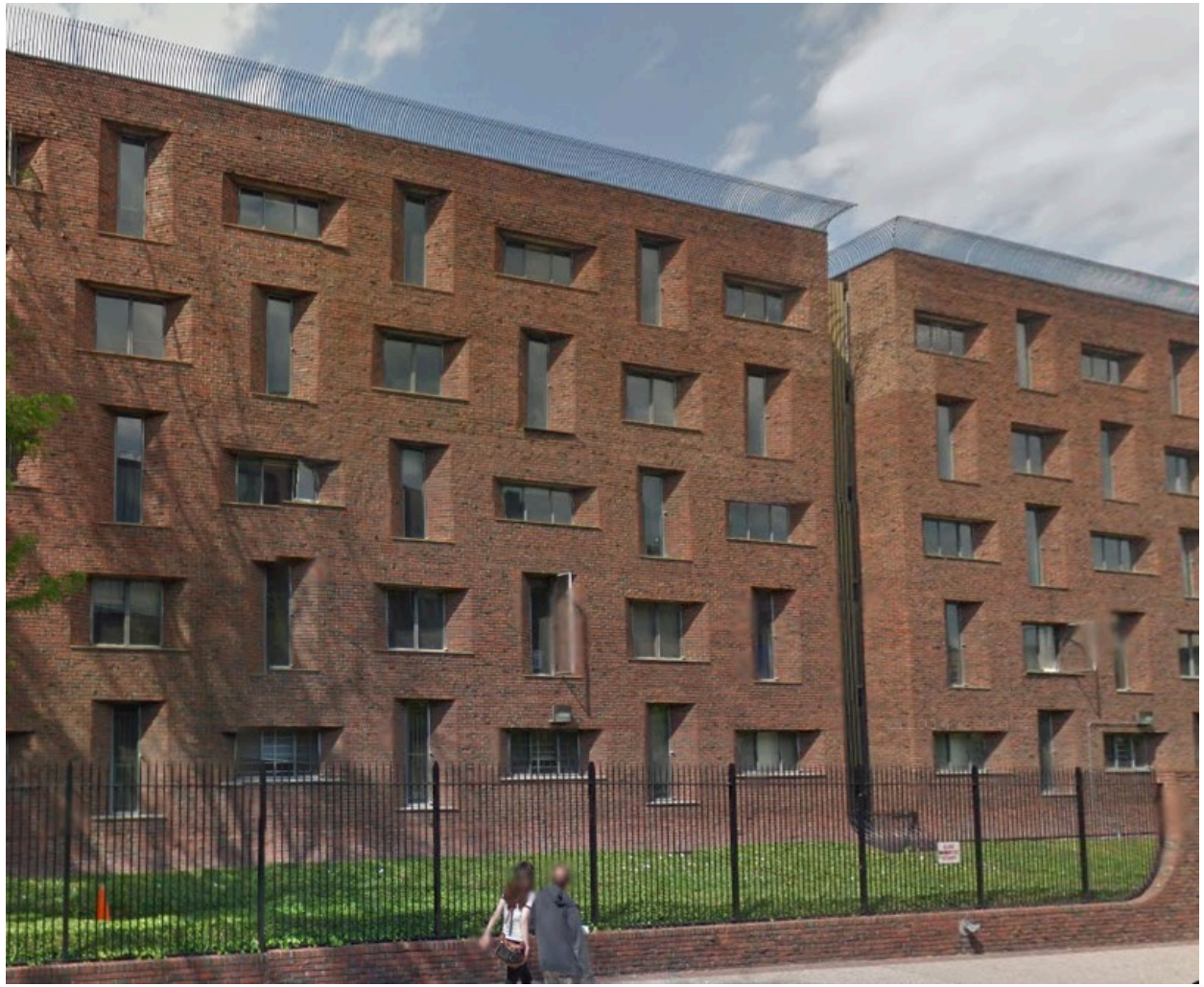
CIS 5800

Machine Perception

Instructor: Lingjie Liu

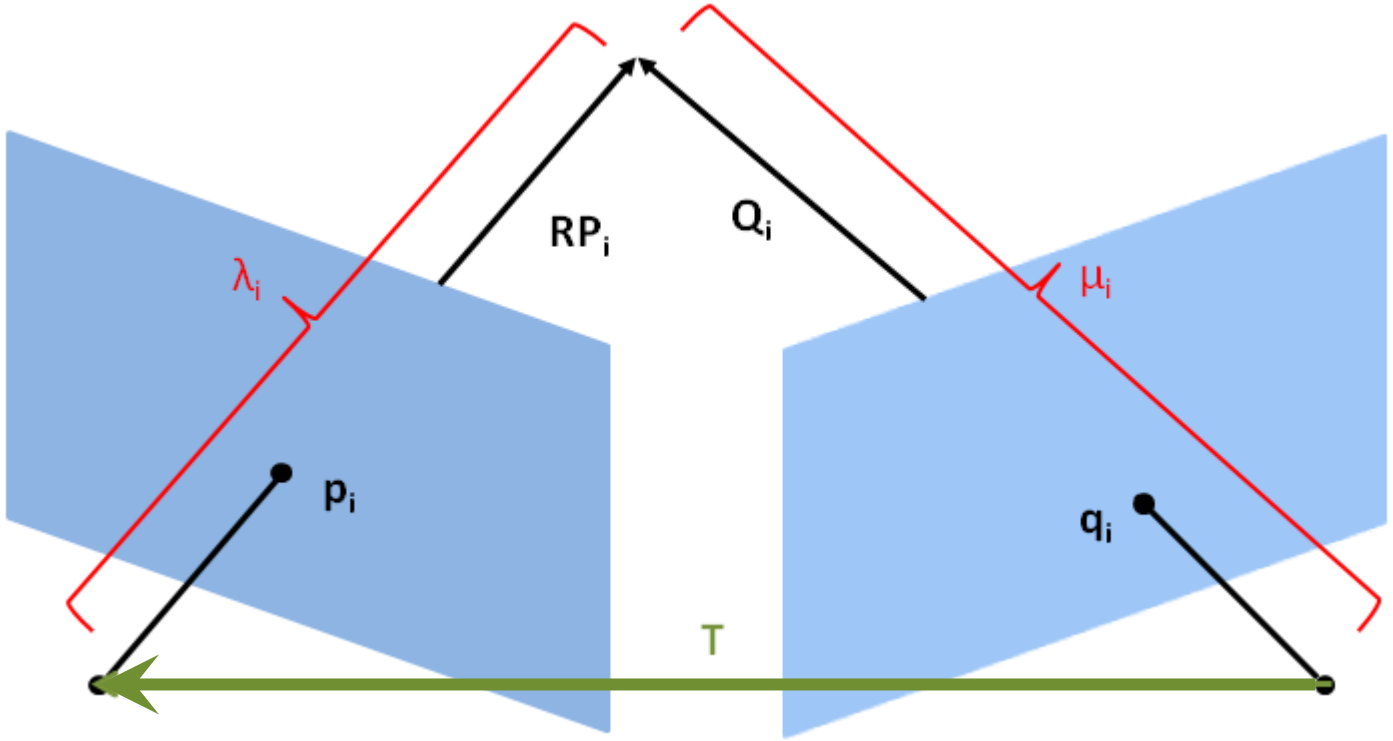
Lec 15: March 31, 2025

Recap: What is the relationship between two views of the same plane? (facade)



Answer: As we saw early in the course, homographies!

Recap: Two Calibrated Views of the Same 3D Scene



$$R(\lambda p) + T = \mu q$$

Given 2D correspondences (p,q)

Find motion R, T and depths λ, μ.

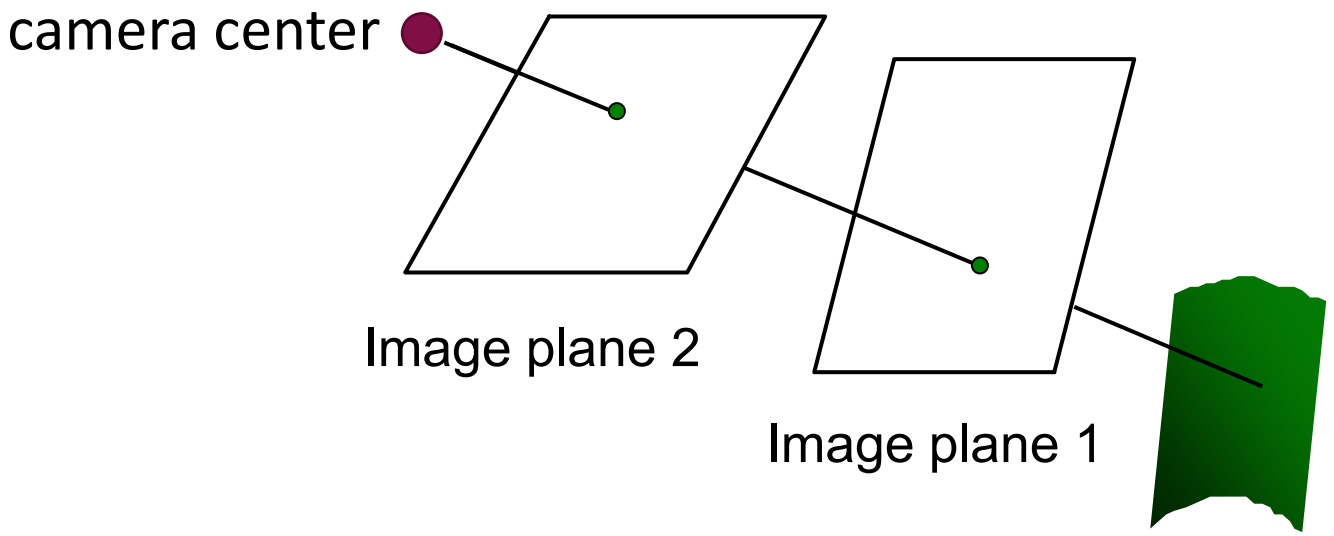
Recap: In 2-view SfM, what if there is no translation?

- Epipolar constraint becomes 0=0! So, cannot do all the SfM stuff above!
- Go back to $\lambda q = \mu R p + T(=0) = \mu R p$
  - In other words: $q \sim Rp$
  - Going back to pixel coordinates from calibrated:
    - $K^{-1} q_{px} \sim R K^{-1} p_{px}$
    - $q_{px} \sim K R K^{-1} p_{px}$
- KRK^{-1} is invertible (determinant =/= 0)
- So this is a homography $H = KRK^{-1}$!

Two views from the same camera center and different camera orientations are related by a homography even if the world is not planar!

Recap: No-Translation Image = Image of a Plane!

Two views from the same camera center and different camera orientations are related by a homography even if the world is not planar!



The image formed of the world on plane 2, may also be thought of as: the image of image plane 1. i.e. image of a plane i.e. homography!

Recap: Checking for no translation during SfM

- If you set up the n point correspondences as $A_{2n\times 9}\boldsymbol{h}_{9\times 1} = 0$ (as we have done before when solving for homography)
- Then, if $\text{rank}(A)$ is (approximately) 8, then you can (approximately) compute H. This means that, actually, there was no (or insignificant) translation, so you can't do SfM!

Q: Is this the only case when H is computable?
A: No, also when imaging a plane, of course.

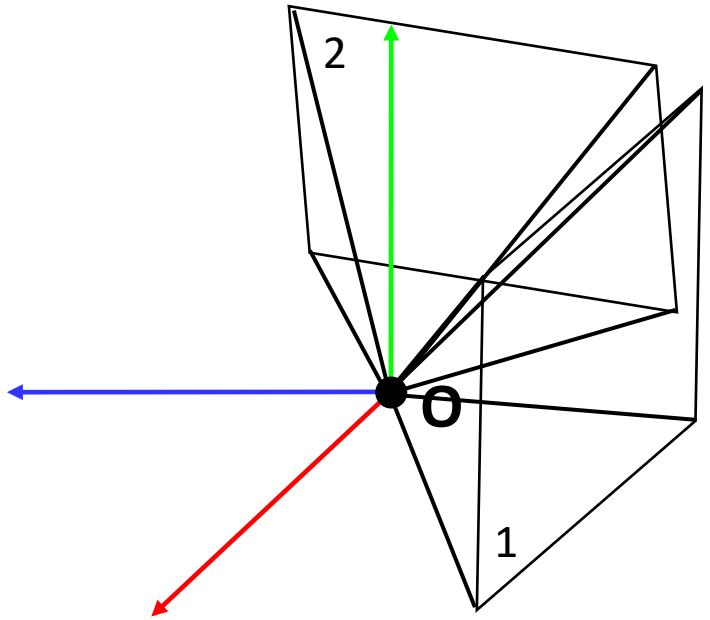
This check is a common component in SfM systems.

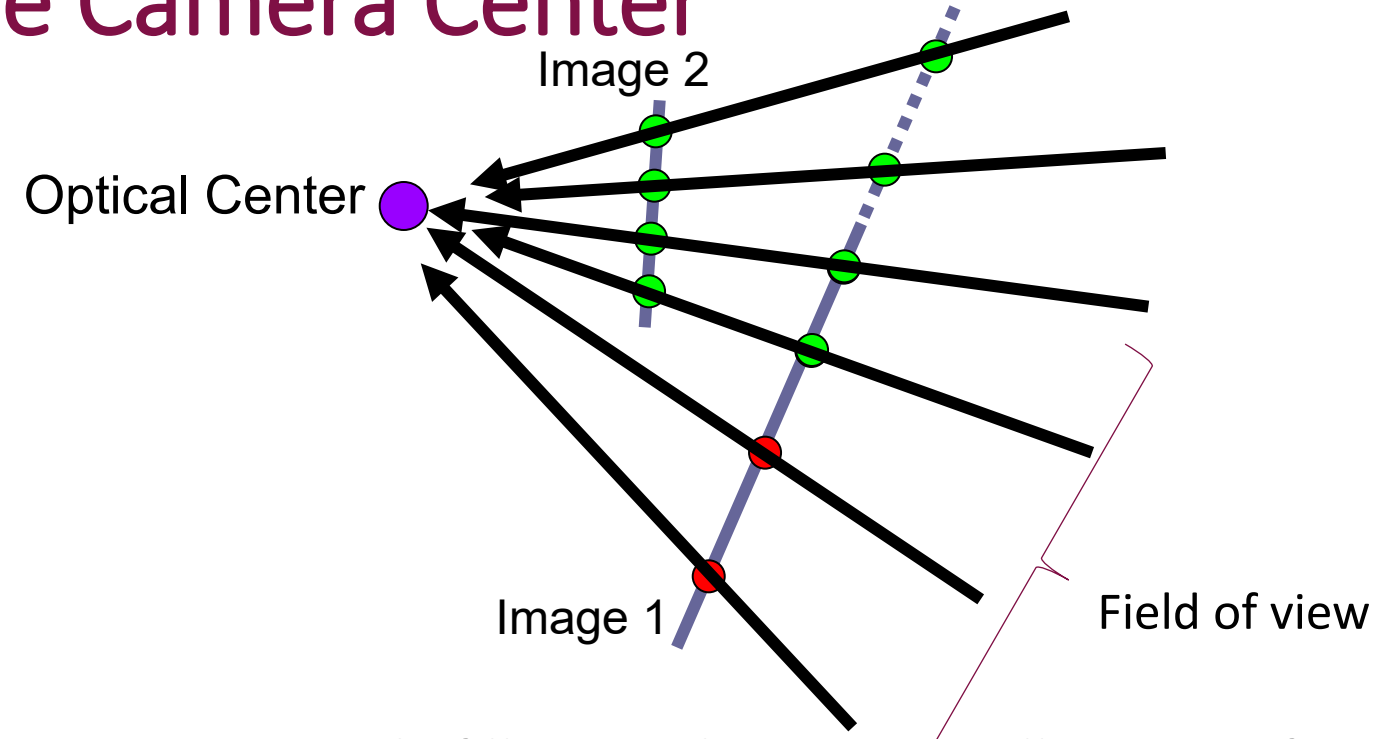
Note: While you can't do *SfM* with no translation of the camera, you can actually do other cool things, like building 360° image "panoramas"!
Coming up next!

Image Stitching / Mosaicing From Rotated Views

Based on slides by Richard Szeliski

2 Views From The Same Camera Center





- Camera field of view (FOV) is finite, so view 1 can only fill in pixels into a small region of image plane 1
- View 2 could see things outside the FOV of view 1. Could we add those pixels into view 1 to extend the image?

Yes. Project both images onto the same image plane e.g. image 1!

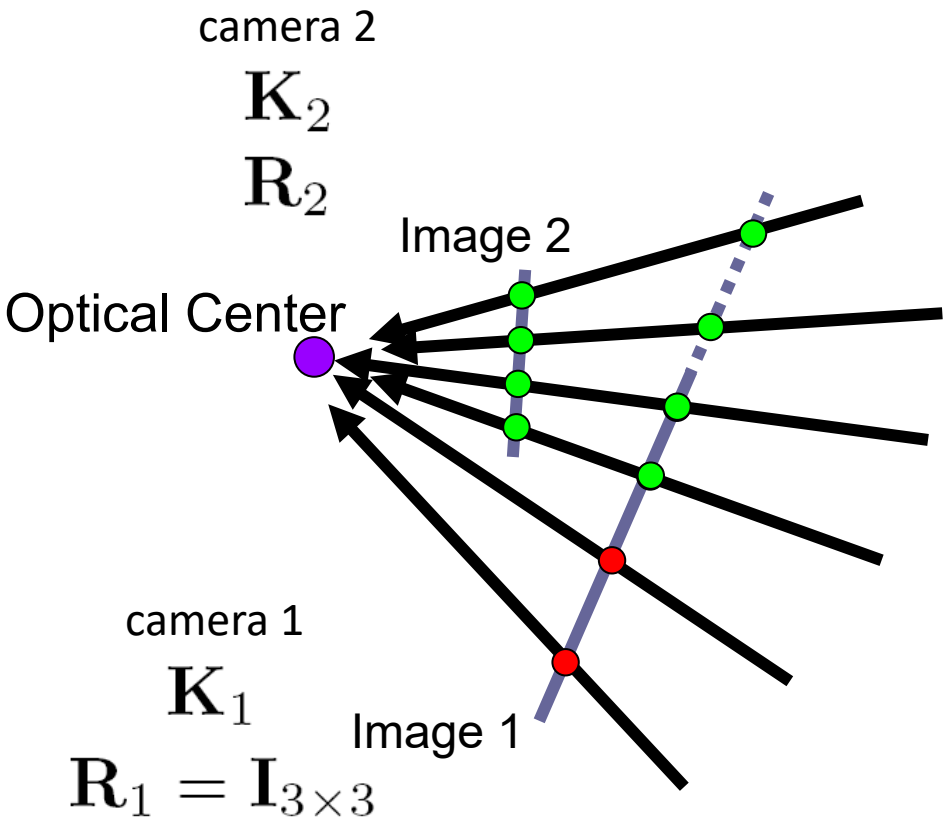
Your human vision system does something like this when your eyeball moves.

Image Stitching Results Example

https://pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/

What is the transformation between these views?

For every pixel in image 2, we need to find the right pixel location to place it inside image 1.



3x3 homography

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \sim \mathbf{K}_1 \mathbf{R}_1 \mathbf{R}_2^{\mathrm{T}} \mathbf{K}_2^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}$$

image coords (in image 1)

image coords (in image 2)

3D ray direction (in cam 2 coordinates)

3D ray direction in world coordinates

3D ray direction in camera 1 coordinates

Project through camera 1!

Homography even though not restricting to imaging a plane!

In practice, the homography is *estimated*

- We now know that the two images are related by a homography.

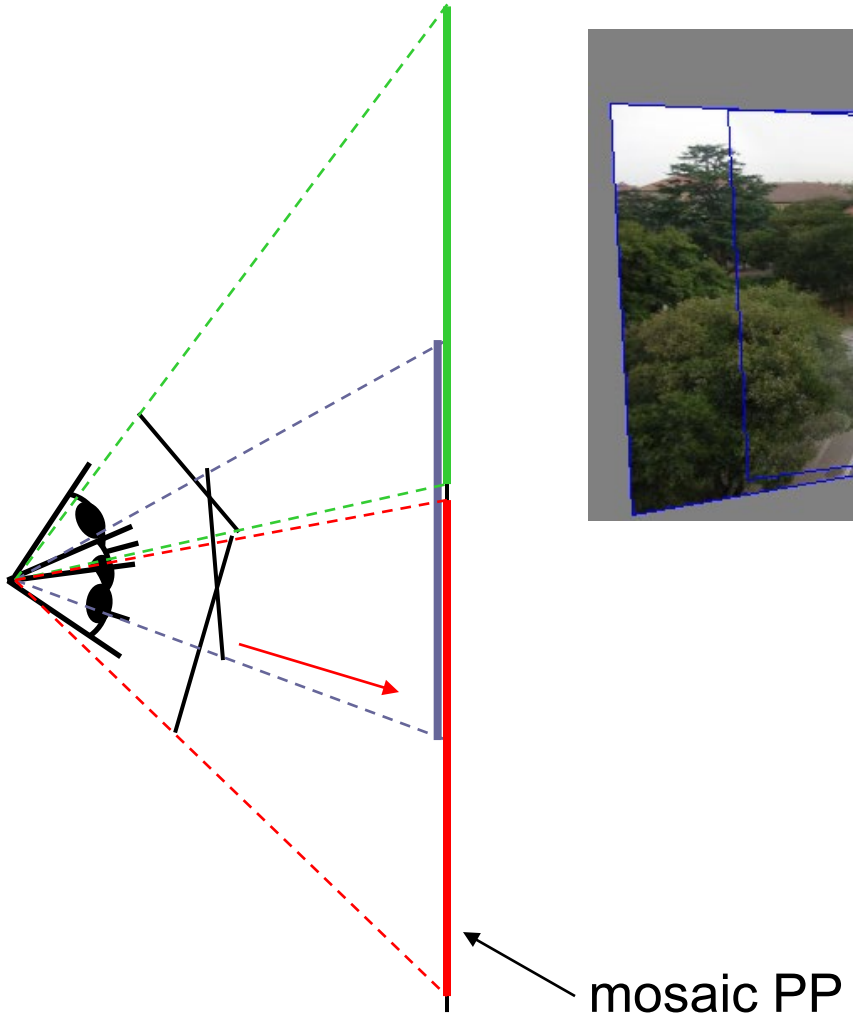
- In practice, we don't know R or even necessarily K_1 or K_2

- But fortunately, we know how to solve for homography H given point correspondences.
  - So we need the two images to overlap so that you can find correspondences and estimate the homography.

Creating a panorama from >2 images

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

Projecting images onto a common plane



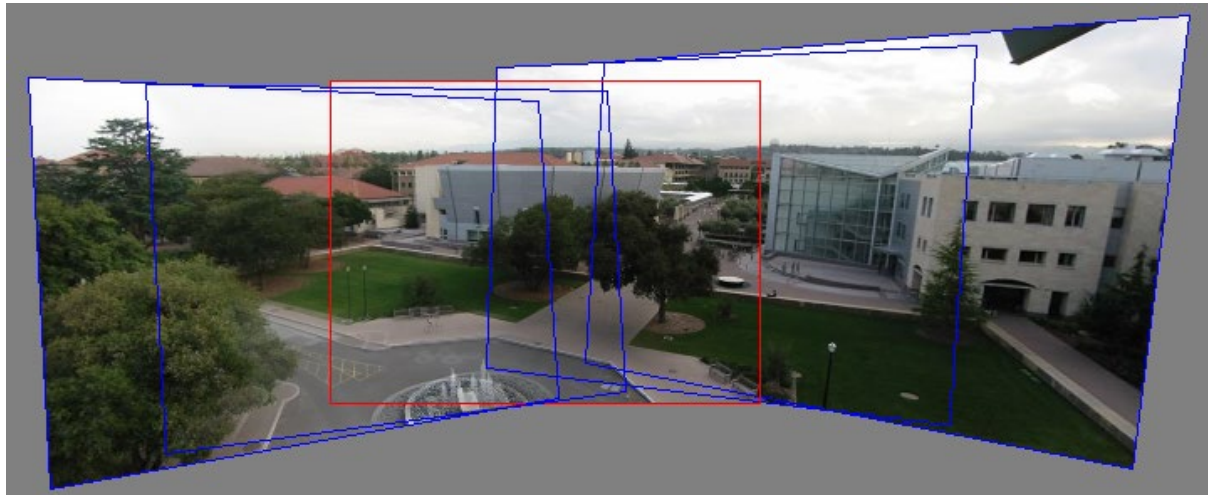
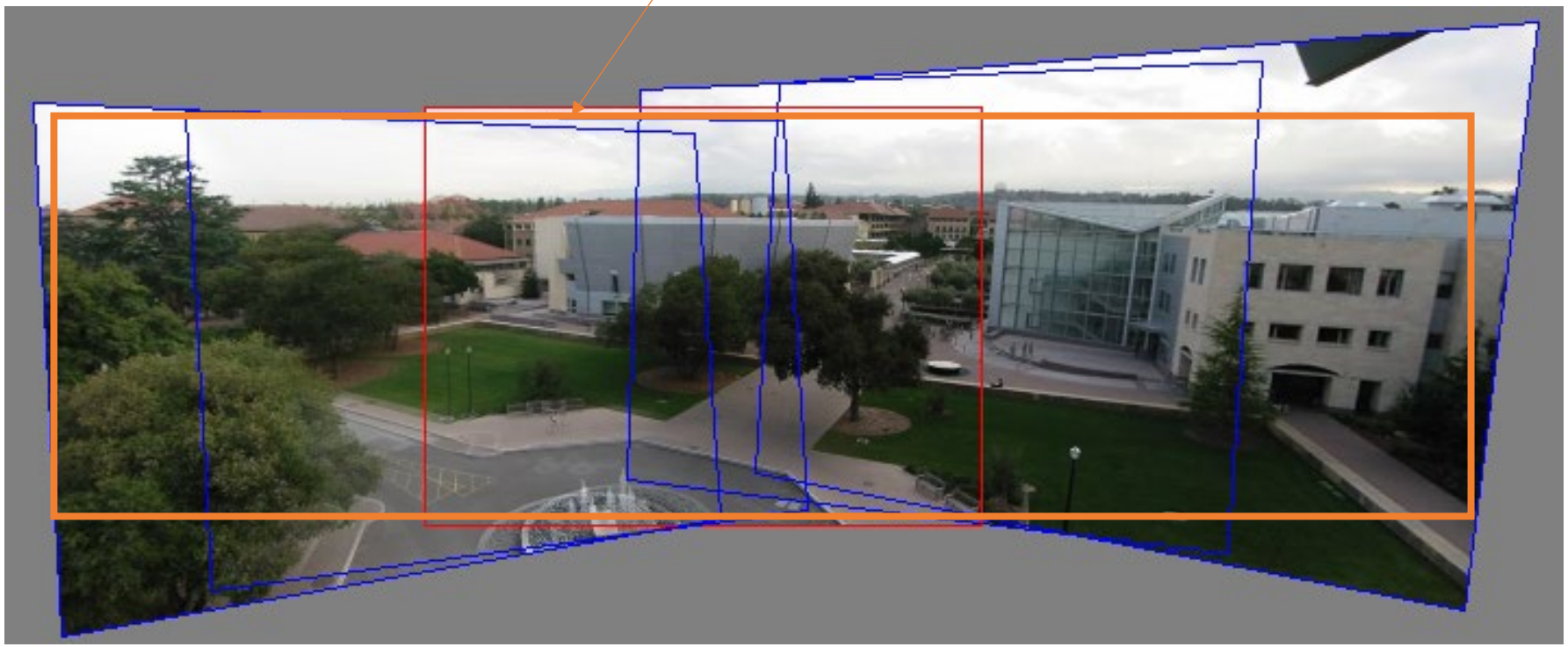


Image alignment

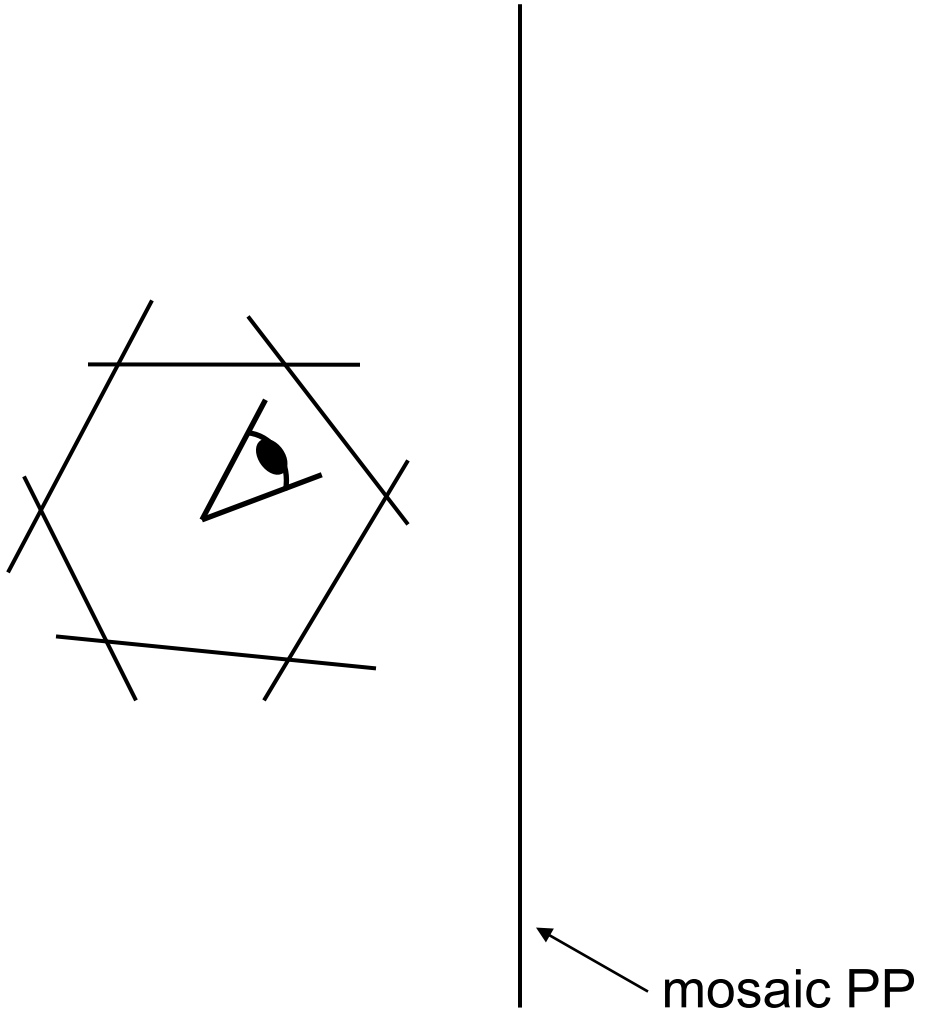
Often cropped like this to produce an image without empty regions



Can we use homography to create a 360° panorama?

Microsoft Lobby: http://www.acm.org/pubs/citations/proceedings/graph/258734/p251-szeliski

Can we use homography to create a 360° panorama?



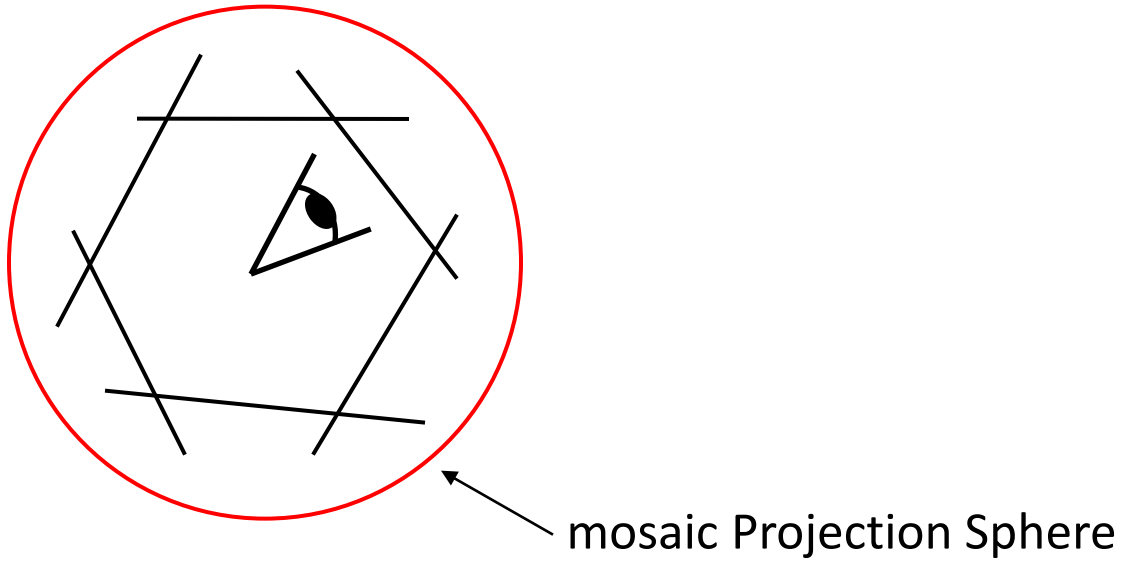
Projecting backwards-facing views onto the same image plane?

Breaks down.

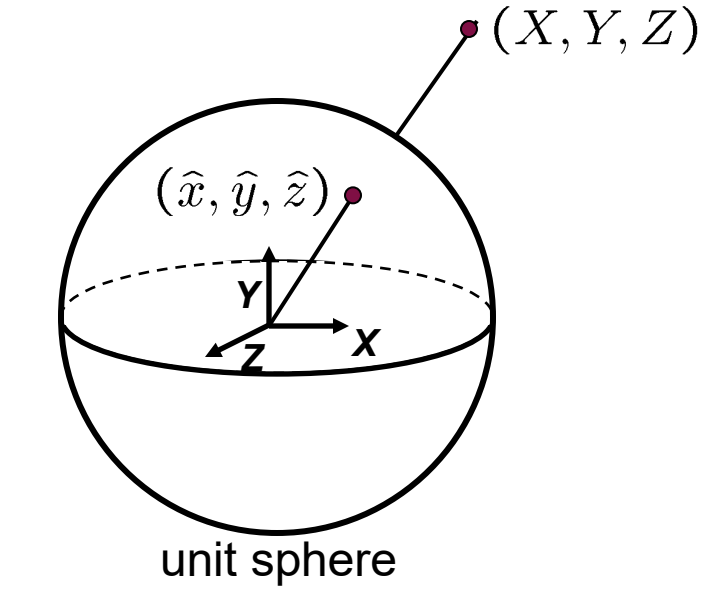
Panoramas

Instead of projecting onto a plane, you project onto a sphere!

i.e. for every 3D point, draw the line connecting it to the camera center, and find its intersection with a projection sphere --- this is the point's image!



Spherical projection (overview)



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

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

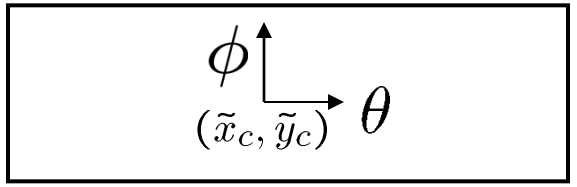
- Convert to spherical coordinates

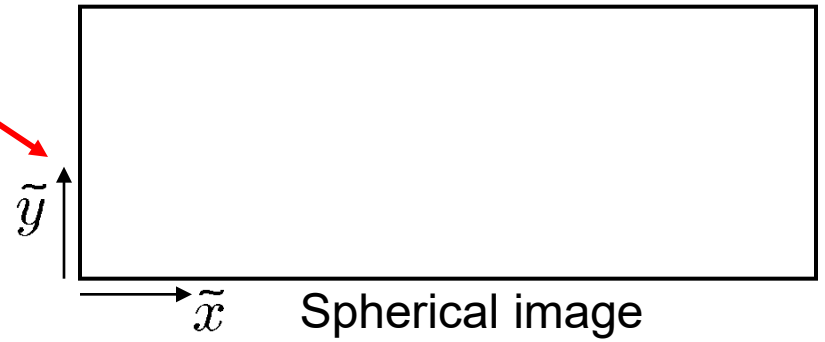
$$(sin\theta cos\phi, sin\phi, cos\theta cos\phi) = (\hat{x}, \hat{y}, \hat{z})$$

- Convert to spherical image coordinates

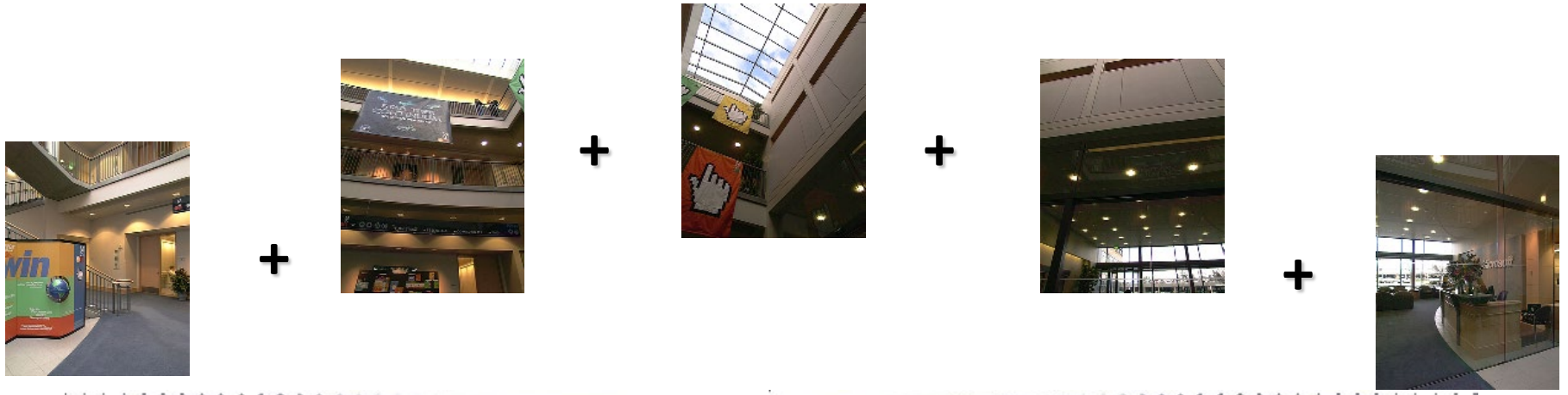
$$(\tilde{x}, \tilde{y}) = (s\theta, s\phi) + (\tilde{x}_c, \tilde{y}_c)$$

  - s defines size of the final image





Spherical Panorama Example



Note the distortions

Microsoft Lobby: http://www.acm.org/pubs/citations/proceedings/graph/258734/p251-szeliski

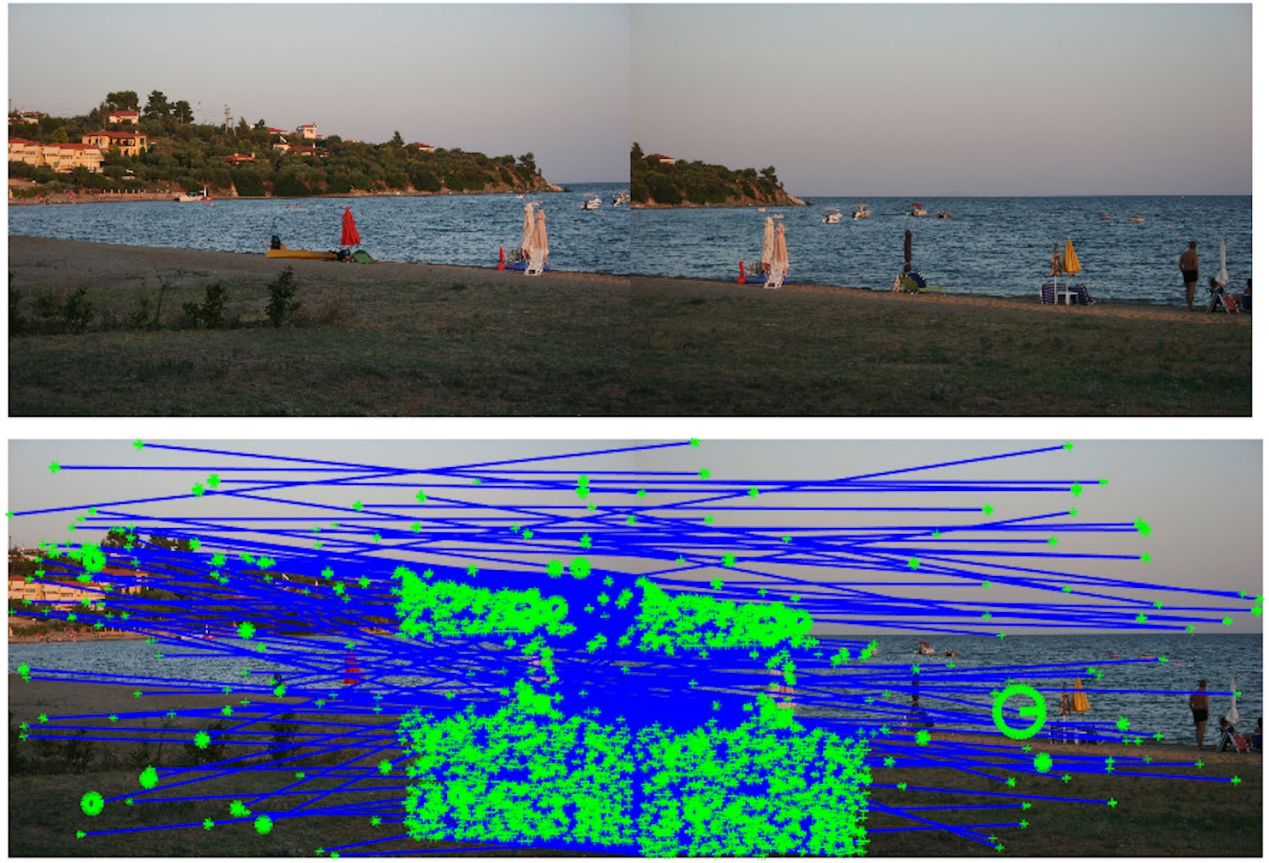
Hough and RANSAC

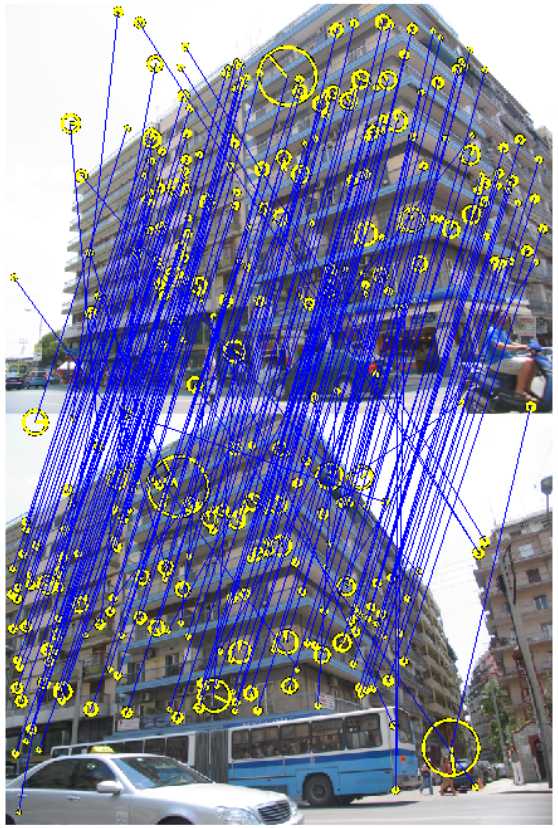
Simple subroutines for feature identification, grouping, and correspondence filtering


And now for something completely different

Correspondences are not clean!

Or we might have to group them !
(two planes=>two homographies)

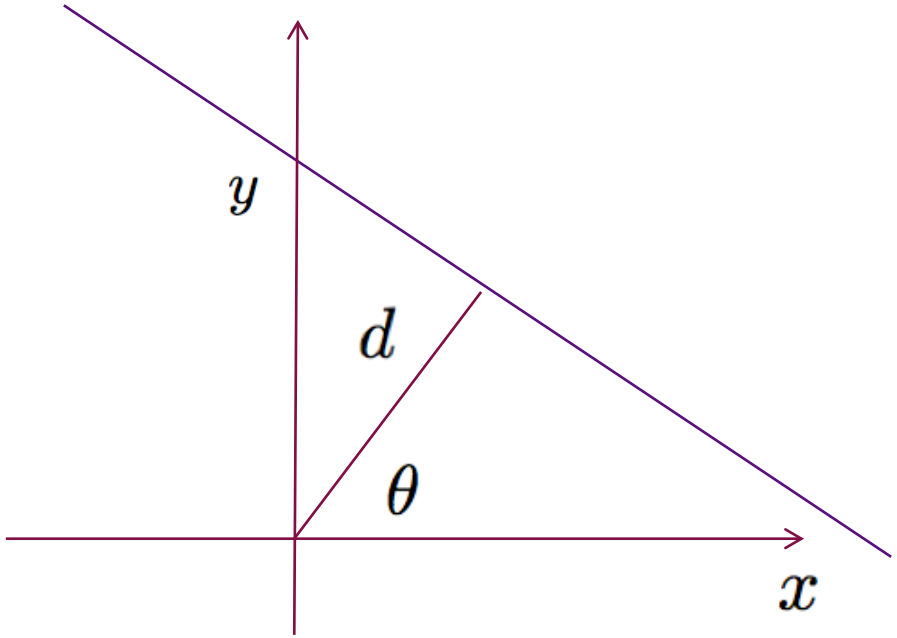




A Case Study: Fitting A Line To Data

Given data (x_i, y_i) belonging to a line

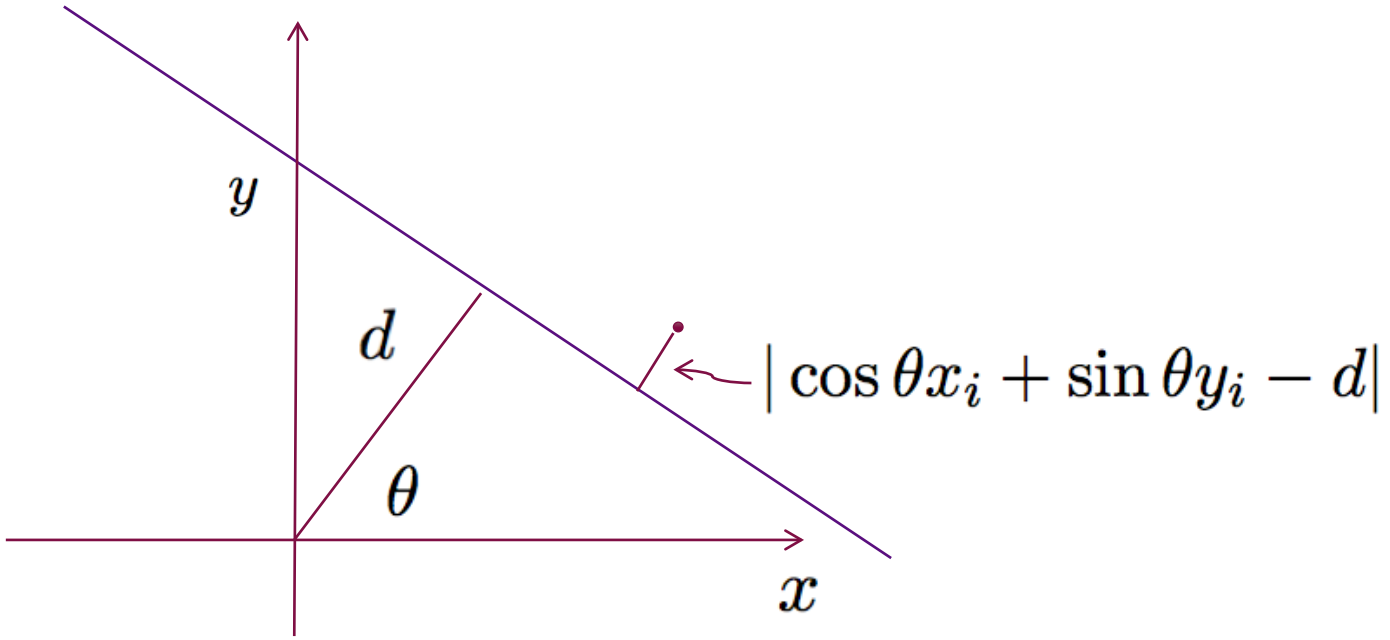
$$x\cos\theta + y\sin\theta = d$$



Finding the line by optimization

To solve for the line, we could minimize the mean squared error:

$$\underset{\theta \in [0,2\pi), d \geq 0}{\operatorname{argmin}} \sum_{i=1}^{N} (x_i \cos\theta + y_i \sin\theta - d)^2$$



This is solvable. Exactly how is not particularly important. (next 2 slides)

Solution sketch (not for testing)

$$\underset{\theta\in[0,2\pi),d\geq 0}{\operatorname{argmin}} \sum_{i=1}^{N}(x_i\cos\theta + y_i\sin\theta - d)^2$$

$$f(\theta,d) = \sum_{i=1}^{N}(x_i\cos\theta + y_i\sin\theta - d)^2$$

$$\frac{\partial f}{\partial d} = \sum_{i=1}^{N}(-2)(x_i\cos\theta + y_i\sin\theta - d) = 0$$

$$d = \frac{1}{N}\sum_{i}(x_i\cos\theta + y_i\sin\theta)$$

Solution sketch (not for testing)

We can eliminate d and obtain

$$\arg\min_{\theta\in[0,2\pi)} \sum_{i=1}^{N} ((x_i - \bar{x})\cos\theta + (y_i - \bar{y})\sin\theta)^2$$

which is equivalent to the minimization of the Rayleight quotient

$$\arg\min_{\theta\in[0,2\pi)} \begin{pmatrix}\cos\theta & \sin\theta\end{pmatrix} \underbrace{\sum_{i=1}^{N} \begin{pmatrix}(x_i - \bar{x})^2 & (x_i - \bar{x})(y_i - \bar{y}) \\ (x_i - \bar{x})(y_i - \bar{y}) & (y_i - \bar{y})^2\end{pmatrix}}_{C} \begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix}$$

Solution is the eigenvector to the minimal eigenvalue of C.

$P(\theta, d|x_i, y_i)$, kind of like a "vote" by (x_i, y_i) for (θ, d)

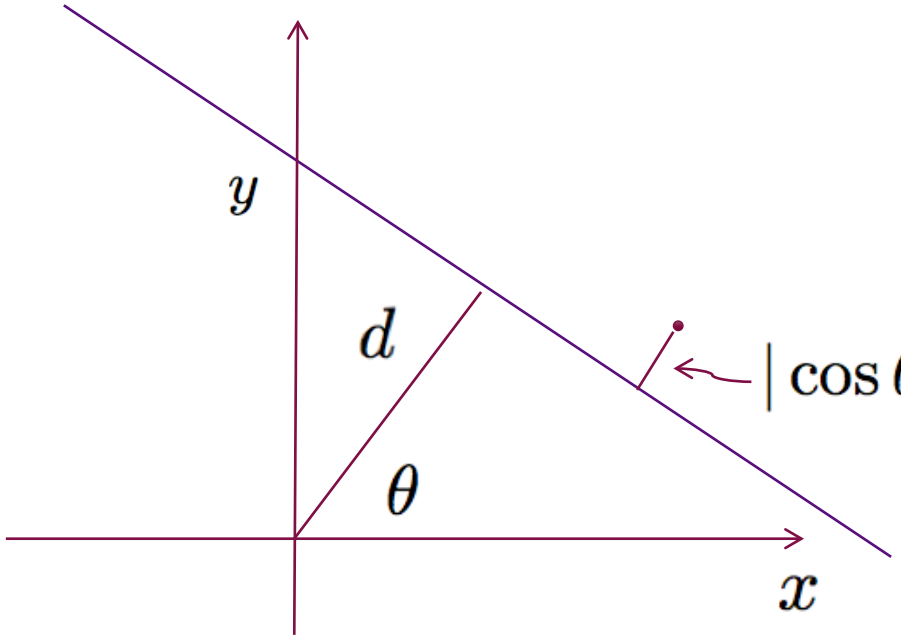
This corresponds to maximizing:

$$\sum_i e^{-\frac{1}{2\sigma_i^2}(\cos\theta x_i + \sin\theta y_i - d)^2}$$

with $\sigma_i = 1$ being the same for all points.

If you have machine learning or statistics background, what is this type of line fitting called?

Maximum likelihood!



$$|\cos\theta x_i + \sin\theta y_i - d|$$

Key idea: find the parameters that the most data points "vote" for.

Line Parameter Space

For a line in the form:

$$x\cos\theta + y\sin\theta = d$$

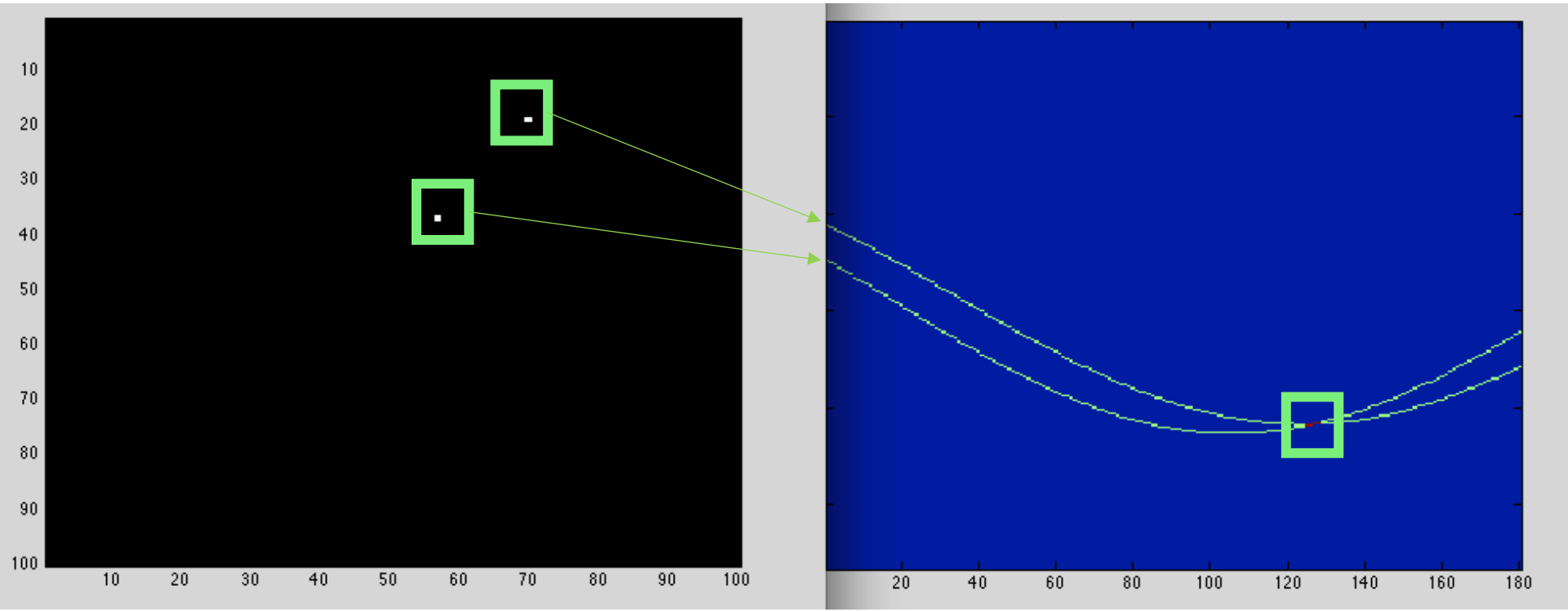
The parameter space of the line is in (θ, d). Each point in this parameter space corresponds to a line in 2D.

For a point (x_i, y_i) in x-y space, what does it correspond to in (θ, d)?

$$d = x_i cos\theta + y_i sin\theta$$

Line through two points (x_1, y_1) and (x_2, y_2) can be found as the intersection (θ, d) of the two curves:

$$\begin{aligned} d &= \cos\theta x_1 + \sin\theta y_1 \\ d &= \cos\theta x_2 + \sin\theta y_2 \end{aligned}$$



Hough Voting Procedure

For every possible line (d, θ) , count the number of points that "support" it.

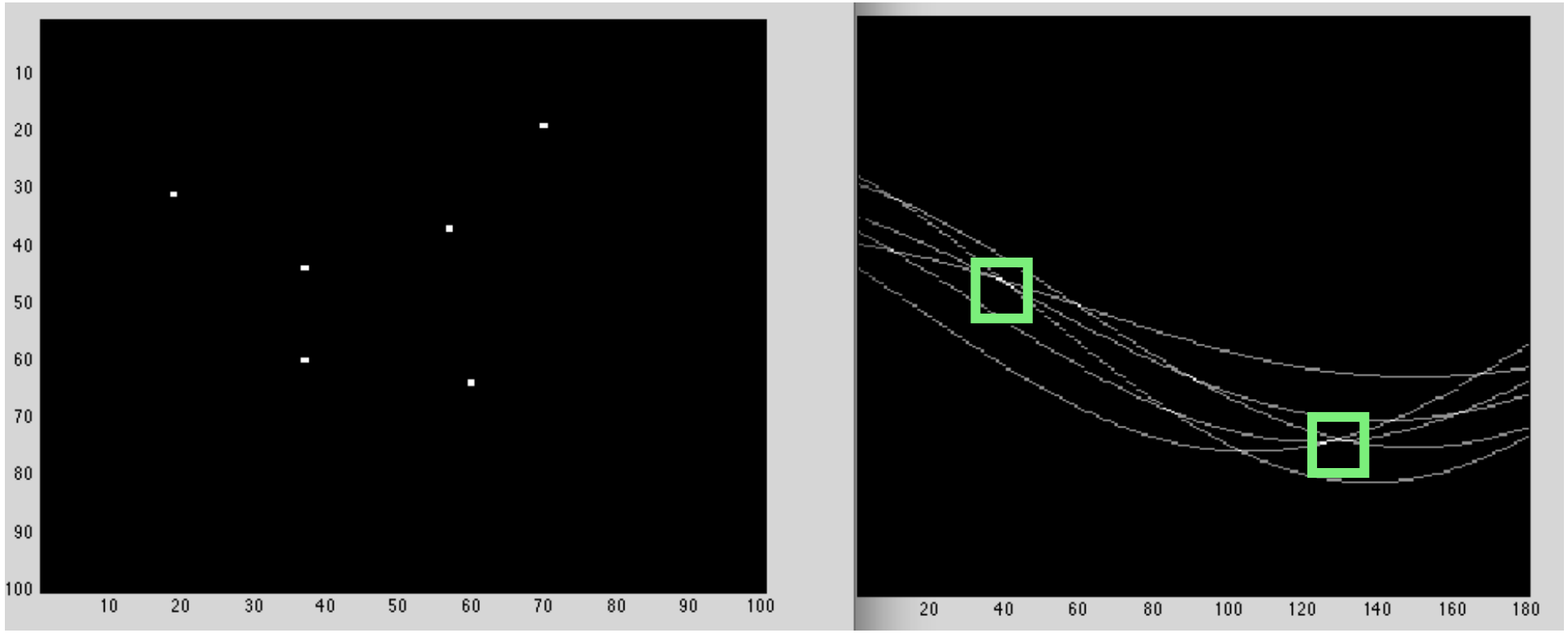
$$\sum_{\text{points } x,y} 1(x \cos\theta - y \sin\theta = d)$$

(can sometimes relax the equality sign to approximate equality, or makes vote "soft" etc.)

In the end, the (d, θ) with the maximum "votes" wins!

(in case of multiple lines, not just one winner, but all local maxima above some threshold win)

Vote Counting to Find Lines!



Algorithm v1, and a practical issue

Seems impractical

- For every possible line (d, θ), count the number of points that "support" it.

$$\sum_{\text{points } x,y} 1(|x\cos\theta + y\sin\theta - d| < \delta_{\text{threshold}})$$

- In the end, local maxima with many votes are declared lines.

Solution: discretizing the parameter space.

- For each of some enumerated discrete parameter combinations $\{(d_j, \theta_j)\}_{j=1}^{J}$, count supporting points among the data.

$$V[d_j, \theta_j] = \sum_{\text{points } x,y} 1(|x \cos \theta_j + y \sin \theta_j - d_j| < \delta_{\text{threshold}})$$

  - For example, $\Theta = \{0°, 15°, \ldots, 180°\} \times D = \{0, 1, 2, 3, 4, 5\} \Rightarrow 13 \times 6 = 78$ parameter combinations
- In the end, local maxima of $V[d, \theta]$ with many votes are declared lines.

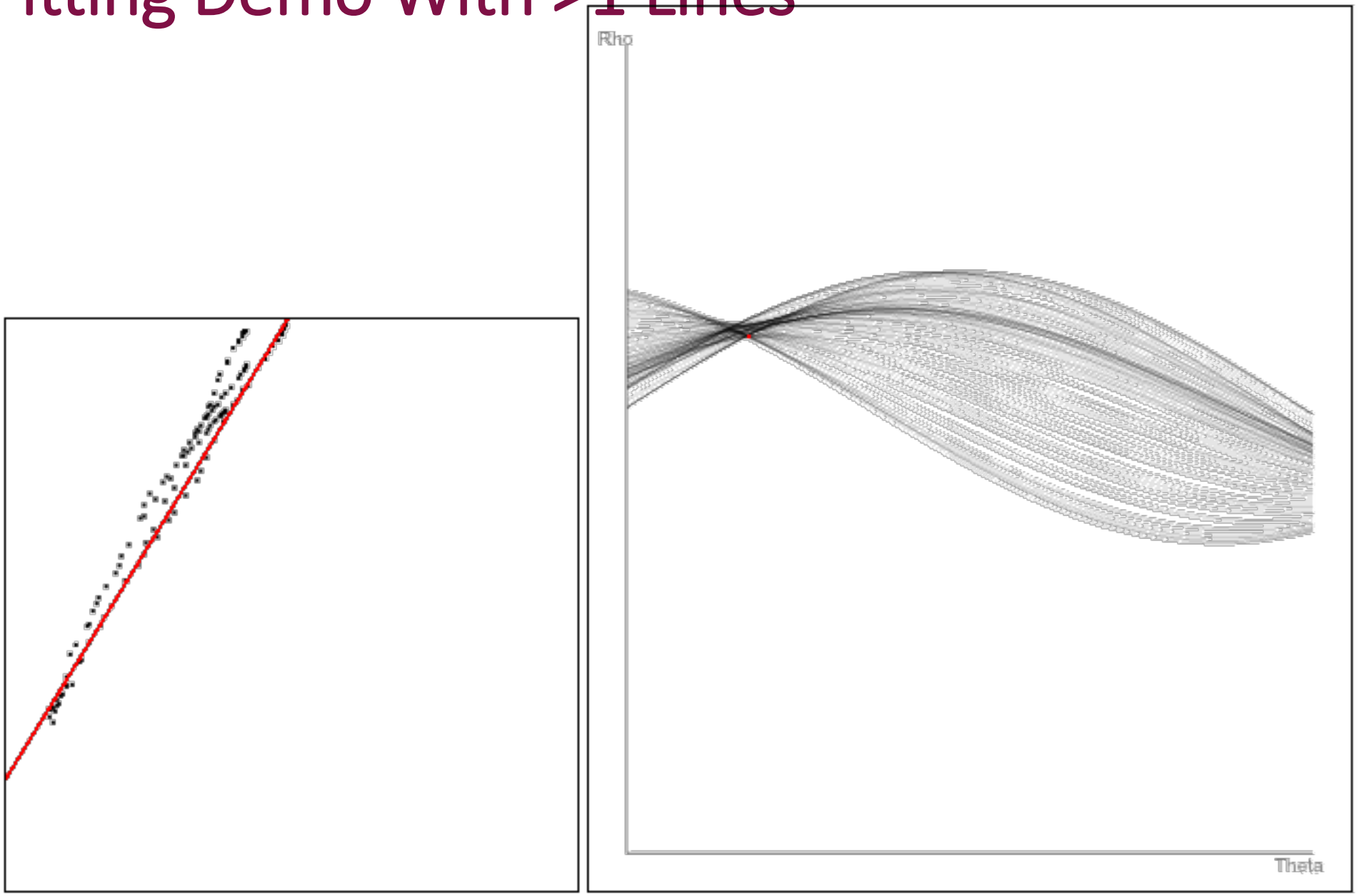
Notes:

Needs careful choices about how to discretize parameters into bins.

Each bin should correspond to an error you are okay with making, so can't be too coarse.

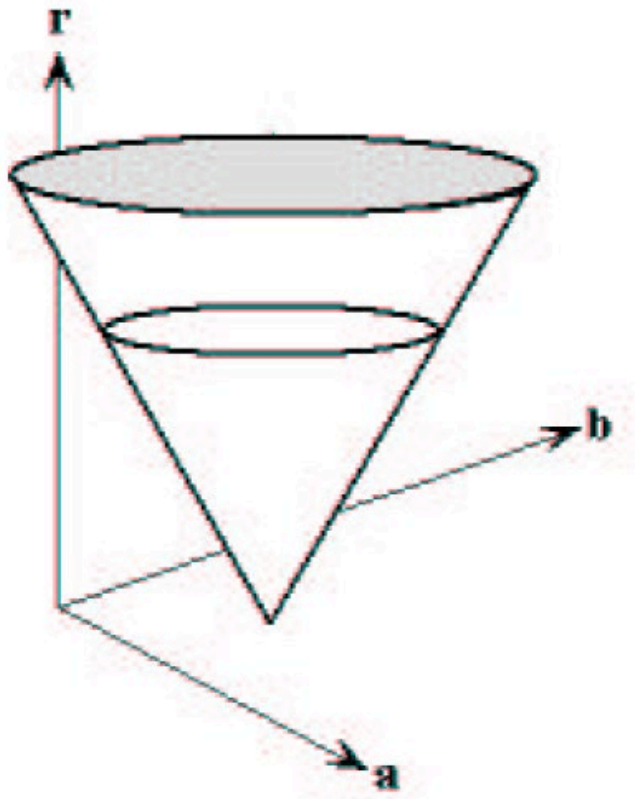
But too fine-grained => computationally intensive!

Line Fitting Demo With >1 Lines



https://www.aber.ac.uk/~dcswww/Dept/Teaching/CourseNotes/current/CS34110/hough.html

Hough Circle Detection (3 parameters)



If we fix the (x, y) in the circle equation

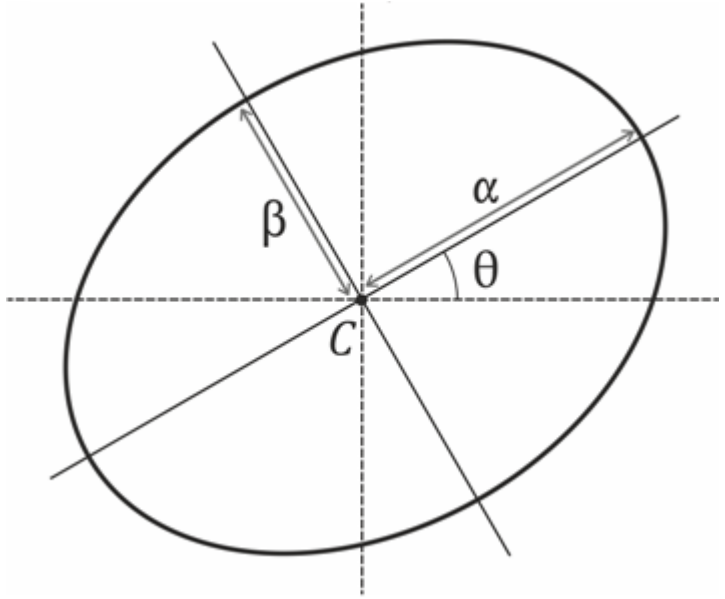
$$(x-a)^2 + (y-b)^2 = r$$

we obtain the equation of a cone in (a, b, r) Hough space. Observe that we used r and not r^2.

Hough Ellipse Detection (5 parameters)

Let us look at the ellipse equation

$$\begin{pmatrix} x - c_x & y - c_y \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} \frac{1}{\alpha^2} & 0 \\ 0 & \frac{1}{\beta^2} \end{pmatrix} \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x - c_x \\ y - c_y \end{pmatrix} = 1$$



Hough transform needs the computation of $V(c_x, c_y, \alpha, \beta, \theta)$ and search for maxima in 5-dimensional space. Hough becomes intractable beyond lines and circles.

Application: Automatically Grouping Vanishing Points

(Roughly, each pair of detected line segments votes for their point of intersection. Larger weights for longer segments, and non-collinear segments)

Disadvantages of Hough Transforms

Hough transforms are great and simple-to-implement approach for grouping k samples into n “groups” (e.g. lines, circles, VPs etc.) + outliers that don’t belong to any group.

- Note: Needs a bounded parameter space to allow a finite discrete set of hypotheses, and careful discretization.
- Main drawback: Poor scaling. Becomes intractable when # parameters(unknowns) for the target groupings is more than 3 or at most 4 in practice.
  - So, e.g. to fit homographies (8 parameters), completely intractable.

Alternative solution that scales much better: RANSAC!