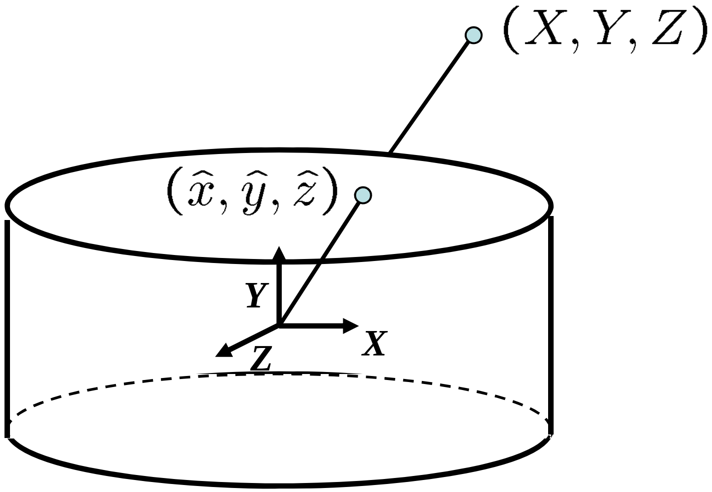
- Background plates
- Image-based lighting

Troy (image-based lighting)

http://www.cgnetworks.com/story_custom.php?story_id=2195&page=4

Spiderman 2 (background plate)

Cylindrical projection



unit cylinder

- Map 3D point (X,Y,Z) onto a unit cylinder

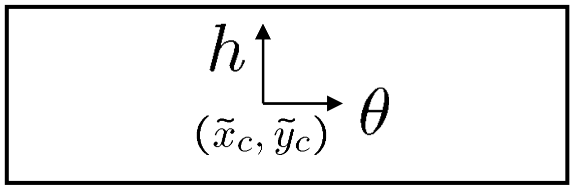
$$(\hat{x},\hat{y},\hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X,Y,Z)$$

- Convert to cylindrical coordinates

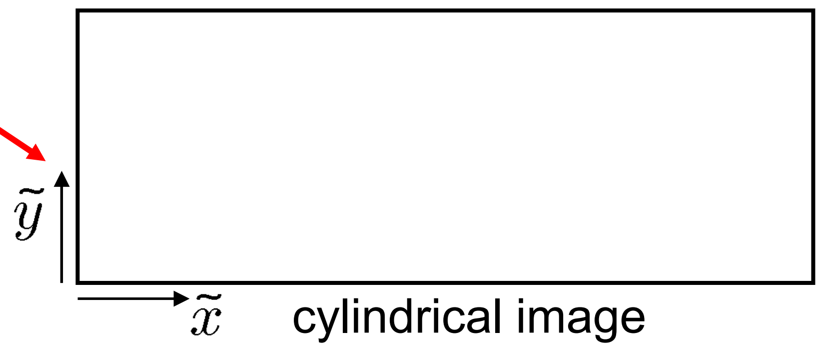
$$(sin\theta, h, cos\theta) = (\hat{x},\hat{y},\hat{z})$$

- Convert to cylindrical image coordinates

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

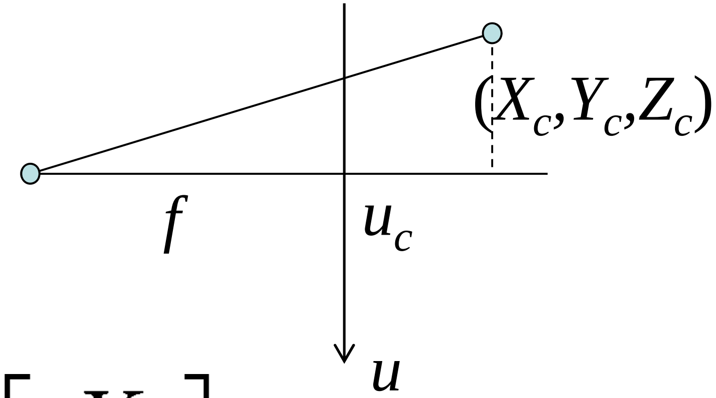


unwrapped cylinder



cylindrical image

3D → 2D perspective projection



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Reference

- Richard Szeliski, Image Alignment and Stitching, unpublished draft, 2005.
- R. Szeliski and H.-Y. Shum. Creating full view panoramic image mosaics and texture-mapped models, SIGGRAPH 1997, pp251-258.
- M. Brown, D. G. Lowe, Recognising Panoramas, ICCV 2003.

Direct vs feature-based

- Direct methods use all information and can be very accurate, but they depend on the fragile "brightness constancy" assumption
- Iterative approaches require initialization
- Not robust to illumination change and noise images
- In early days, direct method is better.

- Feature based methods are now more robust and potentially faster
- Even better, it can recognize panorama without initialization

TODO

- Bundle adjustment
- LM method
- Direct method vs feature-based method

- Frame-rate image alignment for stabilization

- Rick’s CGA 1995 paper? LM method

Project #2 Image stitching

- camera availability
- Tripod?
- http://www.tawbaware.com/maxlyons/
- http://www.cs.washington.edu/education/courses/cse590ss/CurrentQtr/projects.htm
- http://www.cs.ubc.ca/~mbrown/panorama/panorama.html

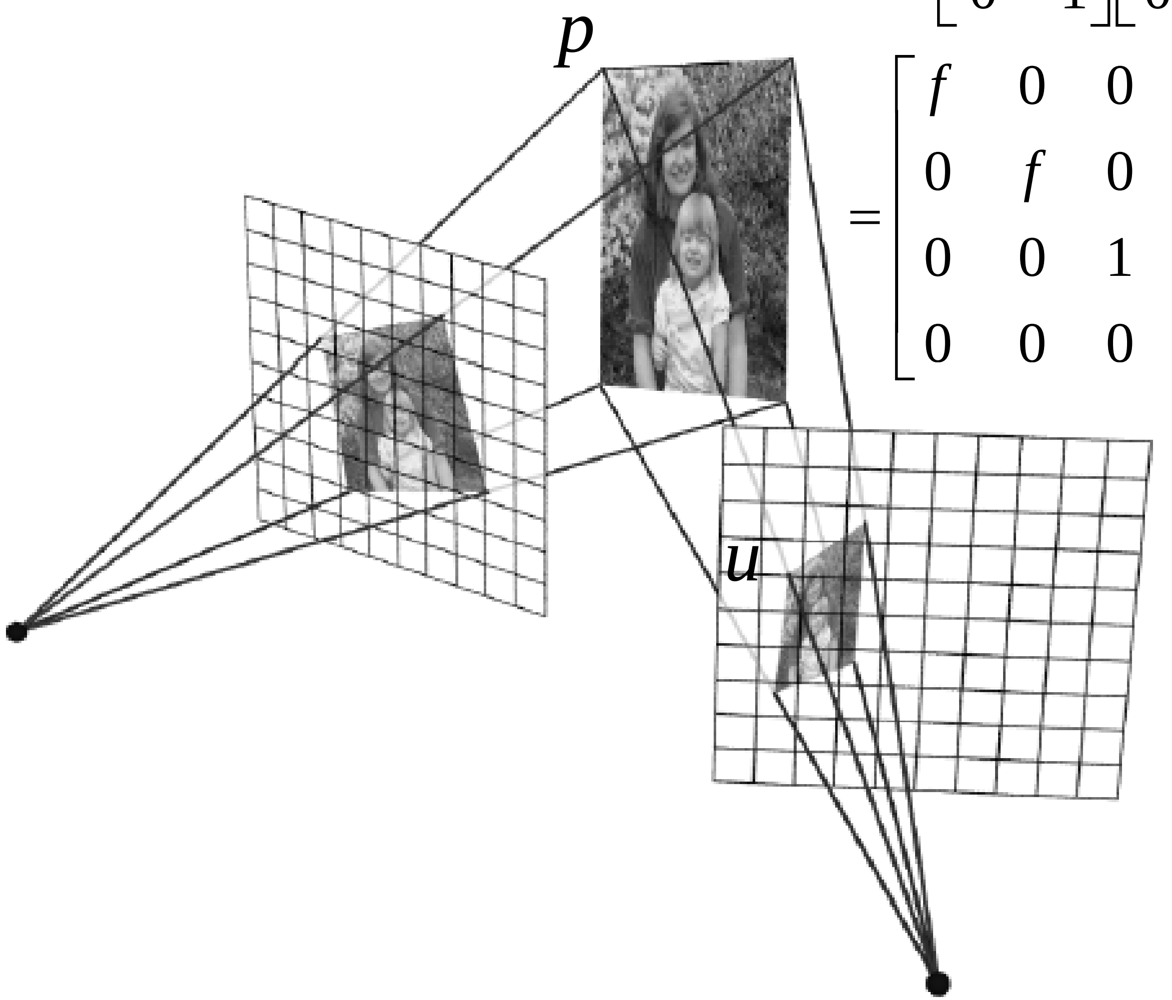
blending

- Alpha-blending
- Photomontage
- Poisson blending
- Adelson’s pyramid blending
- Hdr?

3D interpretation

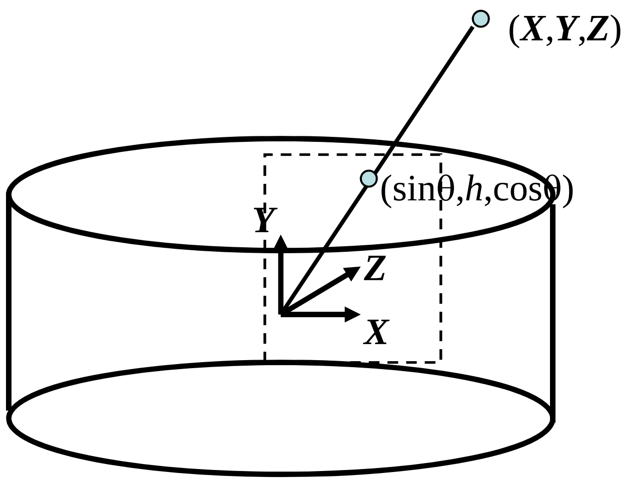
$$u = \begin{bmatrix} K & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} p$$

$$= \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



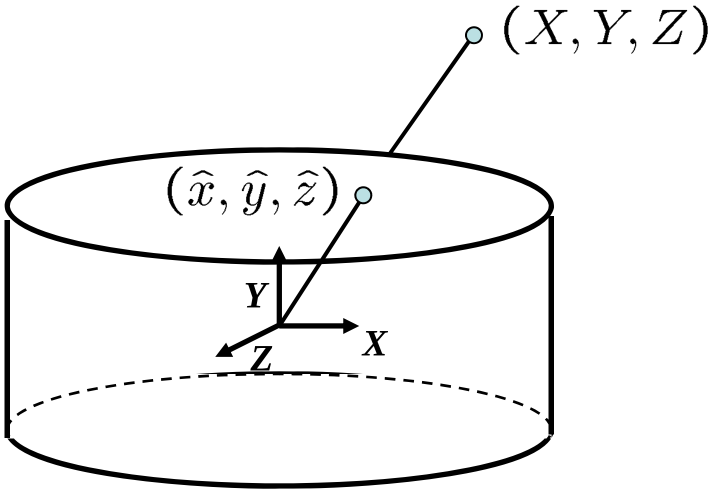
Cylindrical warping

- Given focal length *f* and image center (x_c, y_c)

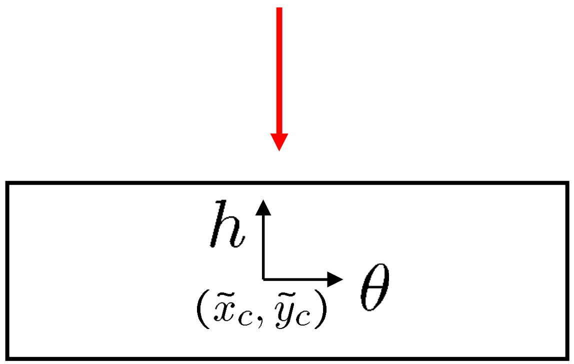


$$\theta = (x_{cyl} - x_c)/f$$
$$h = (y_{cyl} - y_c)/f$$
$$\hat{x} = \sin\theta$$
$$\hat{y} = h$$
$$\hat{z} = \cos\theta$$
$$x = f\hat{x}/\hat{z} + x_c$$
$$y = f\hat{y}/\hat{z} + y_c$$

Cylindrical projection



unit cylinder



unwrapped cylinder

- Map 3D point (X,Y,Z) onto cylinder

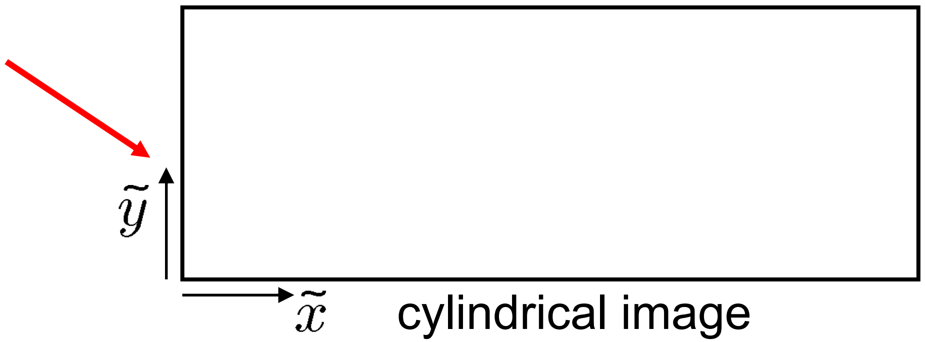
$$(\hat{x},\hat{y},\hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X,Y,Z)$$

- Convert to cylindrical coordinates

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

- Convert to cylindrical image coordinates

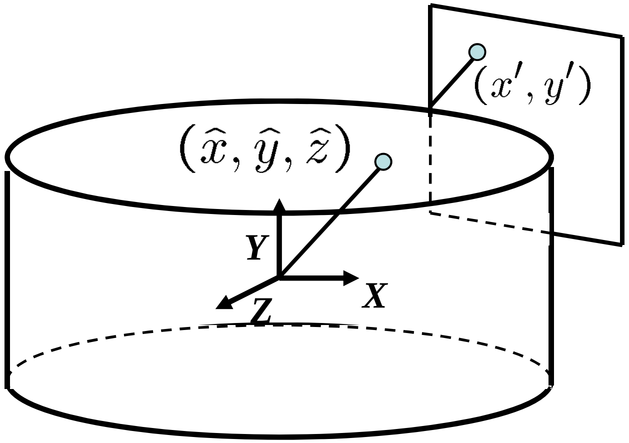
$$(\tilde{x},\tilde{y}) = (f\theta, fh) + (\tilde{x}_c,\tilde{y}_c)$$



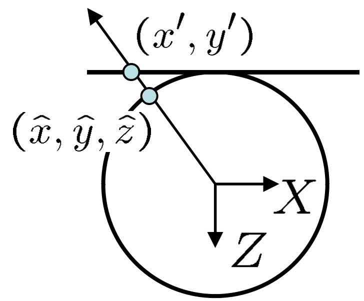
cylindrical image

Cylindrical reprojection

- How to map from a cylinder to a planar image?



side view



top-down view

- Apply camera projection matrix
  - *w* = image width, *h* = image height

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} -f & 0 & w/2 & 0 \\ 0 & -f & h/2 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ 1 \end{bmatrix}$$

- Convert to image coordinates
  - divide by third coordinate (w)

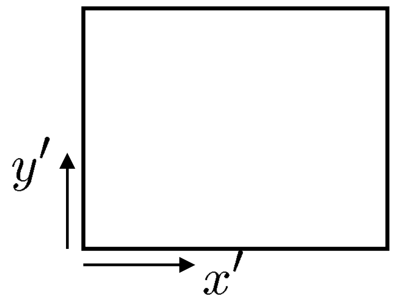
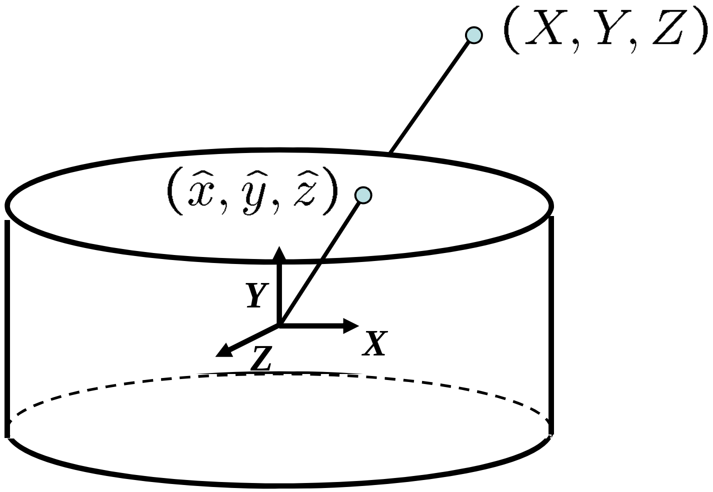
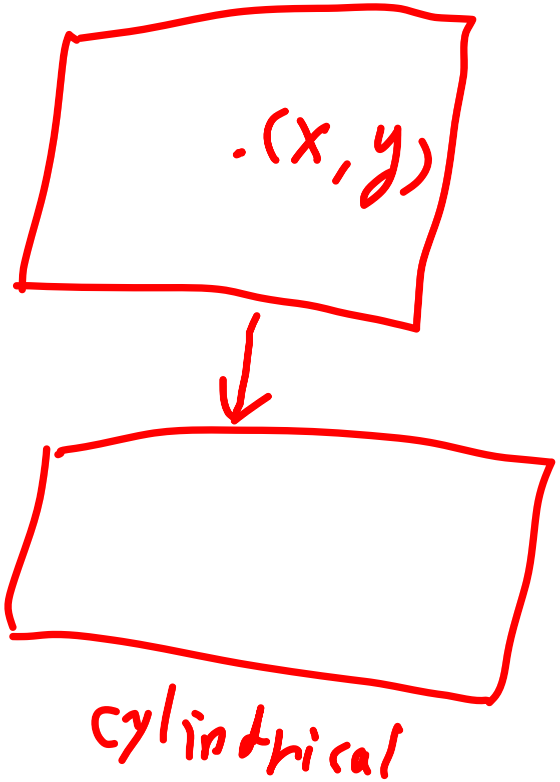


image coords

Cylindrical projection



unit cylinder



$$\hat{x} = \sin\theta$$

Levenberg-Marquardt Method

Alignment

- a rotation of the camera is a **translation** of the cylinder!

$$\begin{bmatrix} \sum_{x,y} I_x^2 & \sum_{x,y} I_x I_y \\ \sum_{x,y} I_x I_y & \sum_{x,y} I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_{x,y} I_x \left(J(x,y) - I(x,y) \right) \\ \sum_{x,y} I_y \left(J(x,y) - I(x,y) \right) \end{bmatrix}$$

LucasKanadeStep

```
void LucasKanadeStep(CByteImage& img1, CByteImage& img2, float t[2]) {
    // Transform the image
    Translation(img2, img2t, t);

    // Compute the gradients and summed error by comparing img1 and img2t
    double A[2][2], b[2];
    for (int y = 1; y < height-1; y++)  {      // ignore borders
        for (int x = 1; x < width-1; x++) {
            // If both have full alphas, then compute and accumulate the error
            double e = img2t.Pixel(x, y, k) - img1.Pixel (x, y, k);
            // Accumulate the matrix entries
            double gx = 0.5*(img2t.Pixel(x+1, y, k) - img2t.Pixel(x-1, y, k));
            double gy = 0.5*(img2t.Pixel(x, y+1, k) - img2t.Pixel(x, y-1, k));

            A[0][0] += gx*gx; A[0][1] += gx*gy;
            A[1][0] += gx*gy; A[1][1] += gy*gy;

            b[0] += e*gx; b[1] += e*gy;
        }
    }
```

LucasKanadeStep (cont.)

```
    // Solve for the update At=b and update the vector

    double det = 1.0 / (A[0][0]*A[1][1] - A[1][0]*A[1][0]);

    t[0] += (A[1][1]*b[0] - A[1][0]*b[1]) * det;
    t[1] += (A[0][0]*b[1] - A[1][0]*b[0]) * det;
}
```

PyramidLucasKanade

```
void PyramidalLucasKanade(CByteImage& img1, CByteImage& img2, float t[2],
                          int nLevels, int nLucasKanadeSteps)
{
   CBytePyramid p1(img1);    // Form the two pyramids
   CBytePyramid p2(img2);

   // Process in a coarse-to-fine hierarchy
   for (int l = nLevels-1; l >= 0; l--)
   {
      t[0] /= (1 << l);   // scale the t vector
      t[1] /= (1 << l);
      CByteImage& i1 = p1[l];
      CByteImage& i2 = p2[l];

      for (int k = 0; k < nLucasKanadeSteps; k++)
            LucasKanadeStep(i1, i2, t);
      t[0] *= (1 << l);   // restore the full scaling
      t[1] *= (1 << l);
   }
}
```

Gaussian pyramid

2D Motion models

- translation: $x' = x + t$ $\quad x = (x,y)$
- rotation: $x' = R\,x + t$
- similarity: $x' = s\,R\,x + t$
- affine: $x' = A\,x + t$
- perspective: $\underline{x}' \cong H\,\underline{x}$ $\quad \underline{x} = (x,y,1)$
  ($\underline{x}$ is a *homogeneous* coordinate)
- These all form a nested *group* (closed under composition w/ inv.)

Video matting

alpha matte

Recognising Panoramas

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

Recognising Panoramas

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

Recognising Panoramas

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

Recognising Panoramas

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

- 2D Rotations (q, f)
  - Ordering $\nRightarrow$ matching images

Recognising Panoramas

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

- 2D Rotations (q, f)
  - Ordering $\nRightarrow$ matching images

Recognising Panoramas

- 1D Rotations (θ)
  - Ordering $\Rightarrow$ matching images

- 2D Rotations (q, f)
  - Ordering $\not\Rightarrow$ matching images

Probabilistic model for verification

- Compare probability that this set of RANSAC inliers/outliers was generated by a correct/false image match

- Choosing values for p_1, p_0 and p_{min}

$$n_i > 5.9 + 0.22 n_f$$

Recognising Panoramas

Overview

- SIFT Feature Matching
- Image Matching
- Bundle Adjustment
- Multi-band Blending

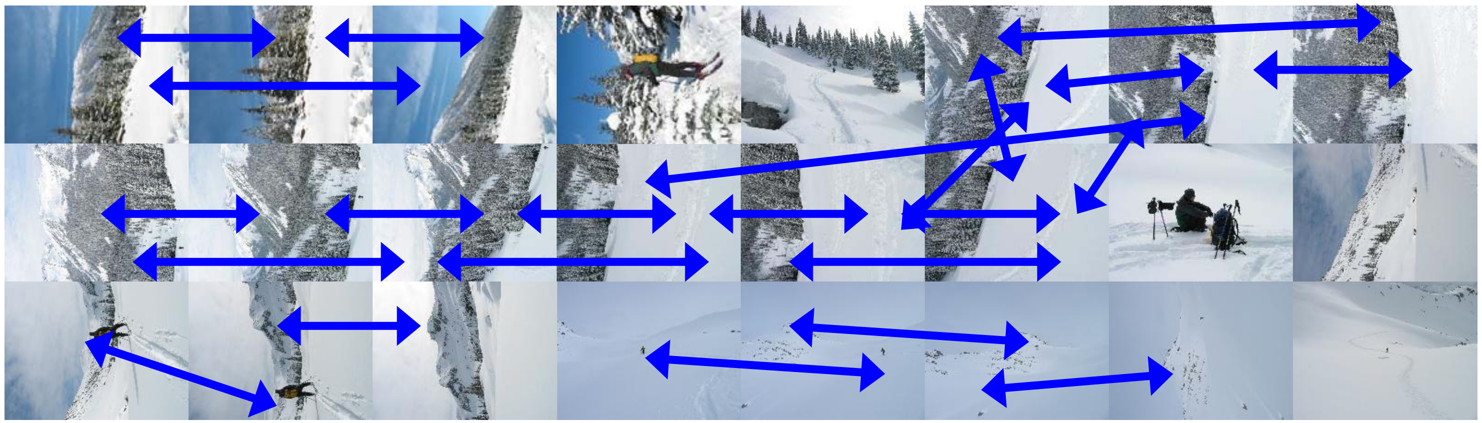
Nearest Neighbour Matching

- Find k-NN for each feature
  - $k \approx$ number of overlapping images (we use $k = 4$)
- Use k-d tree
  - k-d tree recursively bi-partitions data at mean in the dimension of maximum variance
  - Approximate nearest neighbours found in $O(n\log n)$

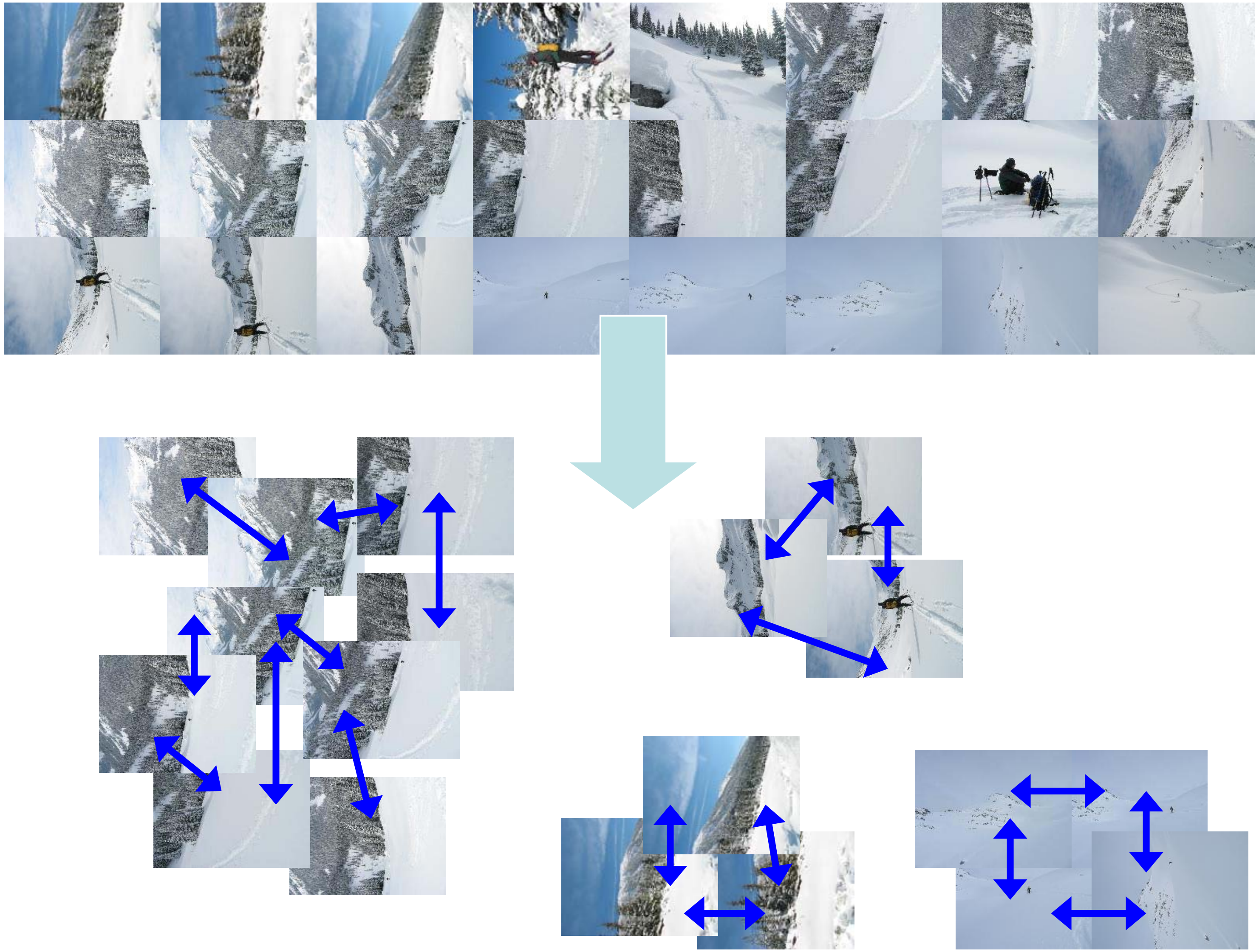
Overview

- SIFT Feature Matching
- Image Matching
  - For each image, use RANSAC to select inlier features from 6 images with most feature matches
- Bundle Adjustment
- Multi-band Blending

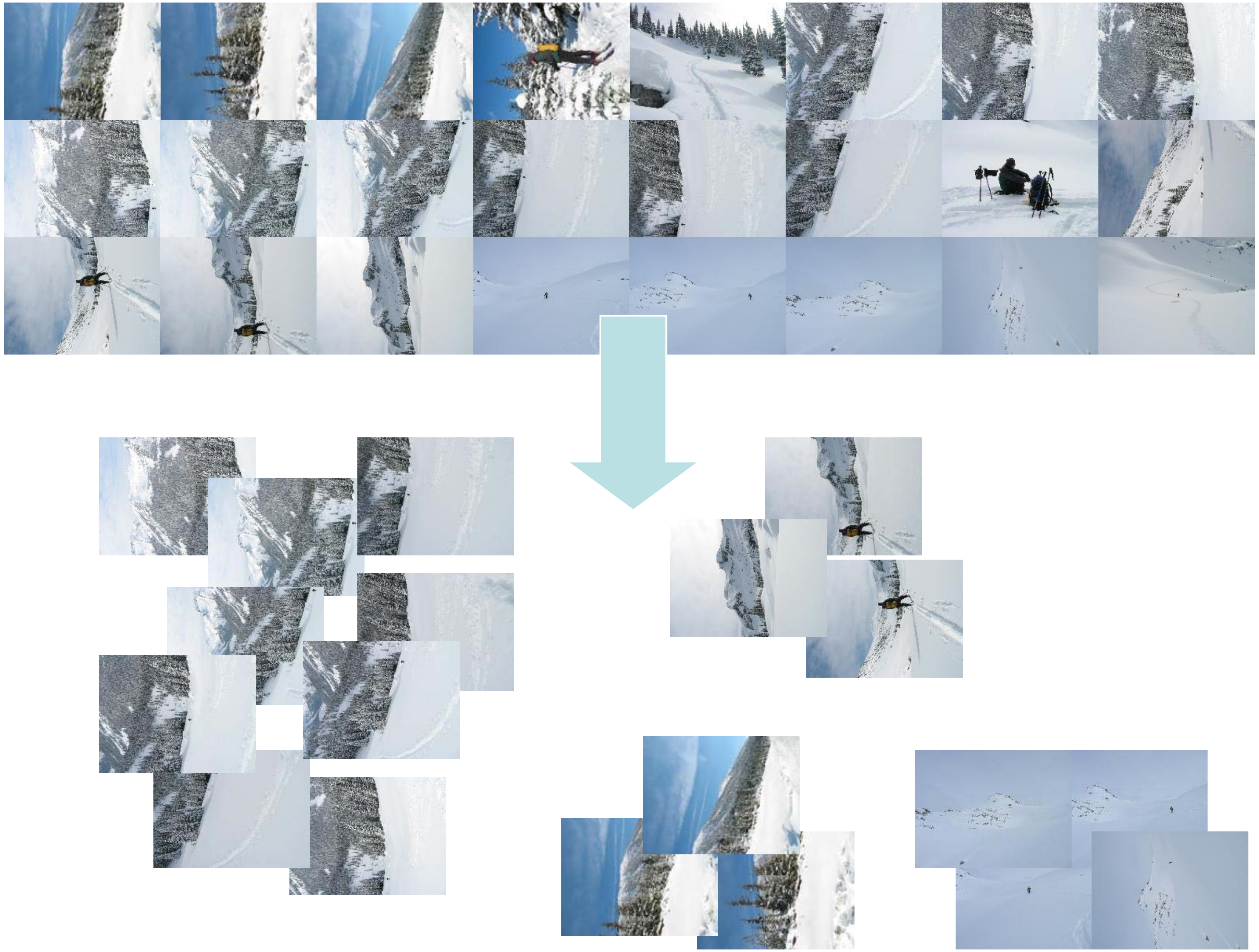
Finding the panoramas



Finding the panoramas



Finding the panoramas



Finding the panoramas

Overview

- SIFT Feature Matching
- Image Matching
- Bundle Adjustment
- Multi-band Blending

Homography for Rotation

- Parameterise each camera by rotation and focal length

$$\mathbf{R}_i = e^{[\boldsymbol{\theta}_i]_\times}, \;\; [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}$$

$$\mathbf{K}_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- This gives pairwise homographies

$$\tilde{\mathbf{u}}_i = \mathbf{H}_{ij}\tilde{\mathbf{u}}_j \, , \;\; \mathbf{H}_{ij} = \mathbf{K}_i \mathbf{R}_i \mathbf{R}_j^T \mathbf{K}_j^{-1}$$

Error function

- Sum of squared projection errors

$$e = \sum_{i=1}^{n} \sum_{j \epsilon \mathcal{I}(i)} \sum_{k \epsilon \mathcal{F}(i,j)} f(\mathbf{r}_{ij}^{k})^2$$

  - n = #images
  - I(i) = set of image matches to image i
  - F(i, j) = set of feature matches between images i,j
  - r_{ij}^{k} = residual of k^{th} feature match between images i,j

- Robust error function

$$f(\mathbf{x}) = \begin{cases} |\mathbf{x}|, & \text{if } |\mathbf{x}| < x_{max} \\ x_{max}, & \text{if } |\mathbf{x}| \geq x_{max} \end{cases}$$

Overview

- SIFT Feature Matching
- Image Matching
- Bundle Adjustment
- Multi-band Blending

Multi-band Blending

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

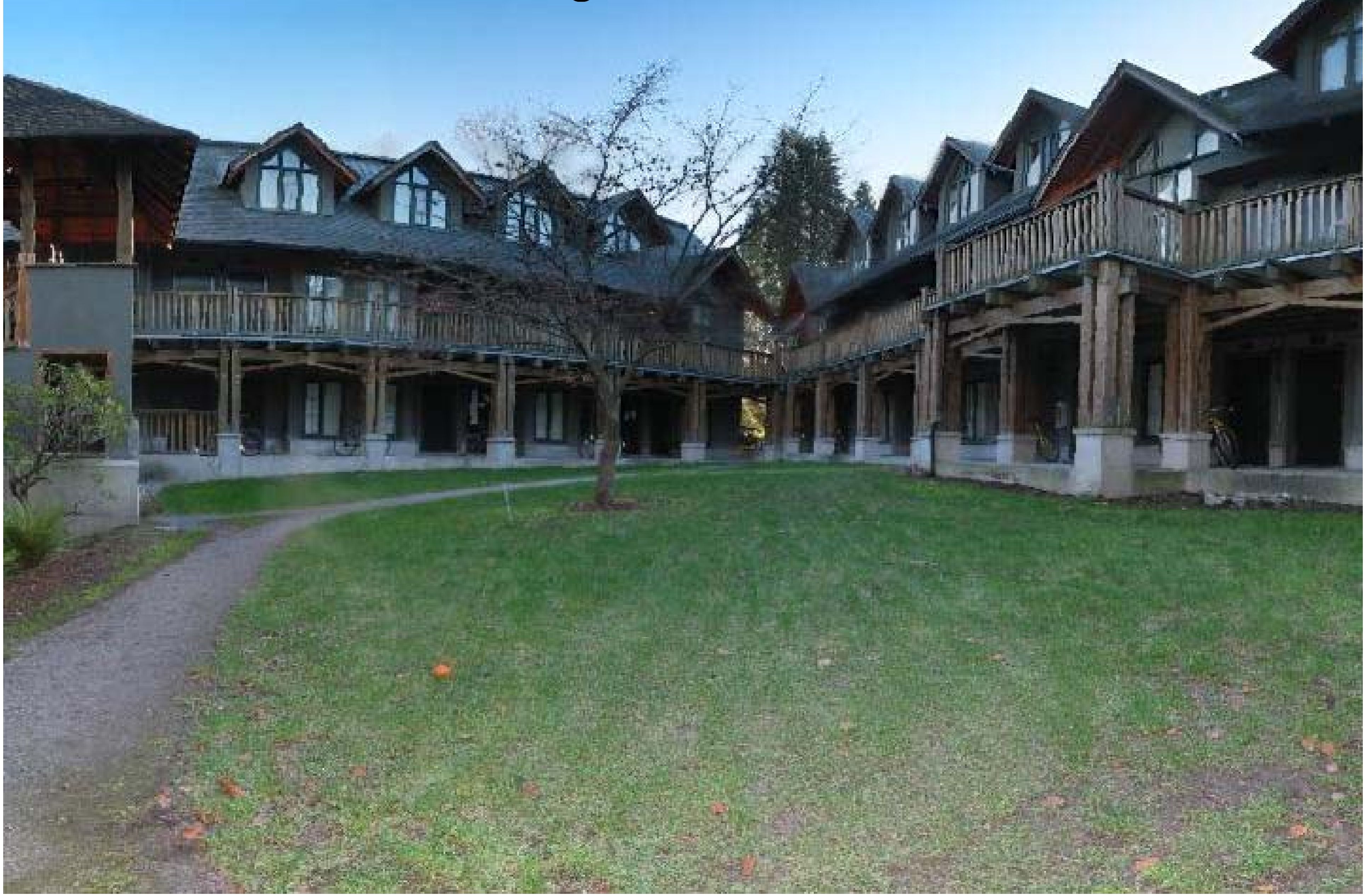
2-band Blending

Low frequency (λ > 2 pixels)

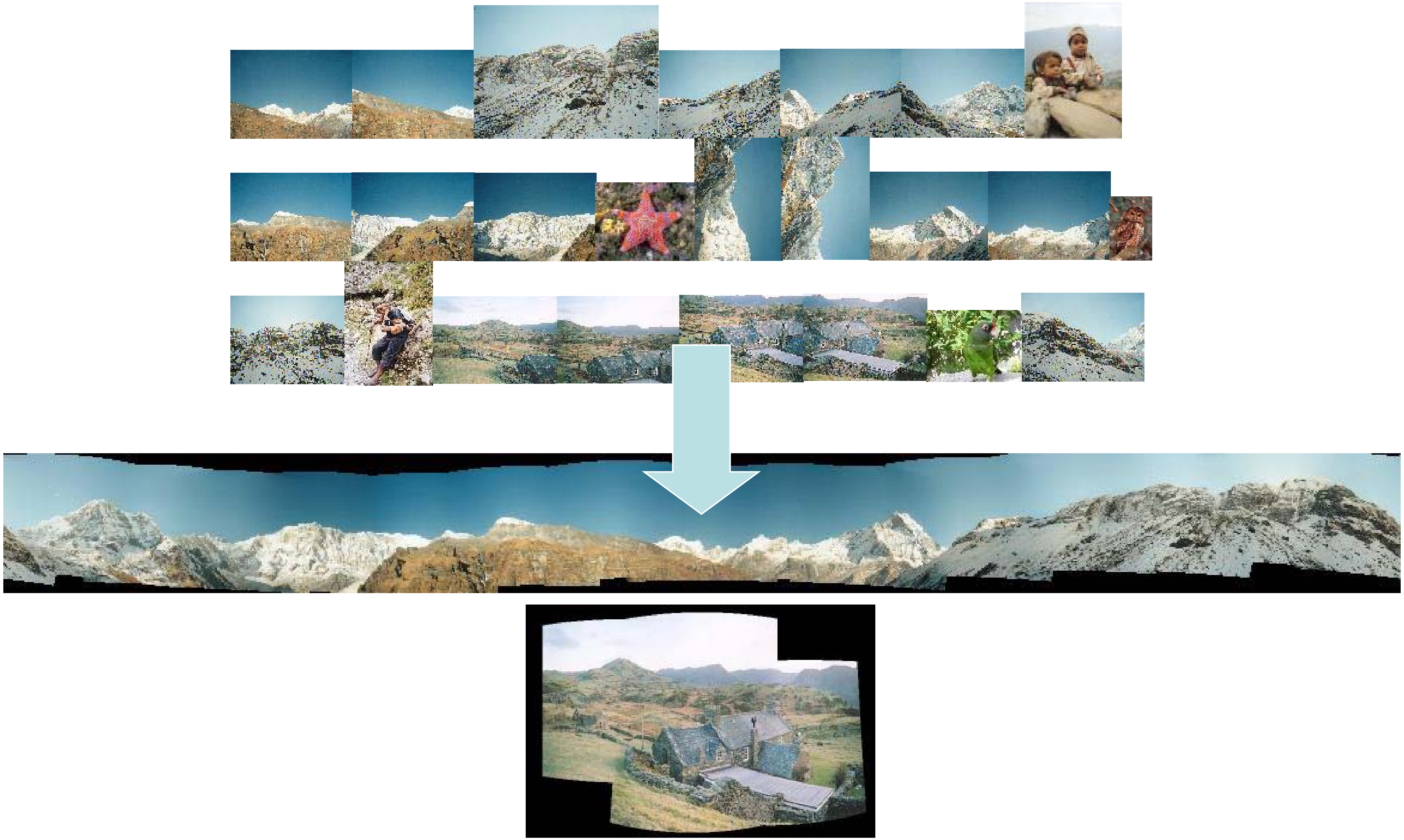
High frequency (λ < 2 pixels)

Linear Blending

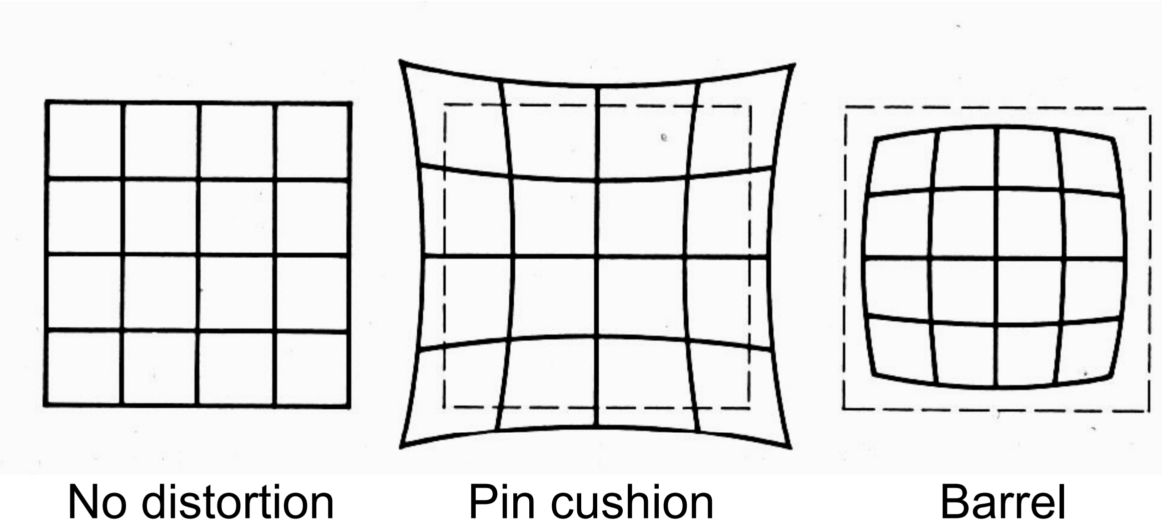
2-band Blending



Results



Distortion



- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens

Radial correction

- Correct for "bending" in wide field of view lenses

$$\begin{aligned}
\hat{r}^2 &= \hat{x}^2 + \hat{y}^2 \\
\hat{x}' &= \hat{x}/(1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
\hat{y}' &= \hat{y}/(1 + \kappa_1 \hat{r}^2 + \kappa_2 \hat{r}^4) \\
x &= f\hat{x}'/\hat{z} + x_c \\
y &= f\hat{y}'/\hat{z} + y_c
\end{aligned}$$