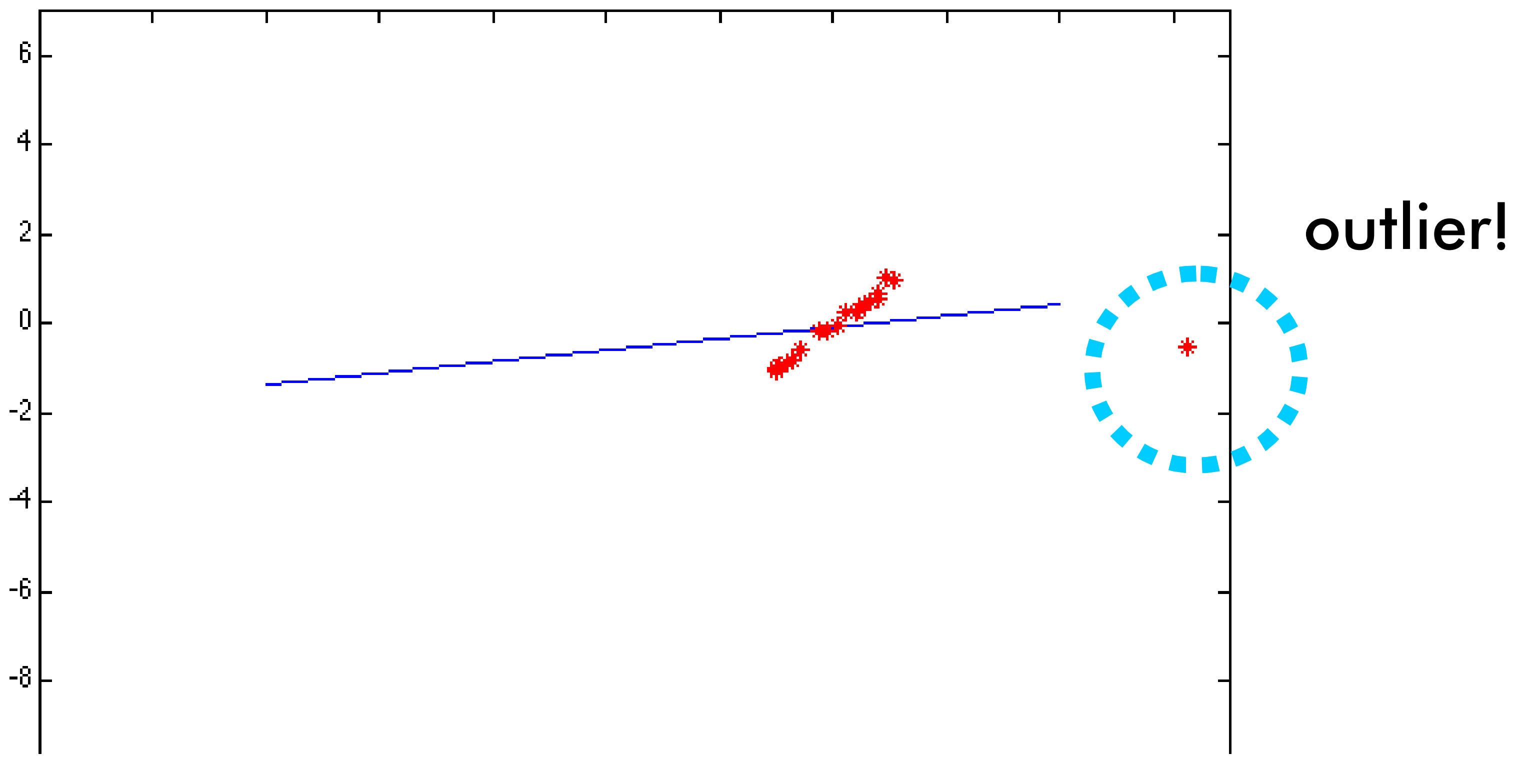
More Mosaic Madness

© Jeffrey Martin (jeffrey-martin.com)

with a lot of slides stolen from Steve Seitz and Rick Szeliski

Slides taken from Alexei Efros

Least squares: Robustness to noise



Problem: squared error heavily penalizes outliers

Silvio

Fitting

Goal: Choose a parametric model to fit a certain quantity from data

Techniques:

- Least square methods
- RANSAC
- Hough transform

Silvio

Basic philosophy
(voting scheme)

- Data elements are used to vote for one (or multiple) models

- Robust to outliers and missing data

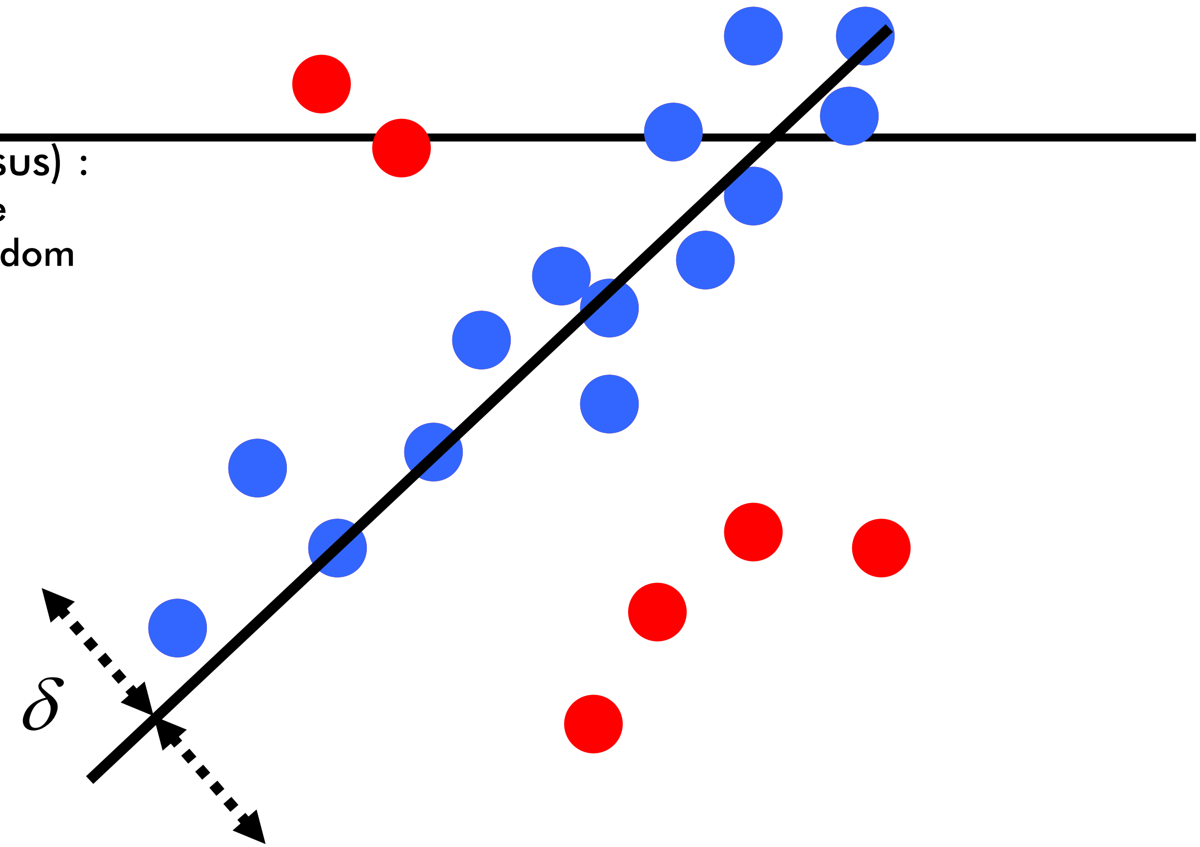
- Assumption1: Noise features will not vote consistently for any single model ("few" outliers)
- Assumption2: there are enough features to agree on a good model ("few" missing data)

Silvio

RANSAC

(RANdom SAmple Consensus) :
Learning technique to estimate parameters of a model by random sampling of observed data

Fischler & Bolles in '81.



$$\pi : \boldsymbol{I} \rightarrow \{\boldsymbol{P}, \boldsymbol{O}\}$$

$$\min_{\pi} |\boldsymbol{O}|$$

such that:

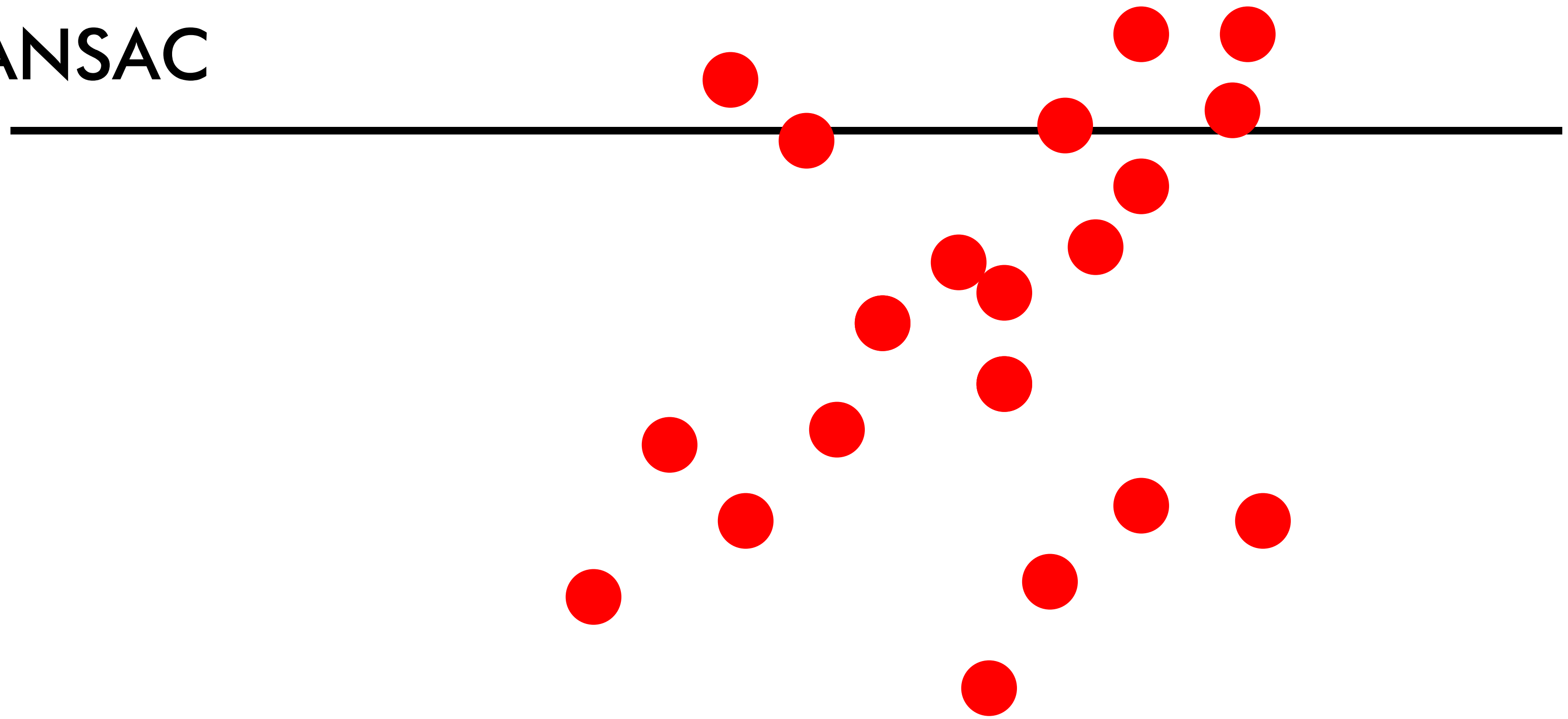
$$\boldsymbol{f}(\boldsymbol{P}, \beta) < \delta$$

Model parameters

$$\boldsymbol{f}(\boldsymbol{P}, \beta) = \left\| \beta - \left(\boldsymbol{P}^{T}\boldsymbol{P}\right)^{-1}\boldsymbol{P}^{T} \right\|$$

Silvio

RANSAC



Sample set = set of points in 2D

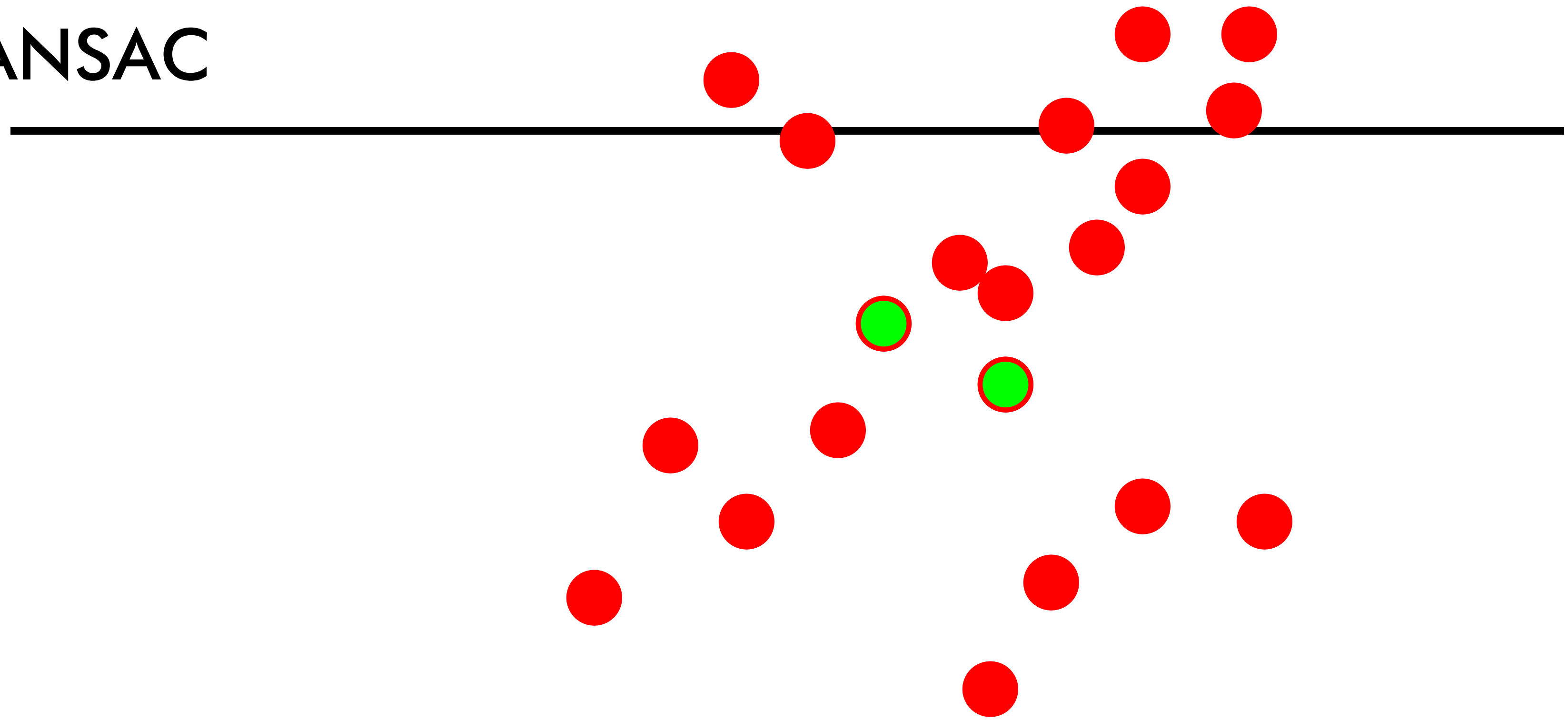
Algorithm:

1. Select random sample of minimum required size to fit model
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

Silvio

RANSAC



Sample set = set of points in 2D

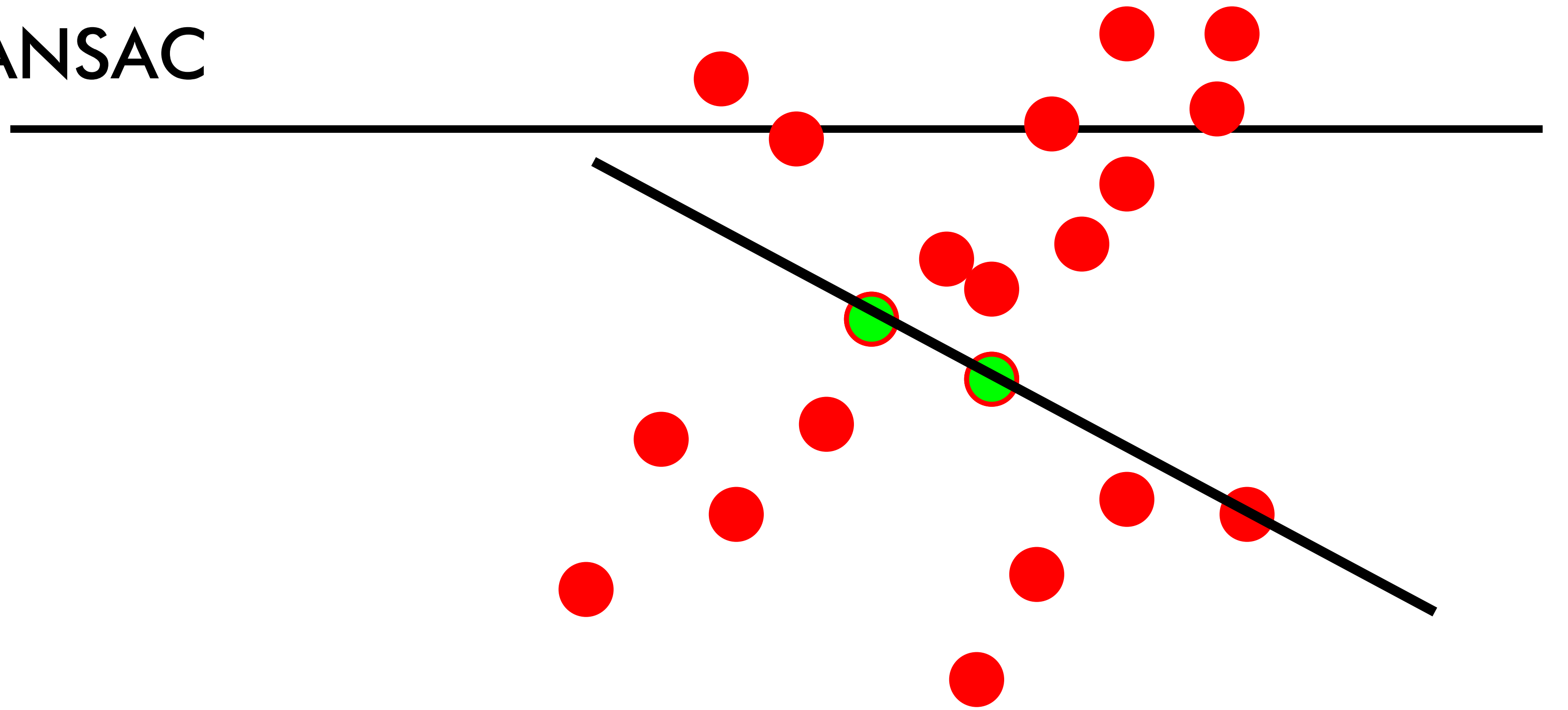
Algorithm:

1. Select random sample of minimum required size to fit model [?] = [2]
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

Silvio

RANSAC



Sample set = set of points in 2D

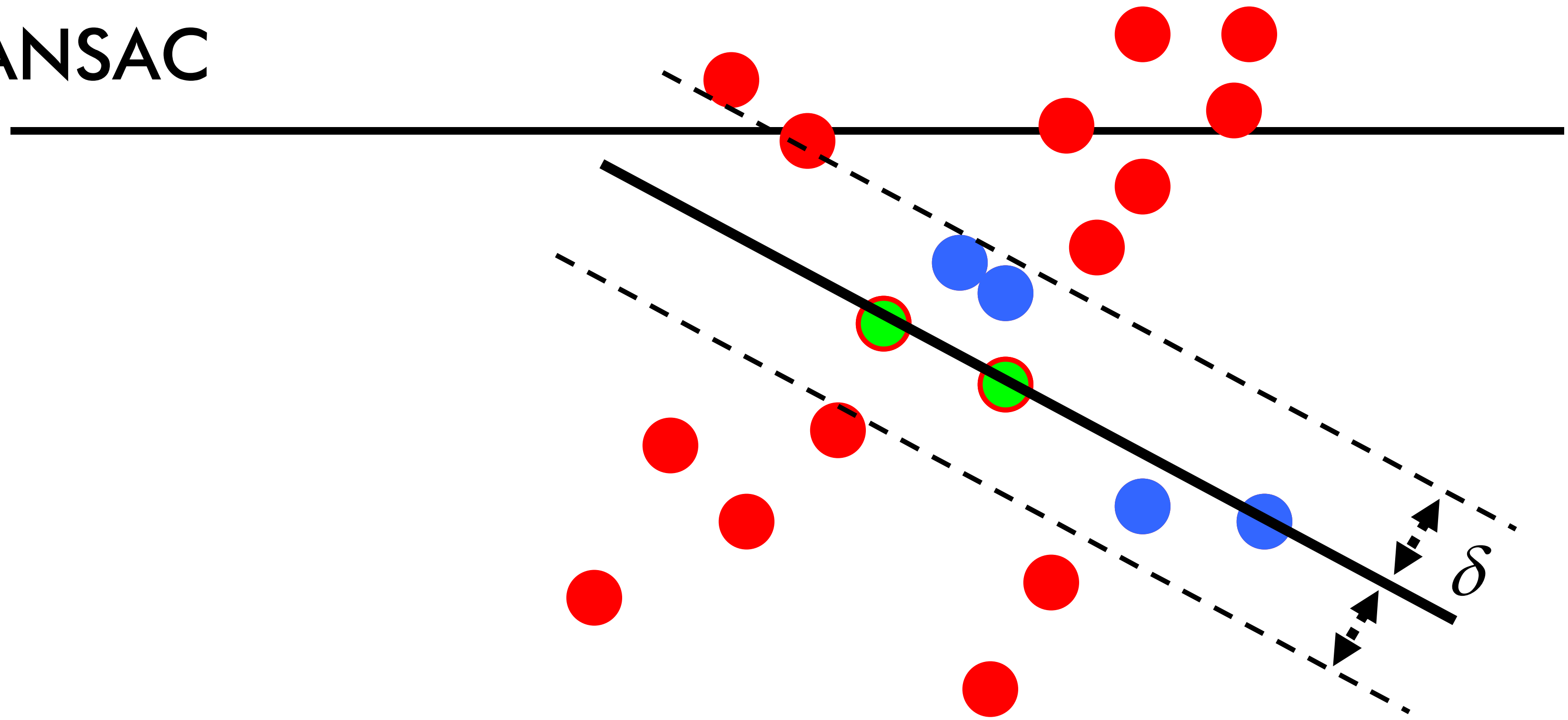
Algorithm:

1. Select random sample of minimum required size to fit model [?] = [2]
2. $\boxed{\text{Compute a putative model from sample set}}$
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

Silvio

RANSAC



Sample set = set of points in 2D

$\left| O \right| = 14$

Algorithm:

1. Select random sample of minimum required size to fit model $[?] = [2]$
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

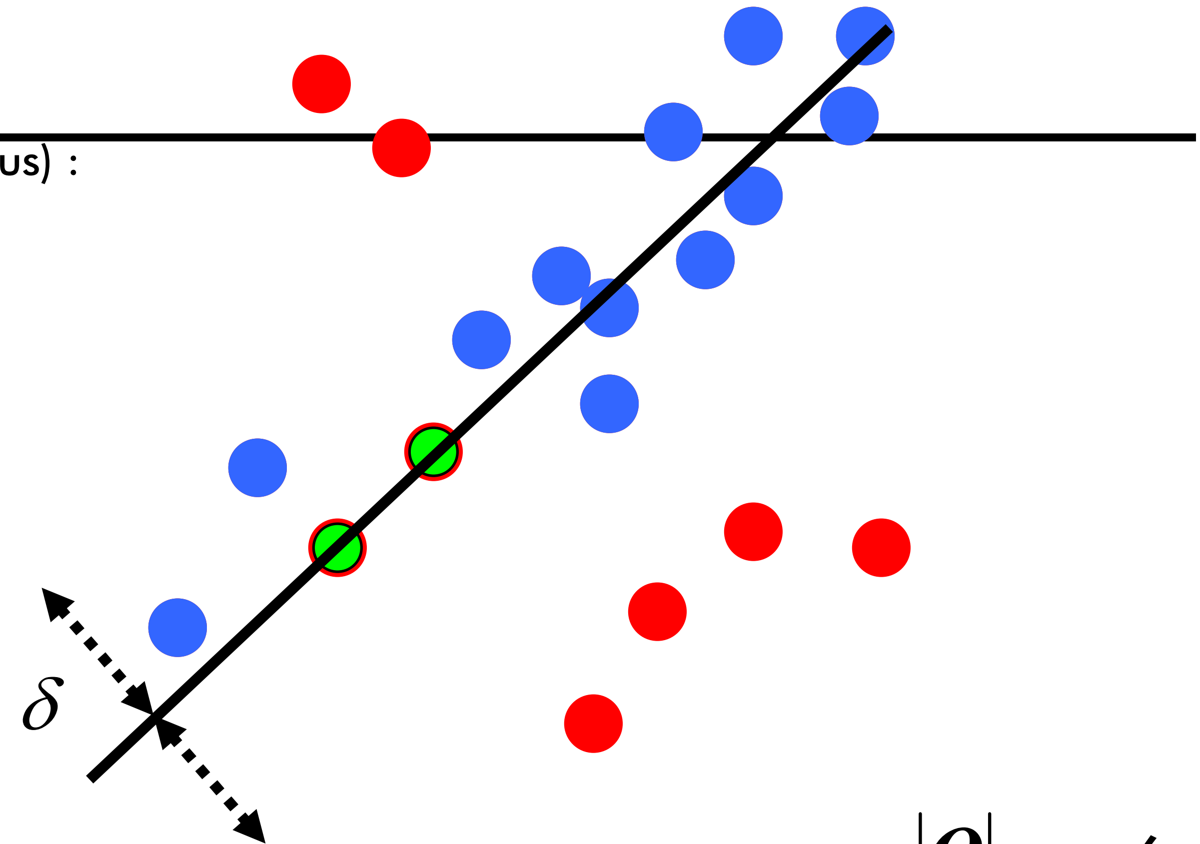
Repeat 1-3 until model with the most inliers over all samples is found

Silvio

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



$$\left| O \right| = 6$$

Algorithm:

1. Select random sample of minimum required size to fit model [?]
2. Compute a putative model from sample set
3. Compute the set of inliers to this model from whole data set

Repeat 1-3 until model with the most inliers over all samples is found

Silvio

How many samples?

- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e)
- Initial number of points s
  - Typically minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so probability for inlier is p (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \log(1-p) / \log\left(1-(1-e)^s\right)$$

	proportion of outliers e						
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: M. Pollefeys

e = probability that a point is an outlier
s = number of points in a sample
N = number of samples (we want to compute this)
p = desired probability that we get a good sample

Solve the following for N:

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

Where in the world did that come from? ….

e = probability that a point is an outlier
s = number of points in a sample
N = number of samples (we want to compute this)
p = desired probability that we get a good sample

$$1 - \left(1 - \left(\underbrace{1 - e}_{\text{Probability that choosing one point yields an inlier}}\right)^{s}\right)^{N} = p$$

e = probability that a point is an outlier

s = number of points in a sample

N = number of samples (we want to compute this)

p = desired probability that we get a good sample

$$1 - \left(1 - \underbrace{(1 - e)^{s}}\right)^{N} = p$$

Probability of choosing s inliers in a row (sample only contains inliers)

e = probability that a point is an outlier
s = number of points in a sample
N = number of samples (we want to compute this)
p = desired probability that we get a good sample

$$1 - \Big(\underbrace{1 - (1 - e)^{s}}_{\text{Probability that one or more points in the sample were outliers (sample is contaminated).}}\Big)^{N} = p$$

Probability that one or more points in the sample were outliers (sample is contaminated).

e = probability that a point is an outlier
s = number of points in a sample
N = number of samples (we want to compute this)
p = desired probability that we get a good sample

$$1 - \underbrace{\left(1 - (1 - e)^s\right)^N}_{\text{Probability that N samples were contaminated.}} = p$$

Probability that N samples were contaminated.

e = probability that a point is an outlier
s = number of points in a sample
N = number of samples (we want to compute this)
p = desired probability that we get a good sample

$$\underbrace{1 - (1 - (1 - e)^{s})^{N}}_{\substack{\text{Probability that at least one sample was not contaminated} \\ \text{(at least one sample of s points is composed of only inliers).}}} = p$$

Probability that at least one sample was not contaminated (at least one sample of s points is composed of only inliers).

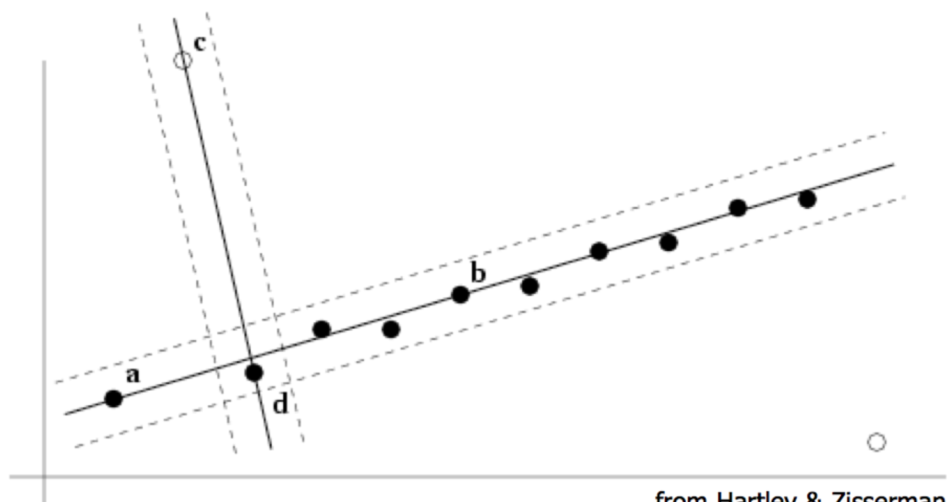
Choose N so that, with probability p, at least one random sample is free from outliers. e.g. p=0.99

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

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

	proportion of outliers e						
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

- $n = 12$ points
- Minimal sample size $s = 2$
- 2 outliers: $e = 1/6$ => 20%
- So $N = 5$ gives us a 99% chance of getting a pure-inlier sample
  - Compared to $N = 66$ by trying every pair of points



from Hartley & Zisserman

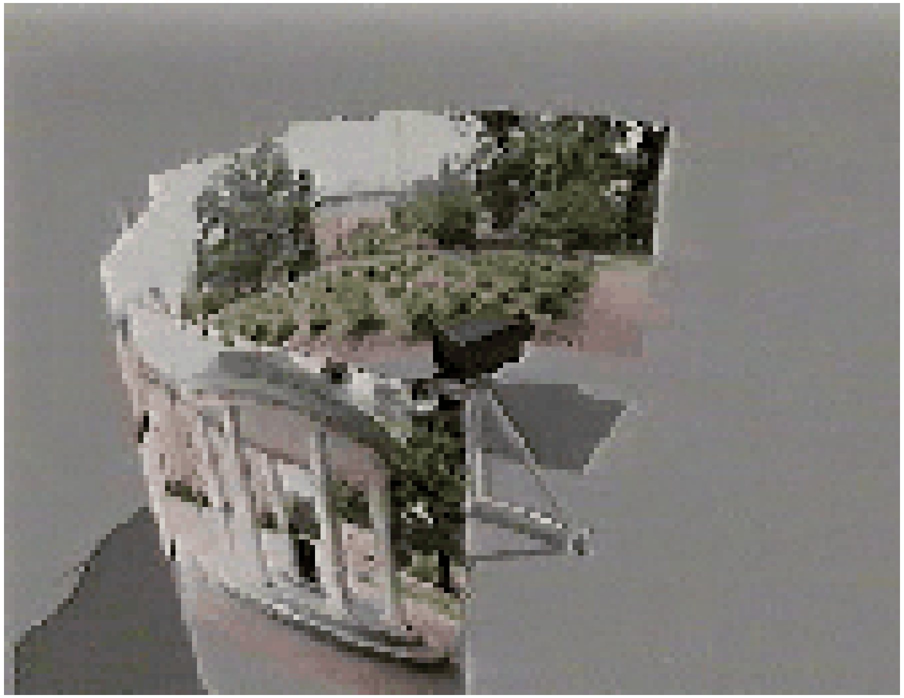
- We have seen that we don't have to exhaustively sample subsets of points, we just need to randomly sample N subsets.

- However, typically, we don't even have to sample N sets!

- Early termination: terminate when inlier ratio reaches expected ratio of inliers

$$T = (1 - e) * (\textit{total number of data points})$$

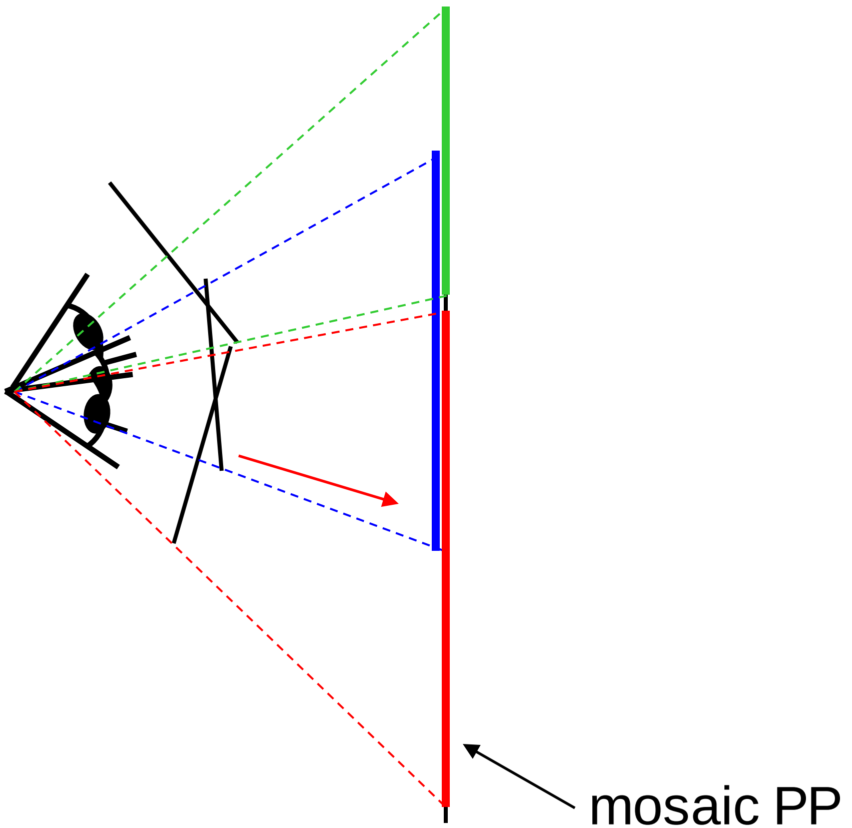
Rotation about vertical axis



What if our camera rotates on a tripod?

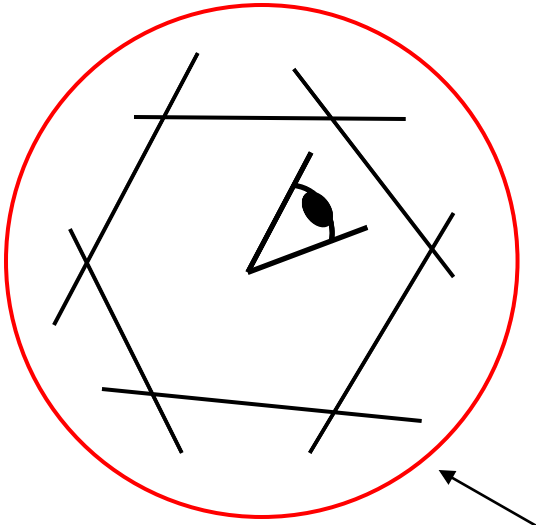
What's the structure of H?

Do we have to project onto a plane?



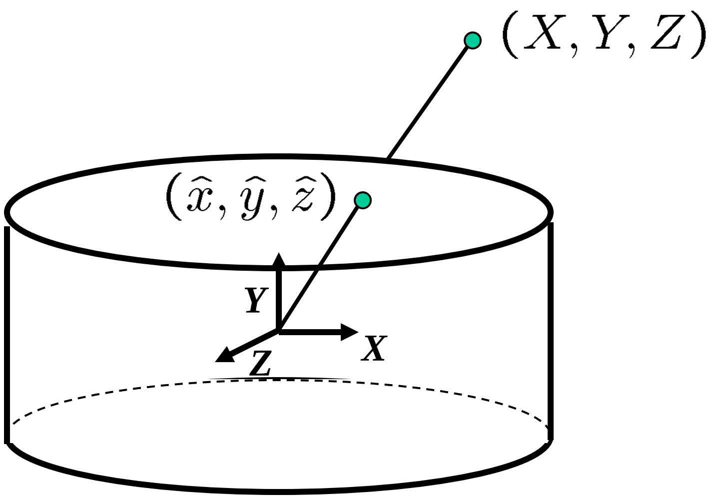
Full Panoramas

What if you want a 360° field of view?



mosaic Projection Cylinder

Cylindrical projection



unit cylinder

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

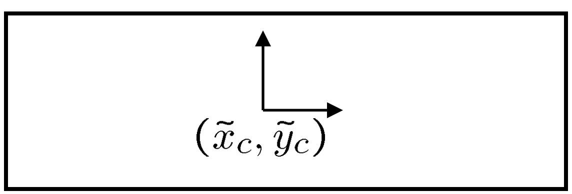
$$(\hat{x},\hat{y},\hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X,Y,Z)$$

- Convert to cylindrical coordinates

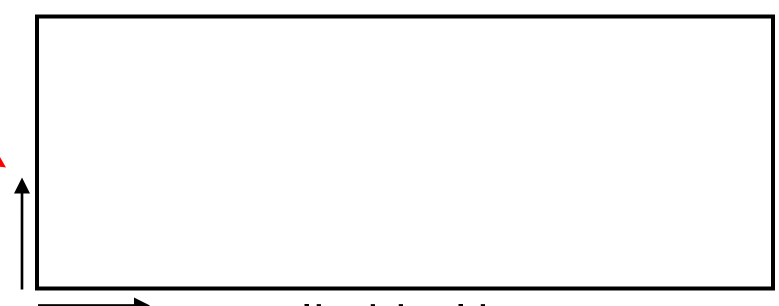
$$(sin\theta, h, cos\theta) = (\hat{x},\hat{y},\hat{z})$$

- Convert to cylindrical image coordinates

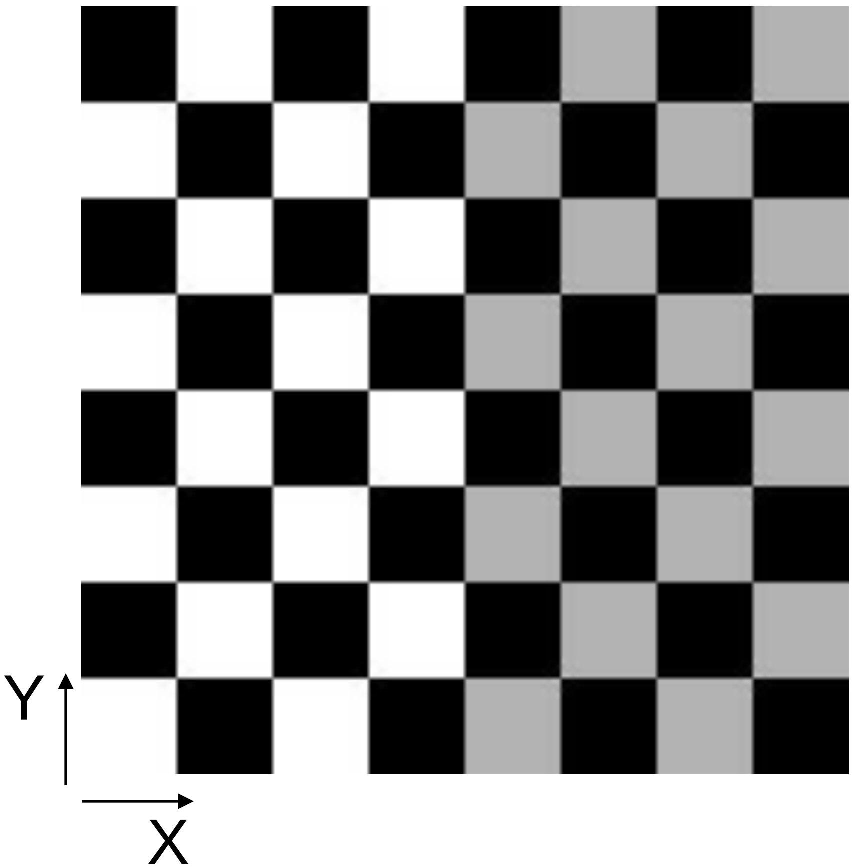
$$(\tilde{x},\tilde{y}) = (f\theta, fh) + (\tilde{x}_c,\tilde{y}_c)$$

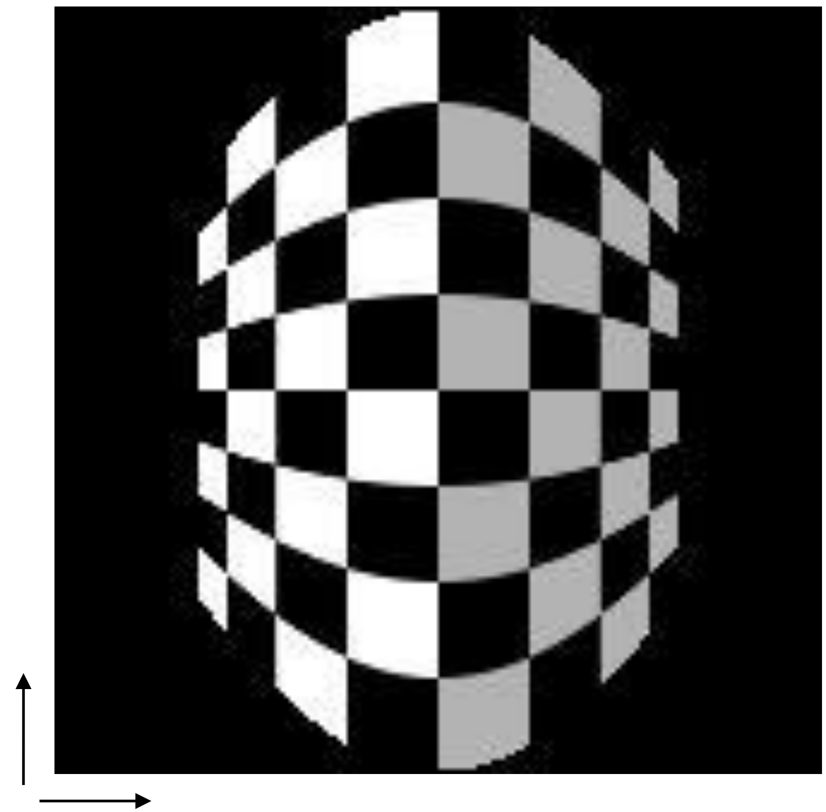


unwrapped cylinder

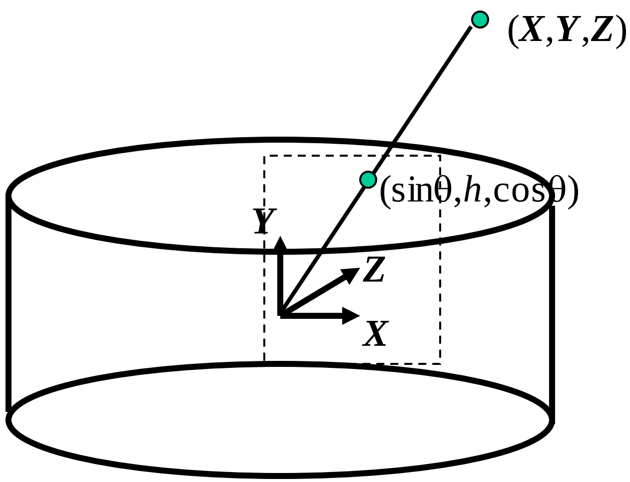

cylindrical image

Cylindrical Projection



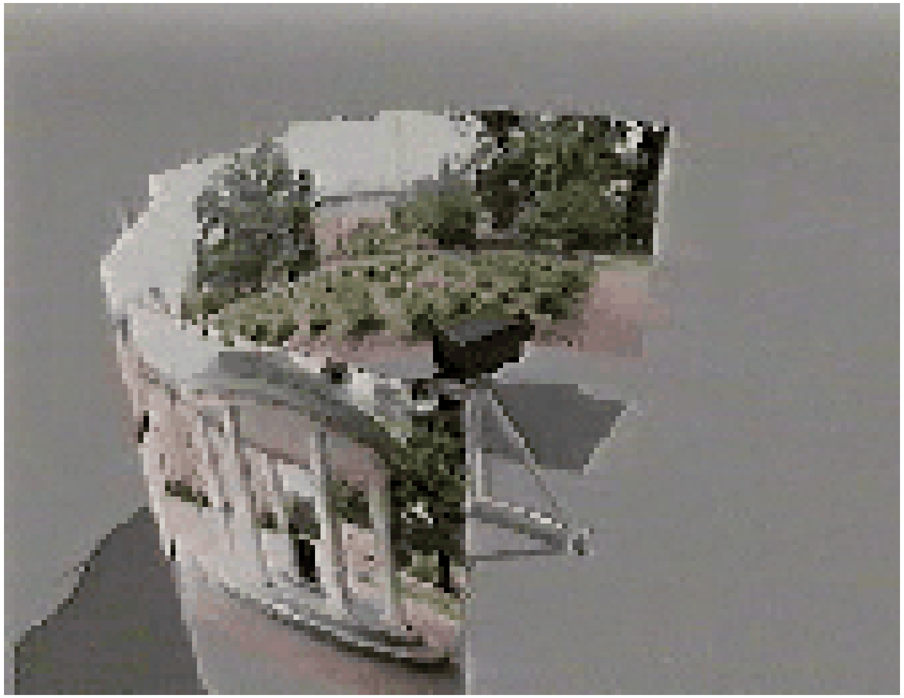


Inverse Cylindrical projection



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

Cylindrical panoramas



Steps

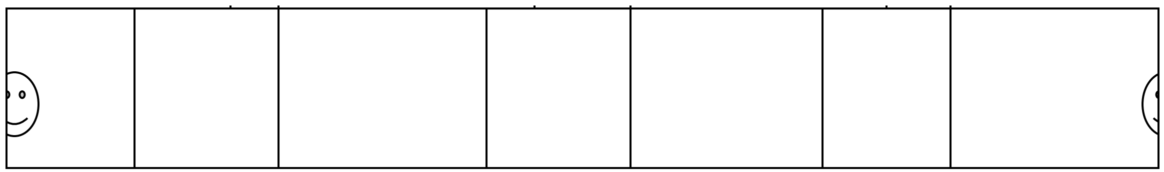
- Reproject each image onto a cylinder
- Blend
- Output the resulting mosaic

Cylindrical image stitching

What if you don’t know the camera rotation?

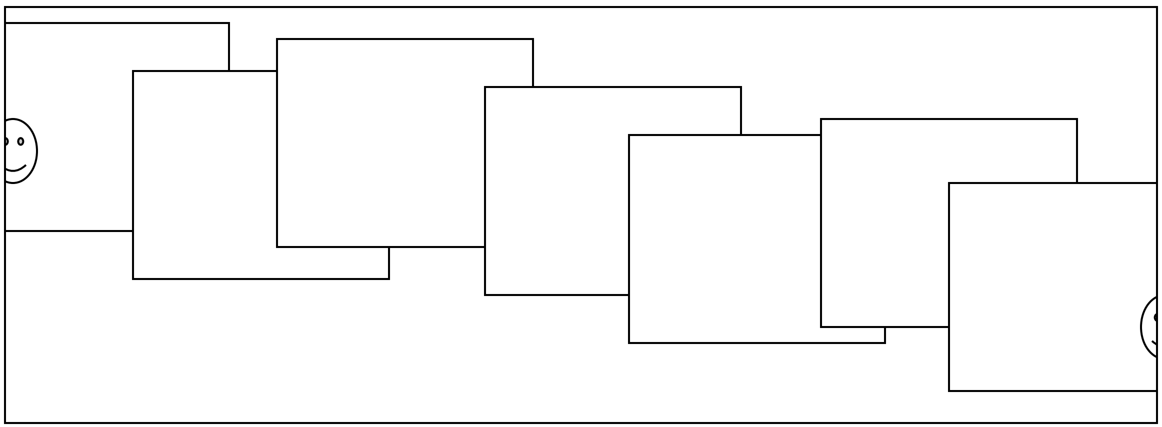
- Solve for the camera rotations
  - Note that a rotation of the camera is a **translation** of the cylinder!

Assembling the panorama



Stitch pairs together, blend, then crop

Problem: Drift



Vertical Error accumulation

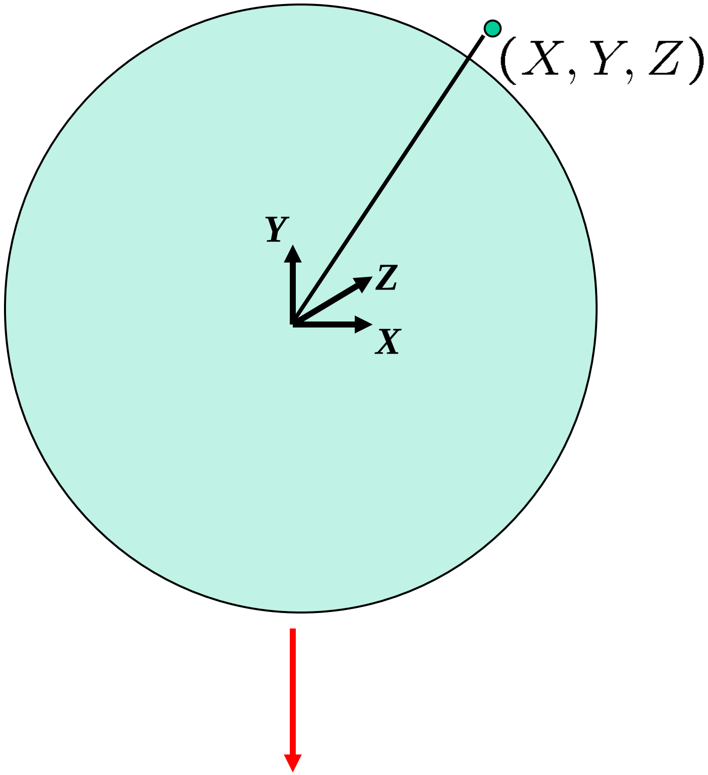
- small (vertical) errors accumulate over time
- apply correction so that sum = 0 (for 360° pan.)

Horizontal Error accumulation

- can reuse first/last image to find the right panorama radius

Full-view (360°) panoramas

Spherical projection



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

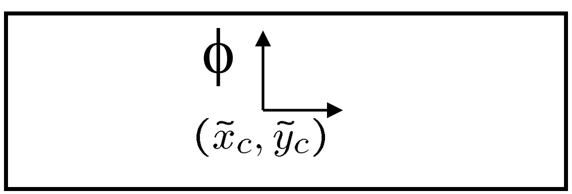
$$(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2 + Y^2 + Z^2}}(X, Y, Z)$$

- Convert to spherical coordinates

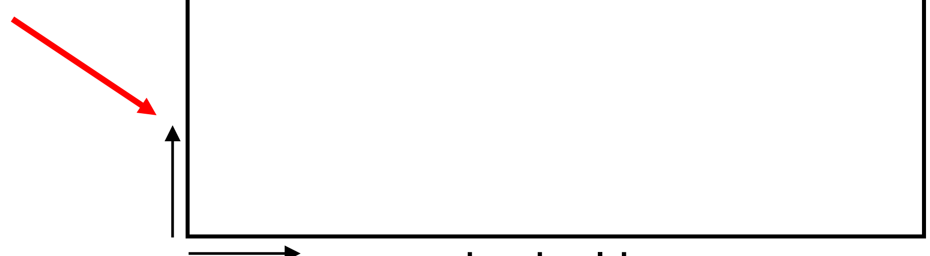
$$(\sin\theta\cos\phi, \sin\phi, \cos\theta\cos\phi) = (\hat{x}, \hat{y}, \hat{z})$$

- Convert to spherical image coordinates

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

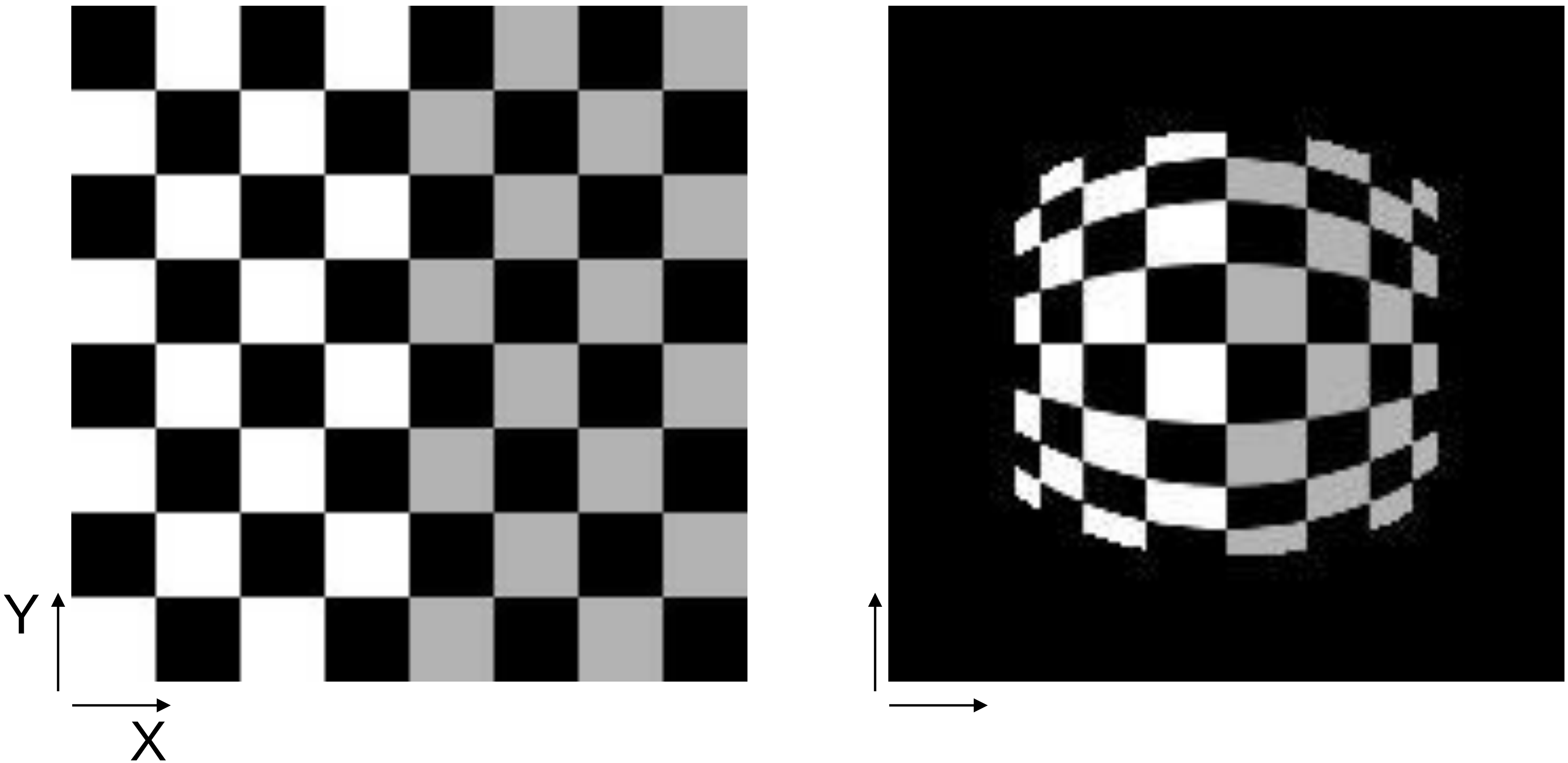


unwrapped sphere

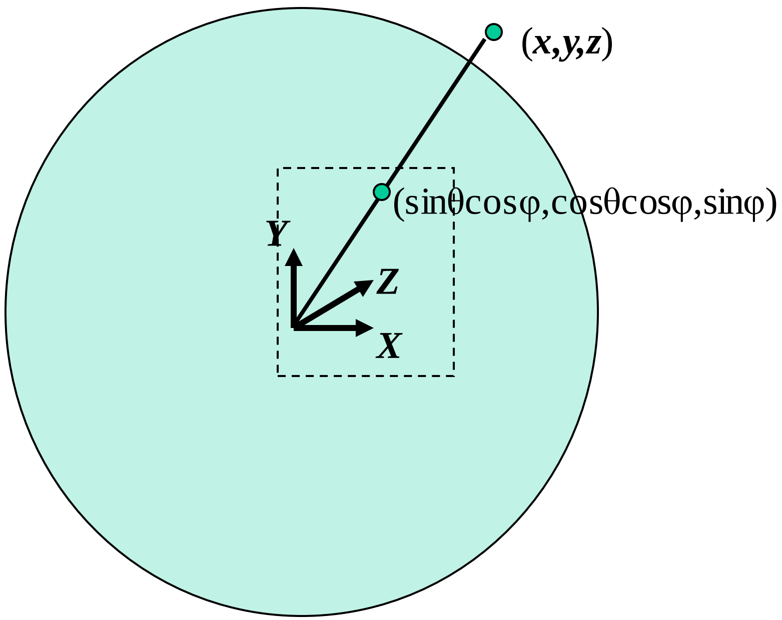


spherical image

Spherical Projection



Inverse Spherical projection



$$\theta = (x_{sph} - x_c)/f$$

$$\varphi = (y_{sph} - y_c)/f$$

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

$$\hat{y} = \sin\varphi$$

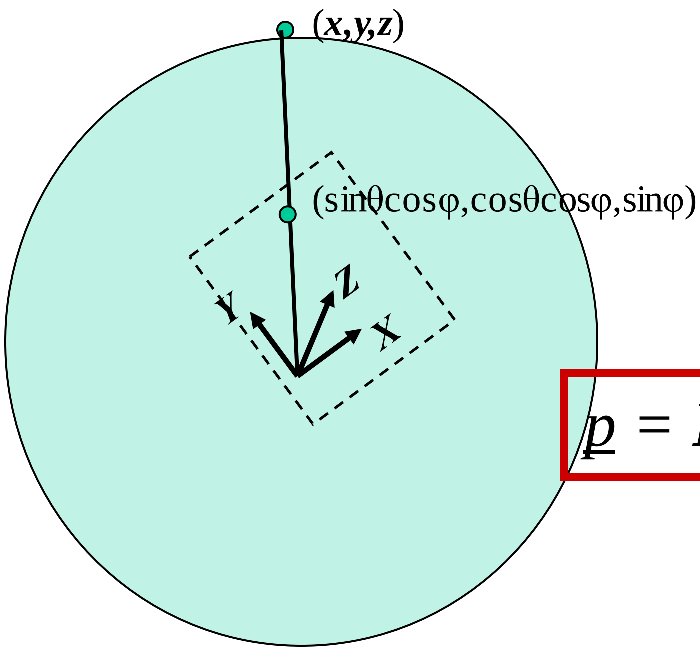
$$\hat{z} = \cos\theta \cos\varphi$$

$$x = f\hat{x}/\hat{z} + x_c$$

$$y = f\hat{y}/\hat{z} + y_c$$

3D rotation

Rotate image before placing on unrolled sphere



$$\boxed{\underline{p} = \boldsymbol{R}\, p}$$

$$\begin{aligned}
\theta &= (x_{sph} - x_c)/f \\
\varphi &= (y_{sph} - y_c)/f \\
\hat{x} &= \sin\theta \cos\varphi \\
\hat{y} &= \sin\varphi \\
\hat{z} &= \cos\theta \cos\varphi \\
x &= f\underline{\hat{x}}/\underline{\hat{z}} + x_c \\
y &= f\underline{\hat{y}}/\underline{\hat{z}} + y_c
\end{aligned}$$

Full-view Panorama

Other projections are possible

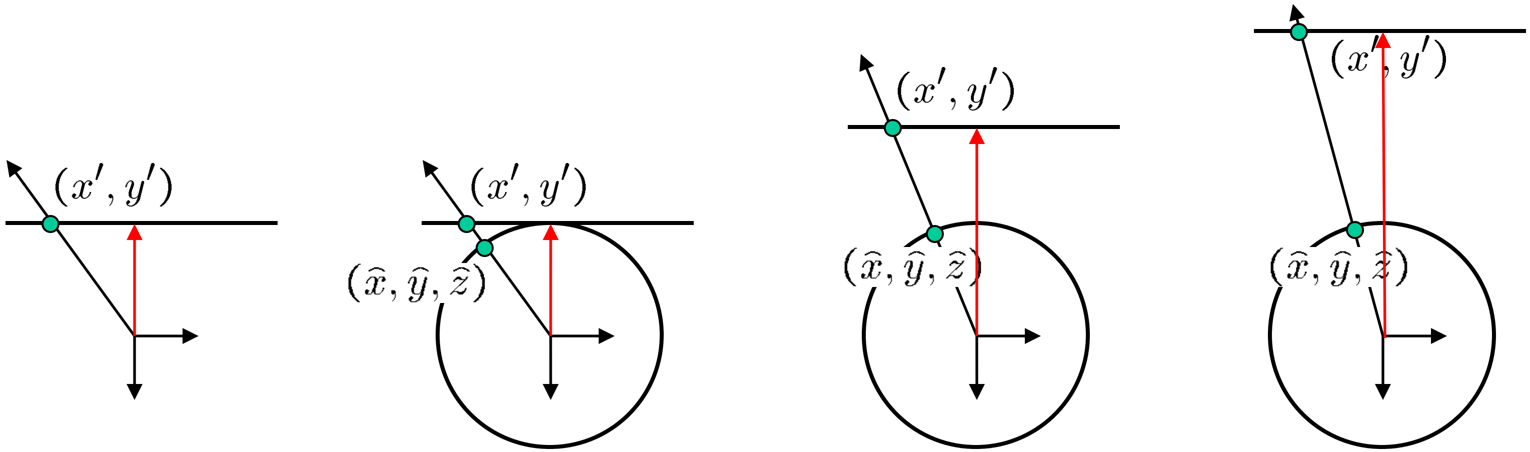
You can stitch on the plane and then warp the resulting panorama

- What's the limitation here?

Or, you can use these as stitching surfaces

- But there is a catch…

Cylindrical reprojection



top-down view

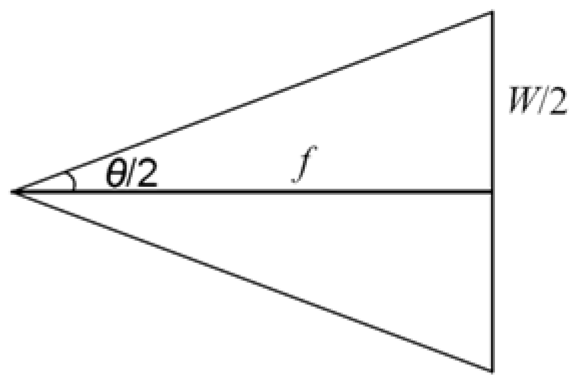
Focal length – the dirty secret…

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

What’s your focal length, buddy?

Focal length is (highly!) camera dependant

- Can get a rough estimate by measuring FOV:

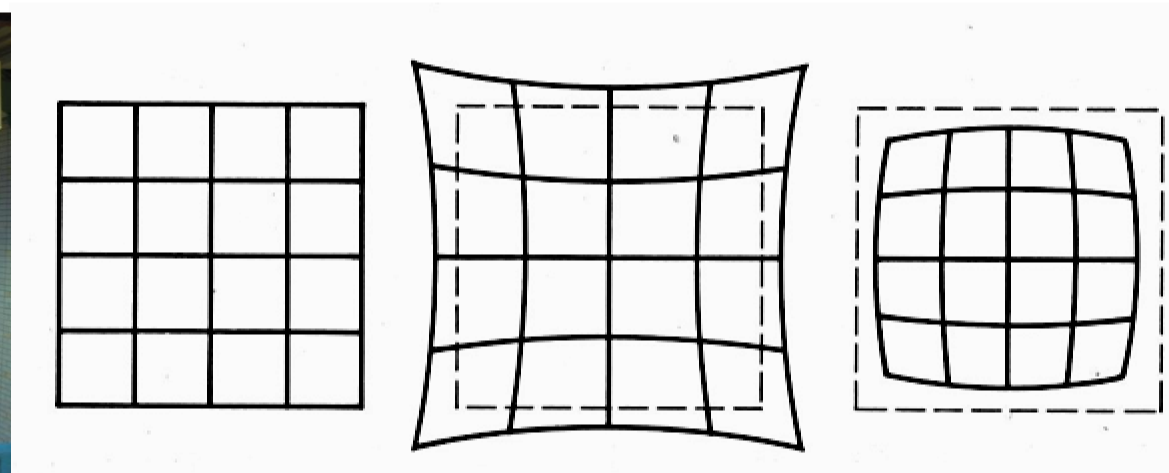


- Can use the EXIF data tag (might not give the right thing)
- Can use several images together and try to find f that would make them match
- Can use a known 3D object and its projection to solve for f
- Etc.

There are other camera parameters too:

- Optical center, non-square pixels, lens distortion, etc.

Distortion



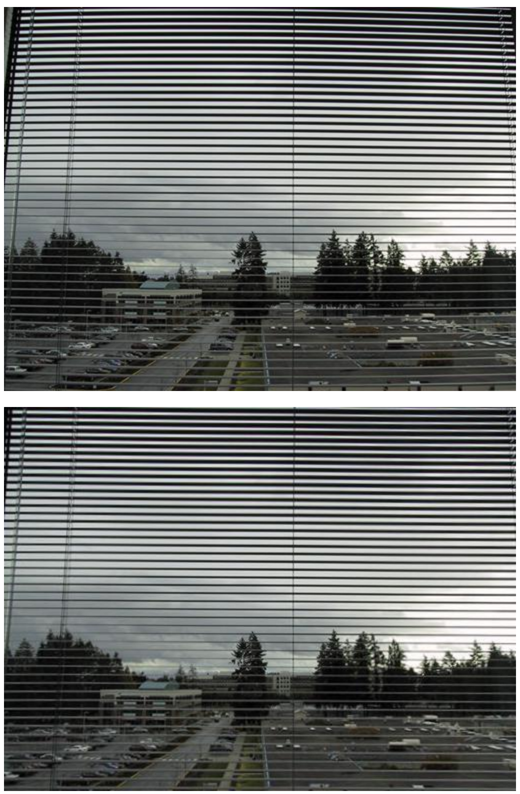
No distortion Pin cushion Barrel

Radial distortion of the image

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

Radial distortion

Correct for “bending” in wide field of view lenses



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

Use this instead of normal projection

Polar Projection

Extreme “bending” in ultra-wide fields of view

$$\hat{r}^2 = \hat{x}^2 + \hat{y}^2$$

$$(\cos\theta \sin\phi, \sin\theta \sin\phi, \cos\phi) = s\,(x, y, z)$$

uations become

$$\begin{aligned} x' &= s\phi\cos\theta = s\frac{x}{r}\tan^{-1}\frac{r}{z}, \\ y' &= s\phi\sin\theta = s\frac{y}{r}\tan^{-1}\frac{r}{z}, \end{aligned}$$

Camera calibration

Determine camera parameters from *known* 3D points or calibration object(s)

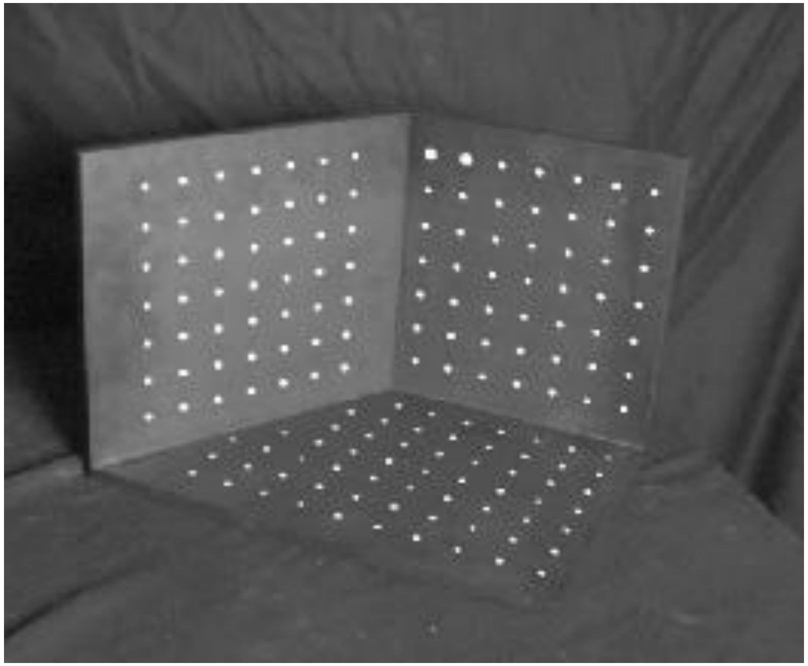
1. *internal* or *intrinsic* parameters such as focal length, optical center, aspect ratio:
   what kind of camera?
2. *external* or *extrinsic* (pose) parameters:
   where is the camera in the world coordinates?
   - World coordinates make sense for multiple cameras / multiple images

How can we do this?

Approach 1: solve for projection matrix

Place a known object in the scene

- identify correspondence between image and scene
- compute mapping from scene to image



$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Direct linear calibration

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

Solve for Projection Matrix Π using least-squares (just like in homework)

Advantages:

- All specifics of the camera summarized in one matrix
- Can predict where any world point will map to in the image

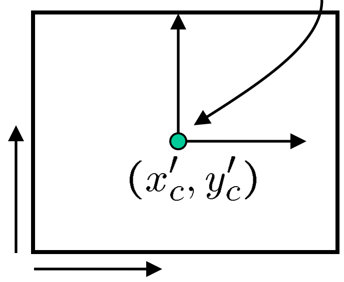
Disadvantages:

- Doesn't tell us about particular parameters
- Mixes up internal and external parameters
  - pose specific: move the camera and everything breaks

Approach 2: solve for parameters

A camera is described by several parameters

- Translation T of the optical center from the origin of world coords
- Rotation R of the image plane
- focal length f, principle point (x'_c, y'_c), pixel size (s_x, s_y)
- blue parameters are called "extrinsics," red are "intrinsics"



Projection equation

$$\mathbf{x} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi X}$$

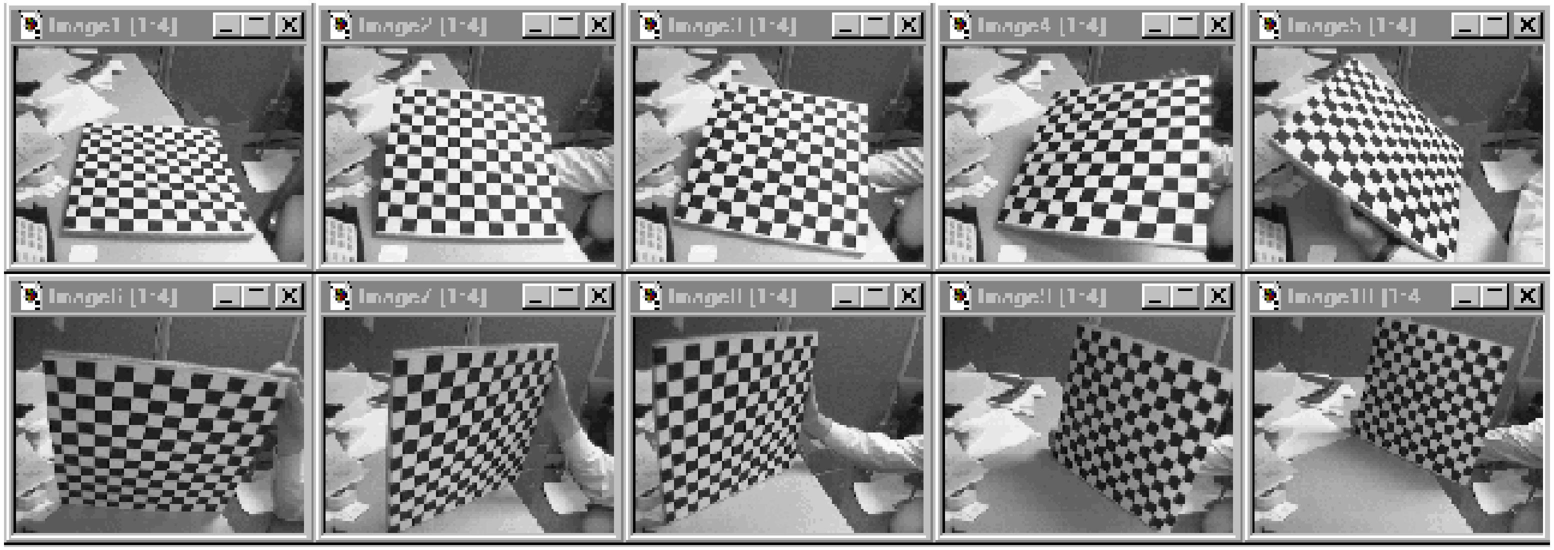
- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

identity matrix

$$\mathbf{\Pi} = \underbrace{\begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{rotation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{translation}}$$

- Solve using non-linear optimization

Multi-plane calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

- Only requires a plane
- Don’t have to know positions/orientations
- Good code available online!
  - Intel’s OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Zhengyou Zhang’s web site: http://research.microsoft.com/~zhang/Calib/