

Feature matching

Semester 2, 2022

Kris Ehinger

Outline

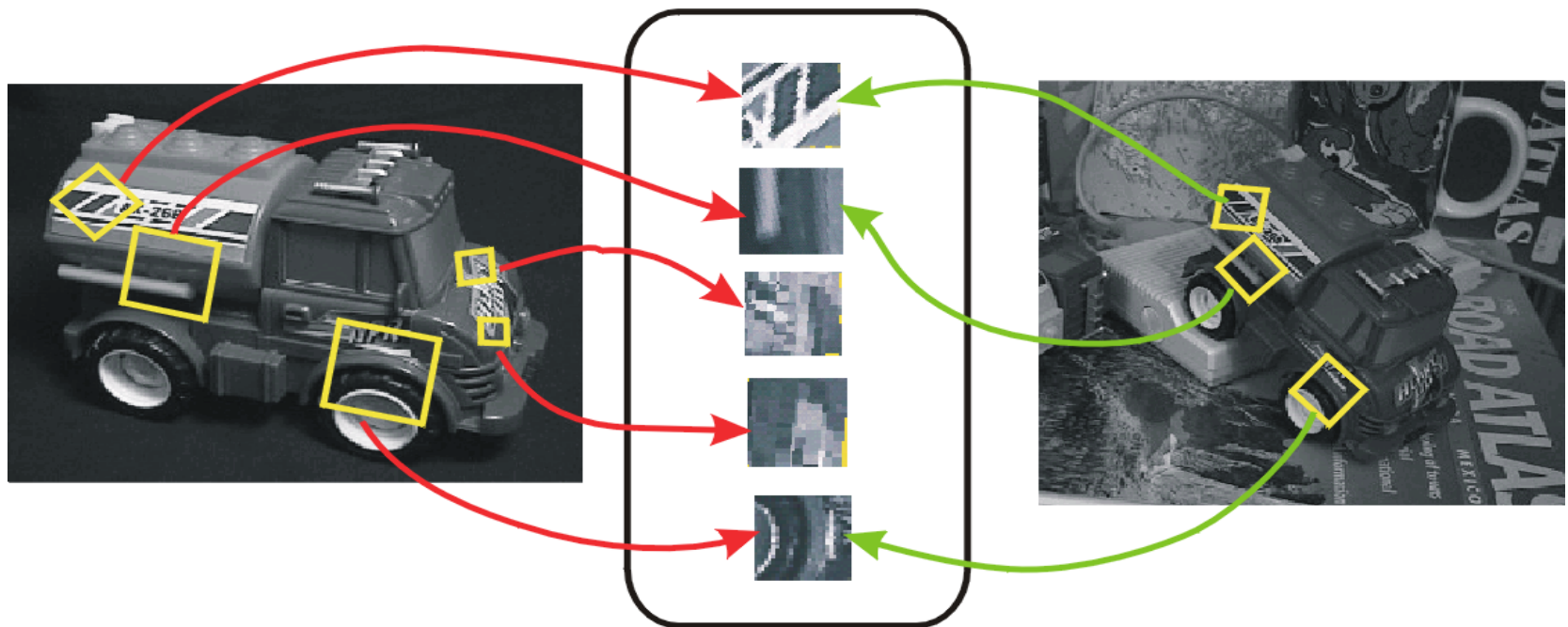
- Feature descriptors
- Introduction to feature matching
- Hough transform
- RANSAC

Learning outcomes

- Explain two methods for detecting structure in images (Hough transform, RANSAC) and how they are commonly applied in images
- Implement an instance-recognition model using feature detection, feature matching, and RANSAC

Feature matching

Feature matching



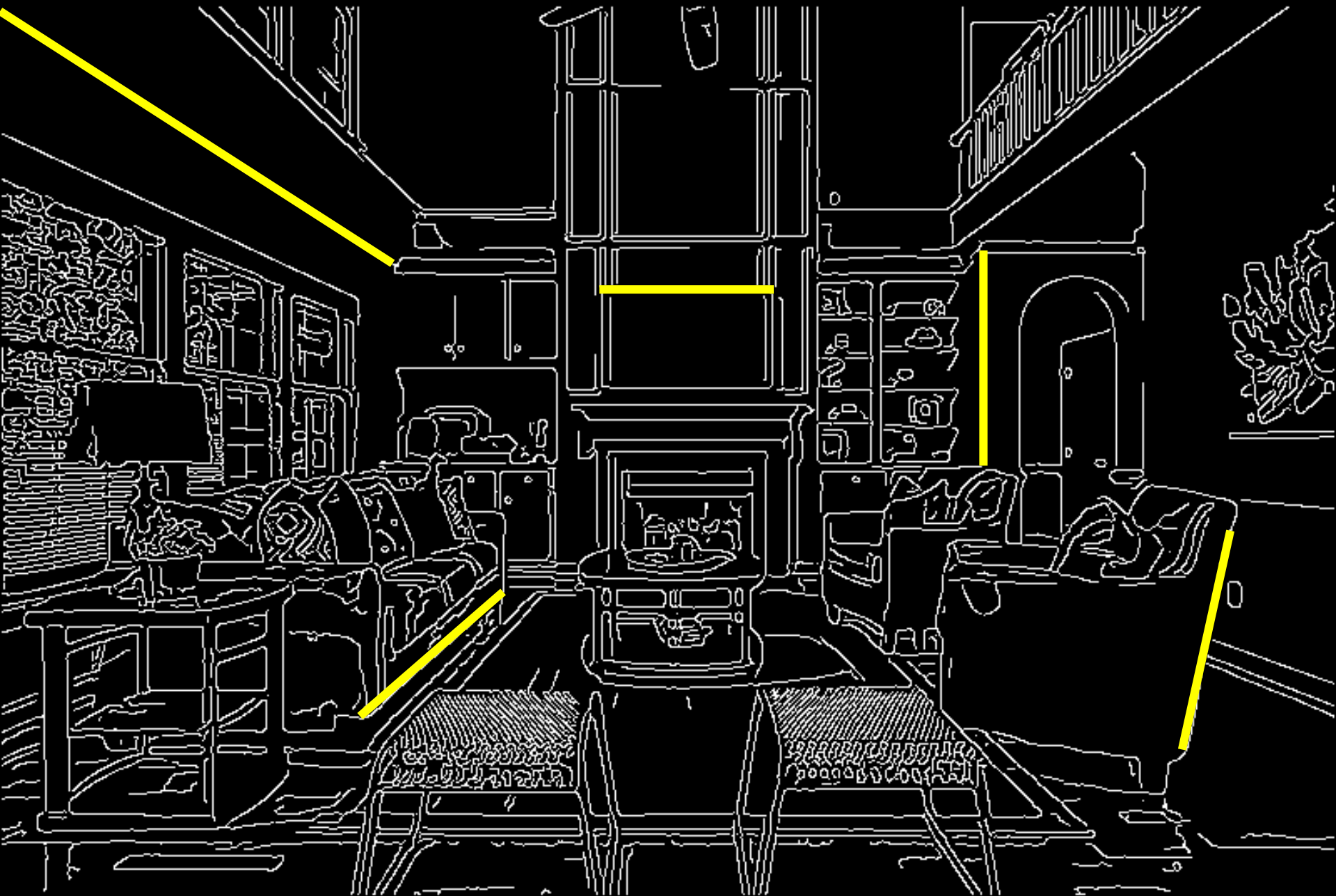
Feature matching as model fitting

- Not just looking for same features, but same spatial arrangement of features
- Model-fitting problem:
 - Propose a model that explains correspondence between matched points
 - Find points that fit model / measure goodness of fit / find model parameters
- Problems:
 - Outliers (data not explained by model)
 - Noise in data that is explained by model

Hough transform



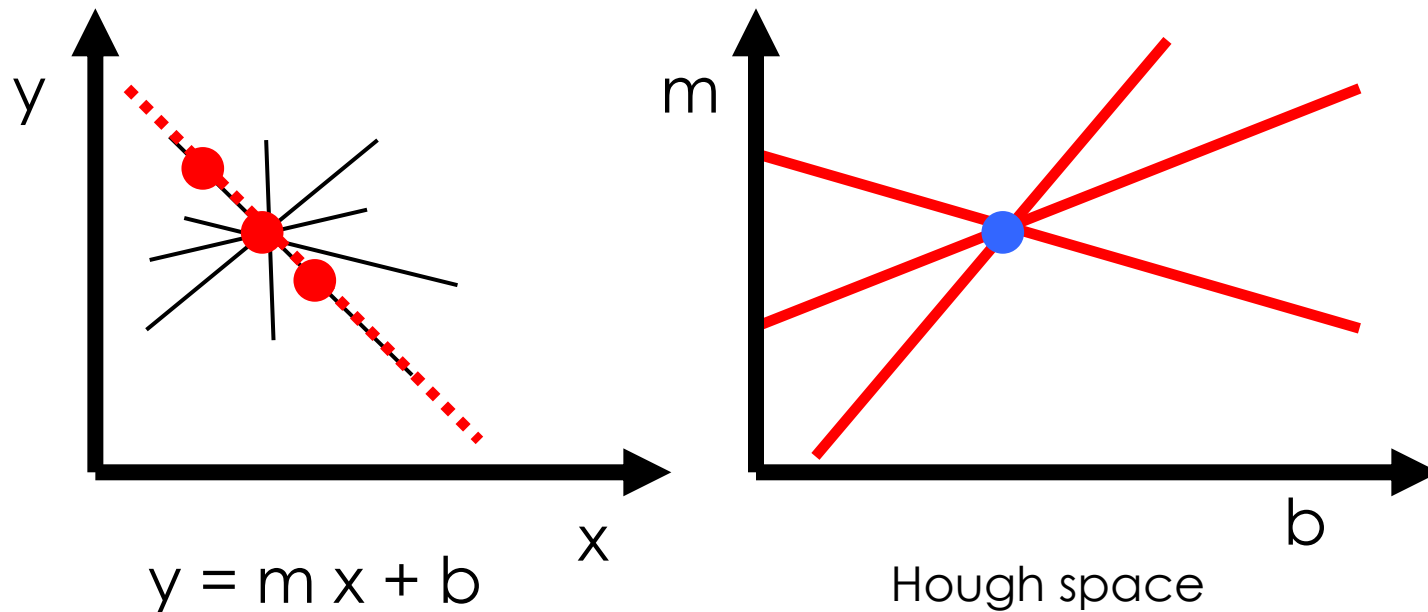
Canny edge map



How to find lines?

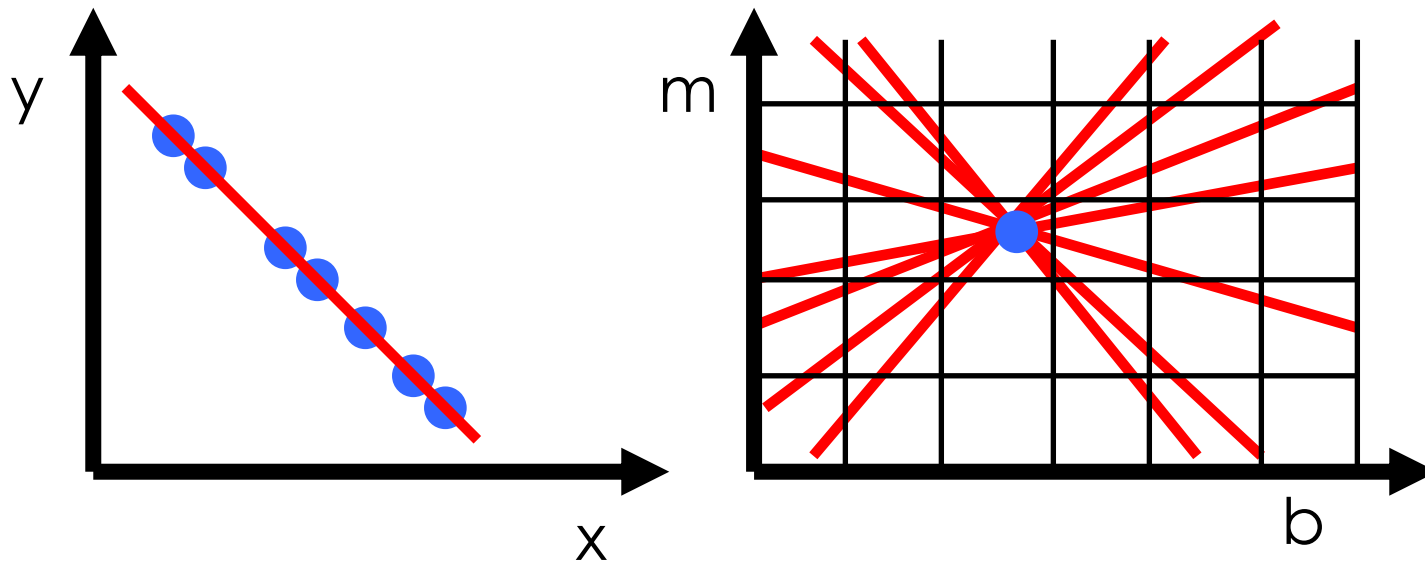
Hough transform

- Each edge point “votes” for the lines that pass through that point
- Identify lines with the most “votes”



Hough transform

- Identifying the parameters with most votes is easy if there is no noise in point locations
- Usually there is some noise



Hough transform

- Solution to noise: bin parameters
- Points vote for bins

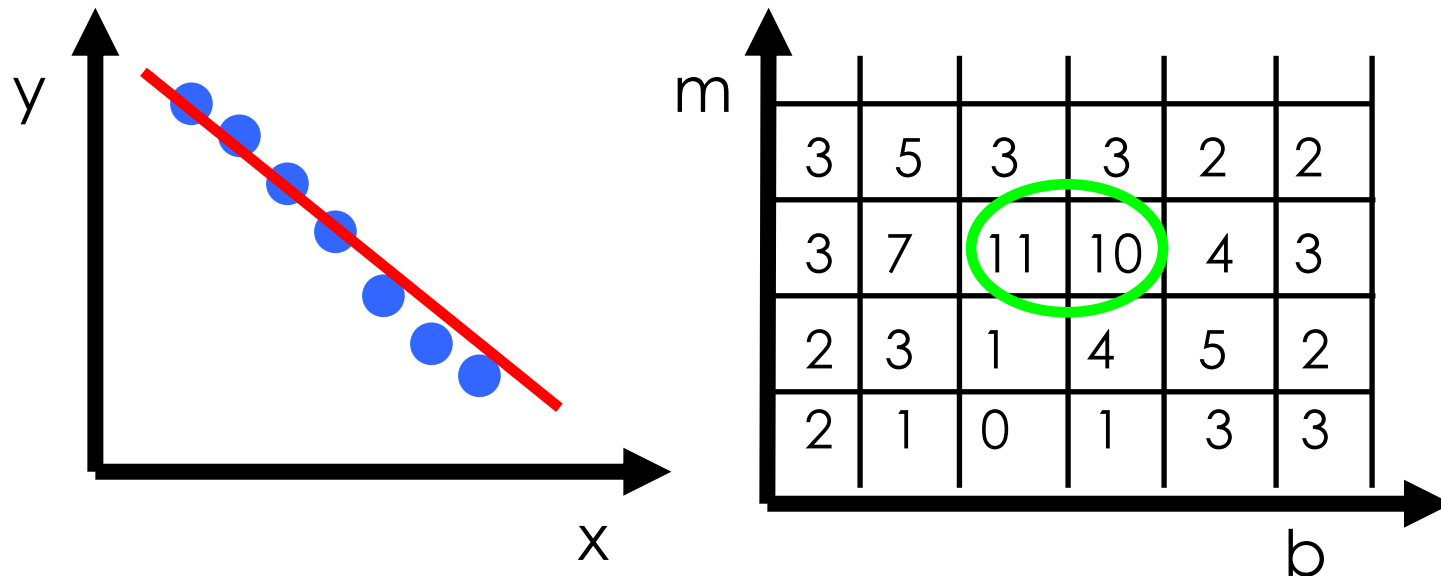
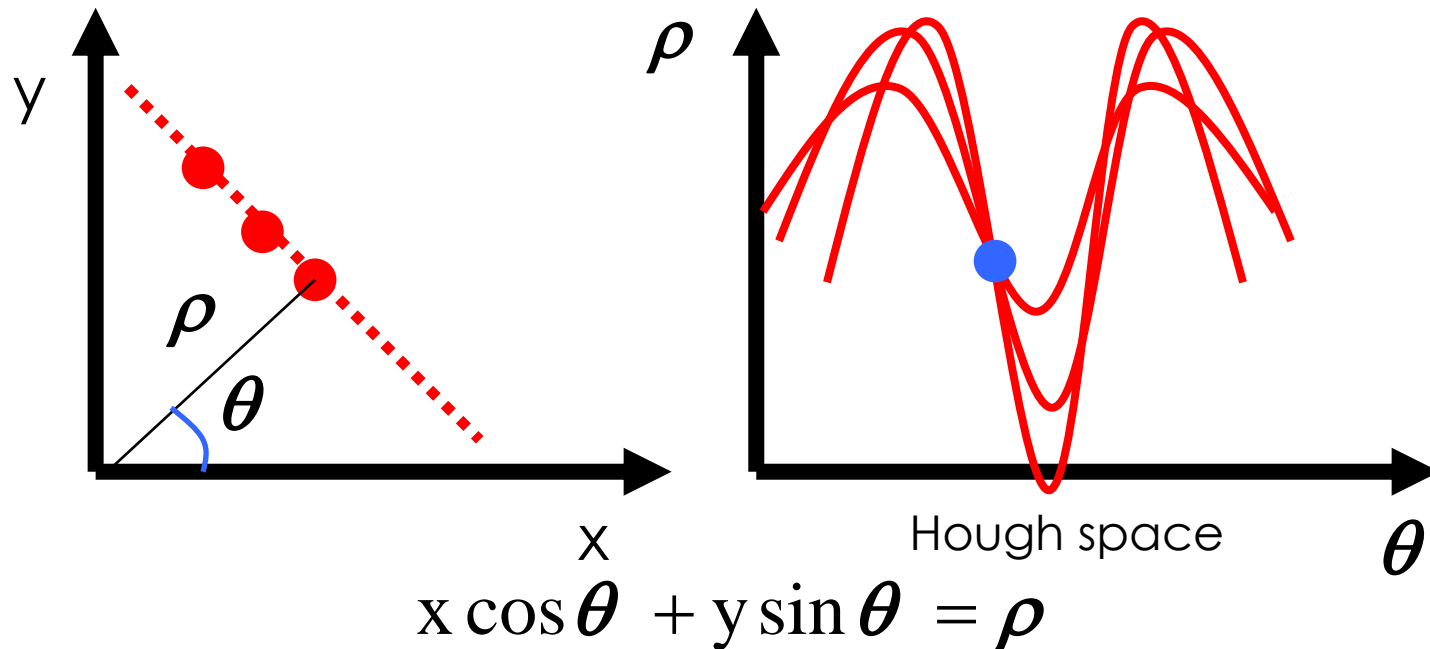


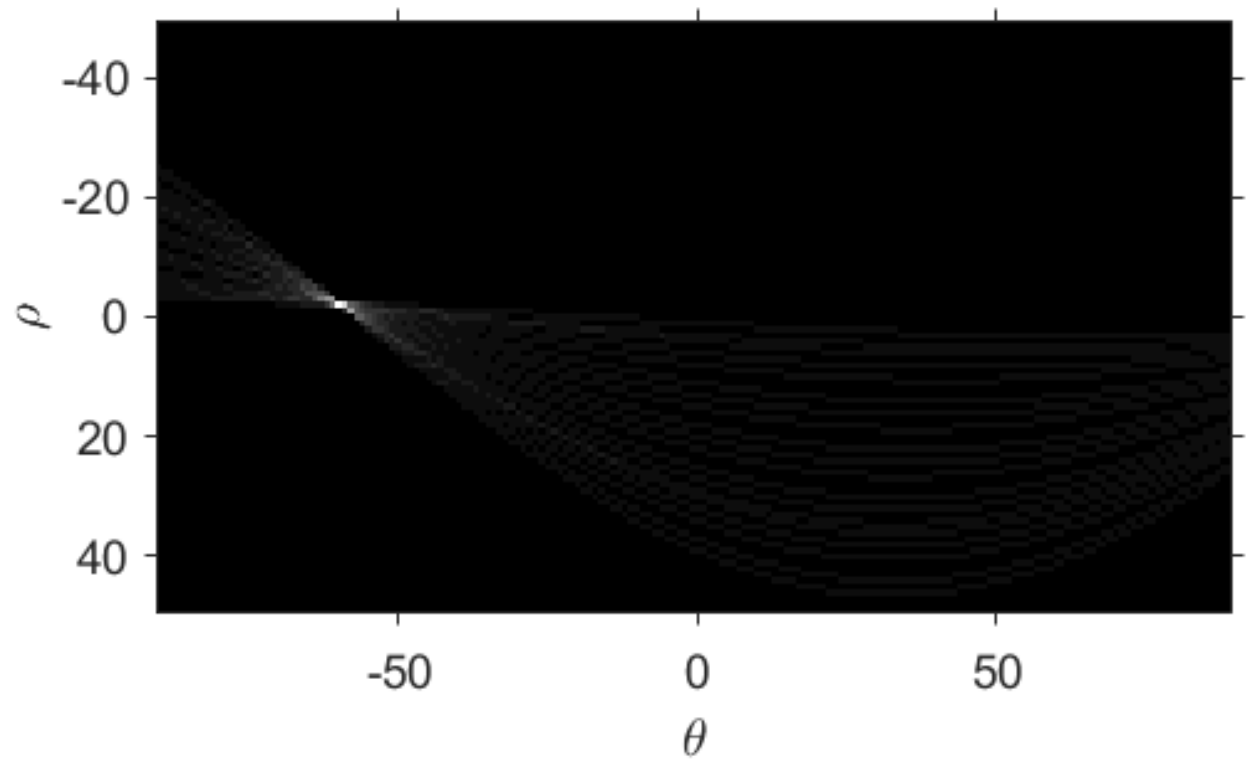
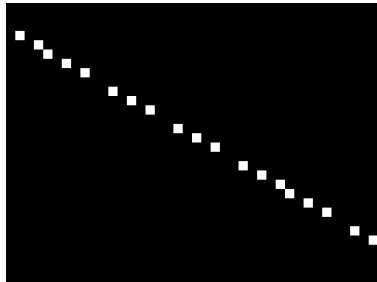
Figure: S. Savarese

Hough transform

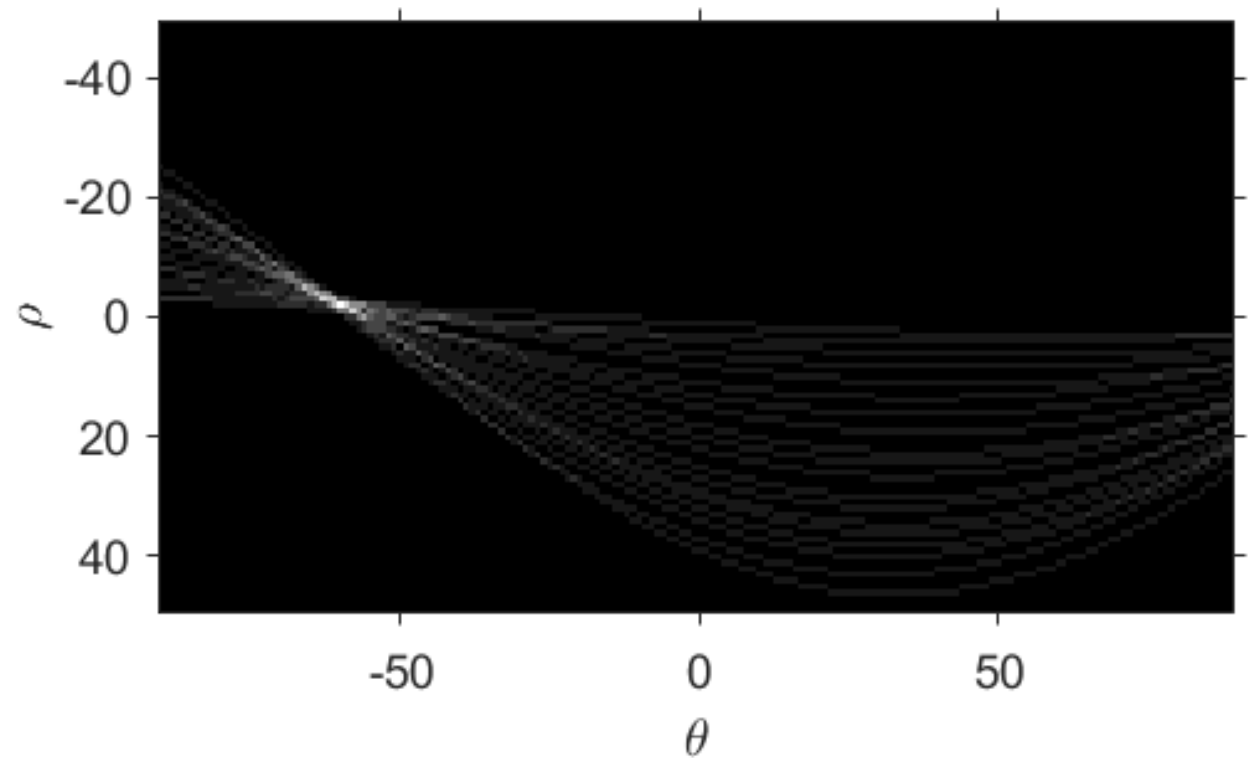
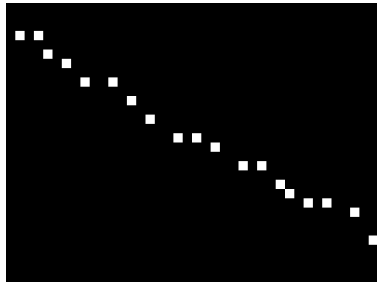
- Another problem: slope and intercept (m,b) are unbounded
- Solution: use a polar representation of (m,b)



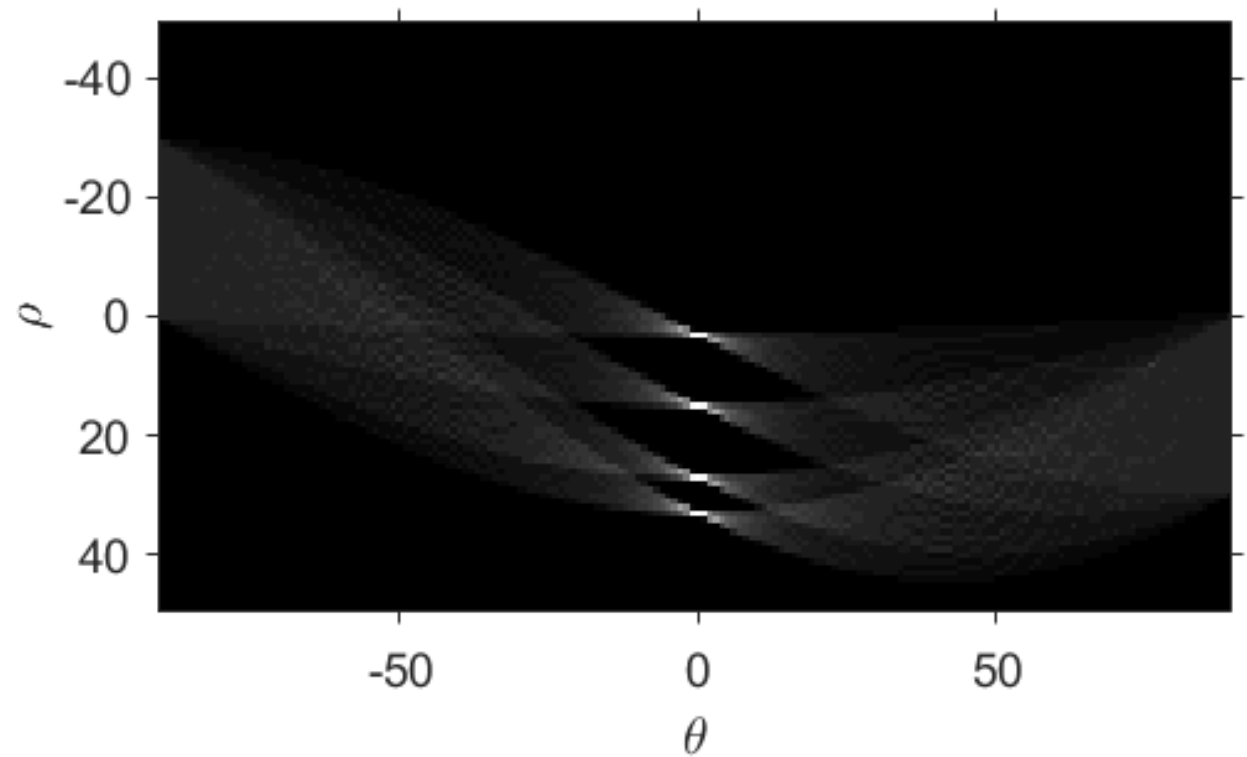
Example: Hough transform



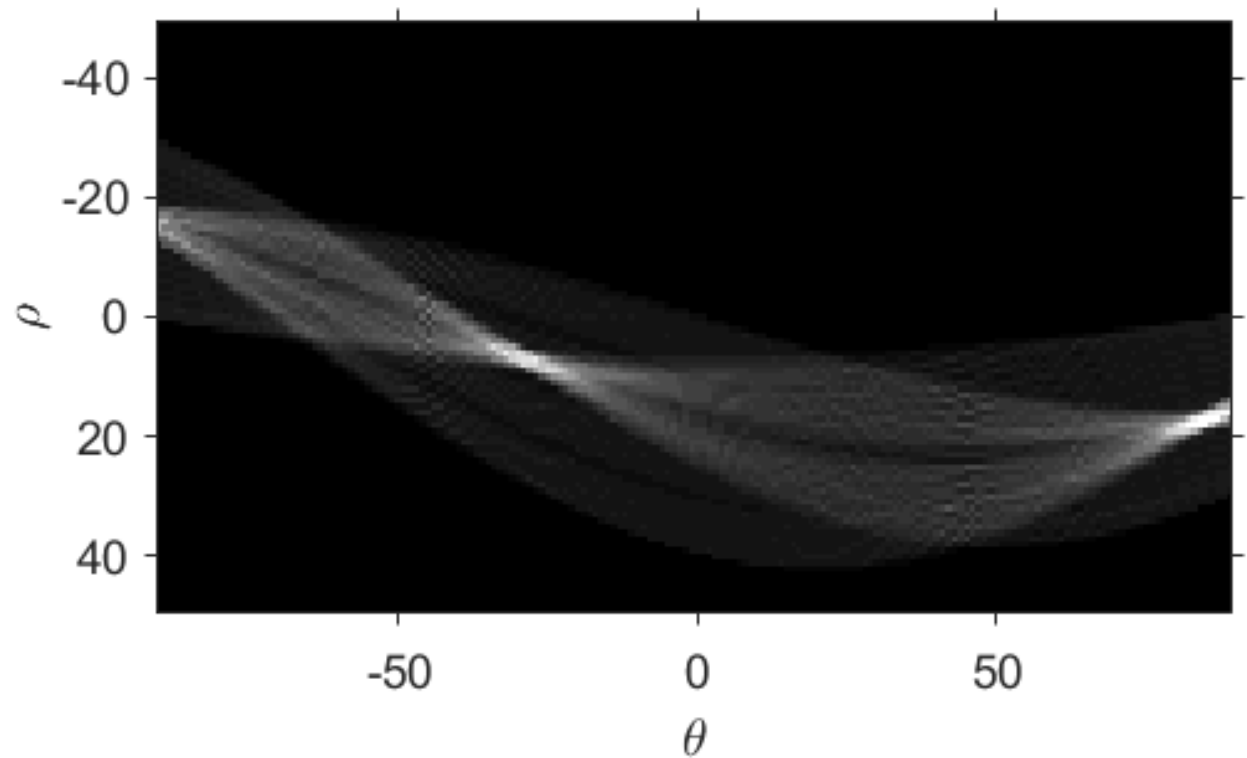
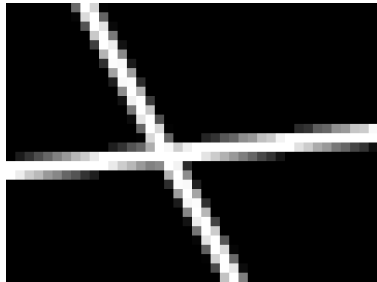
Example: Hough transform



Example: Hough transform



Example: Hough transform



Hough transform parameters

- Bin size
- Threshold for peaks
- Number of peaks (= number of lines)
- Minimum line length
- Maximum allowed gap (to treat segments as the same line)

Canny edge map

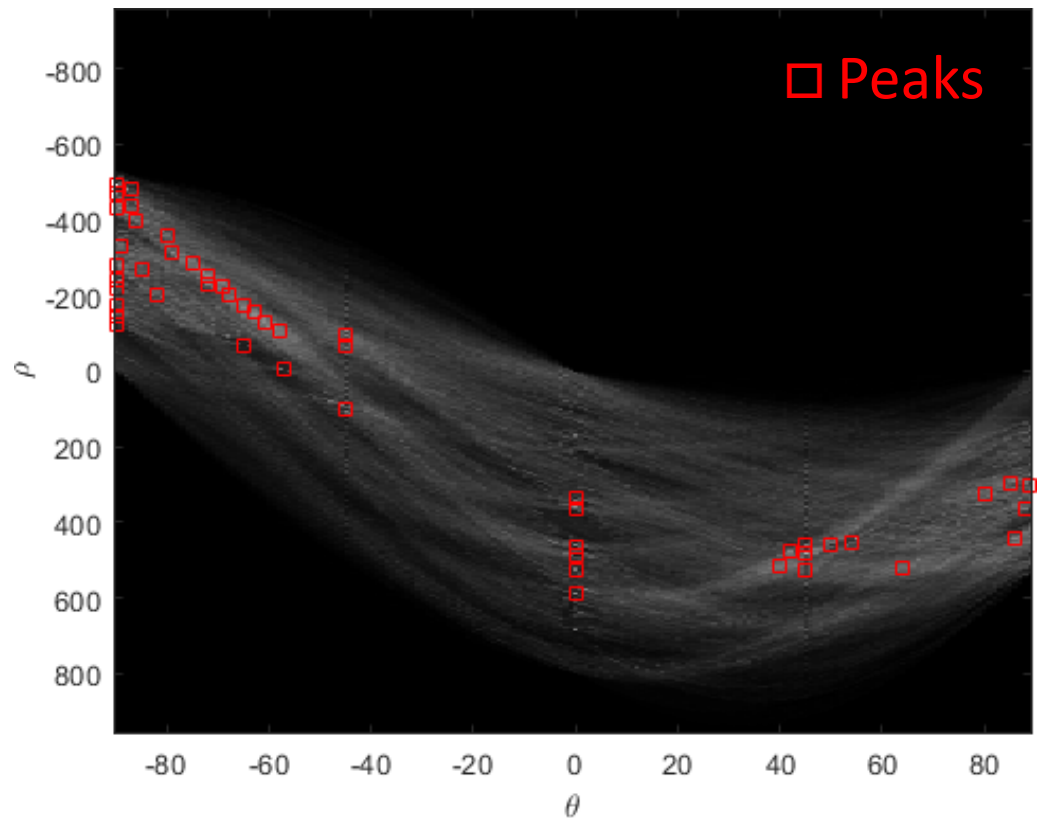


Hough transform peaks

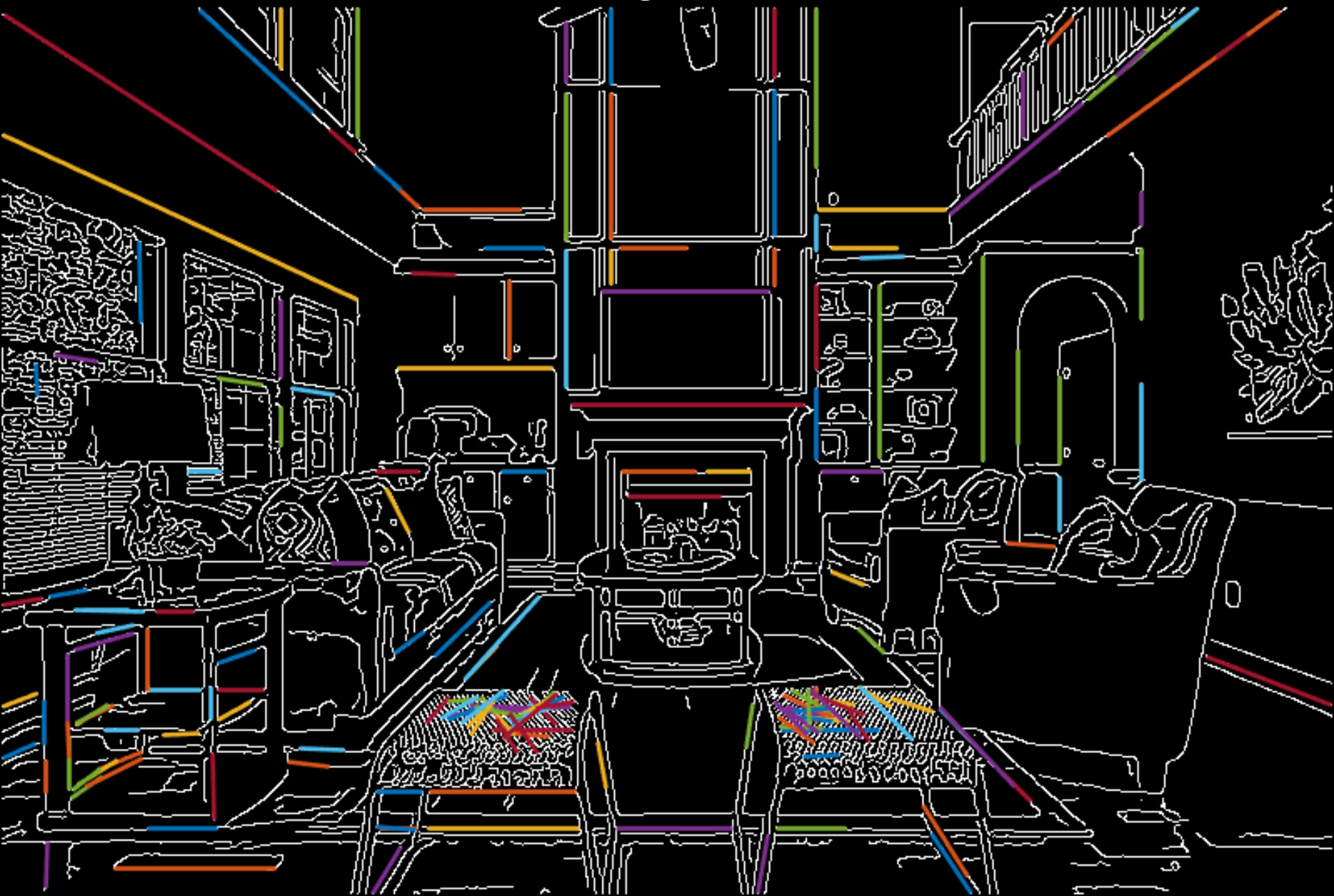
Canny edge map



Hough transform

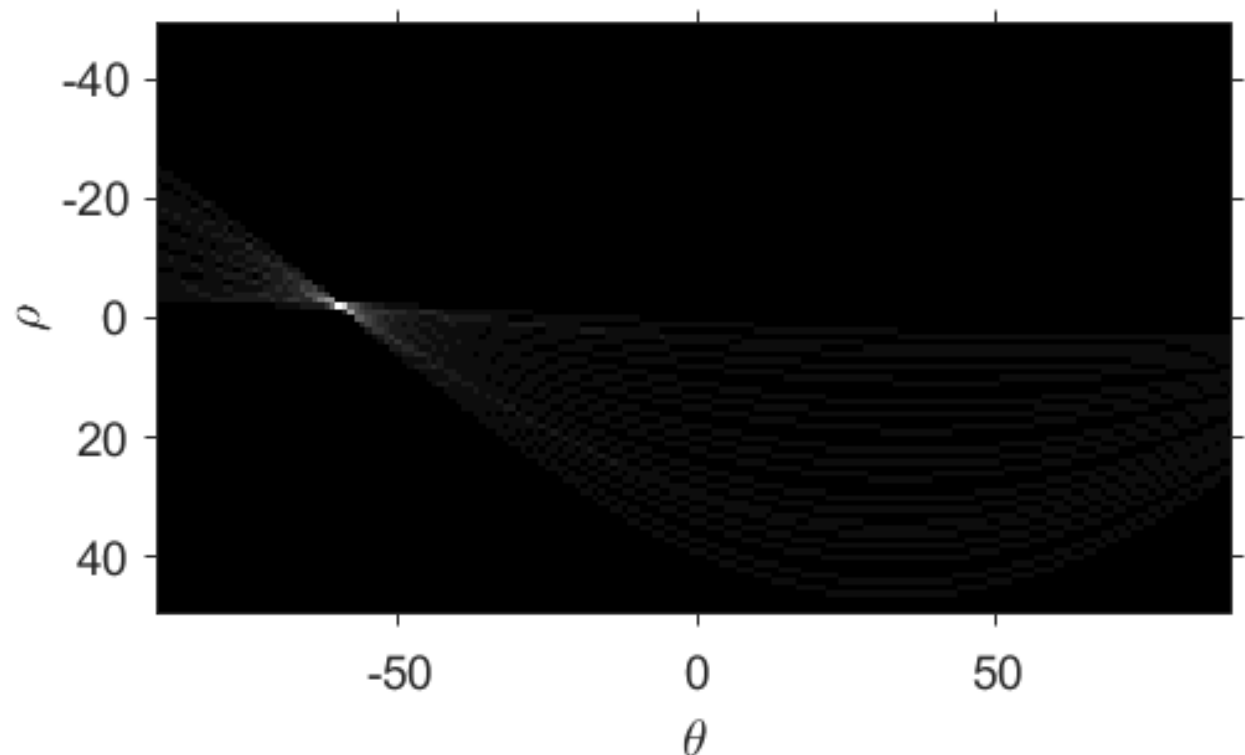
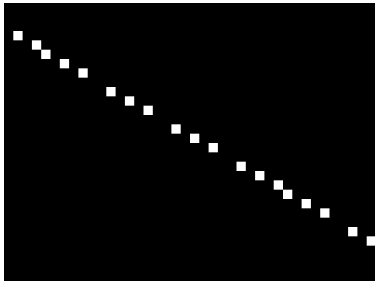


Hough lines



Hough transform efficiency

- Hough transform is a grid search for parameters
- What happens as N parameters increases?



Summary

- Hough transform is a method for detecting structure in images
- Each pixel “votes” for parameters
- Common applications:
 - Line detection
 - Circle detection
- Basically grid search over all possible values of each parameter, so limited to just a few parameters

RANSAC

RANSAC

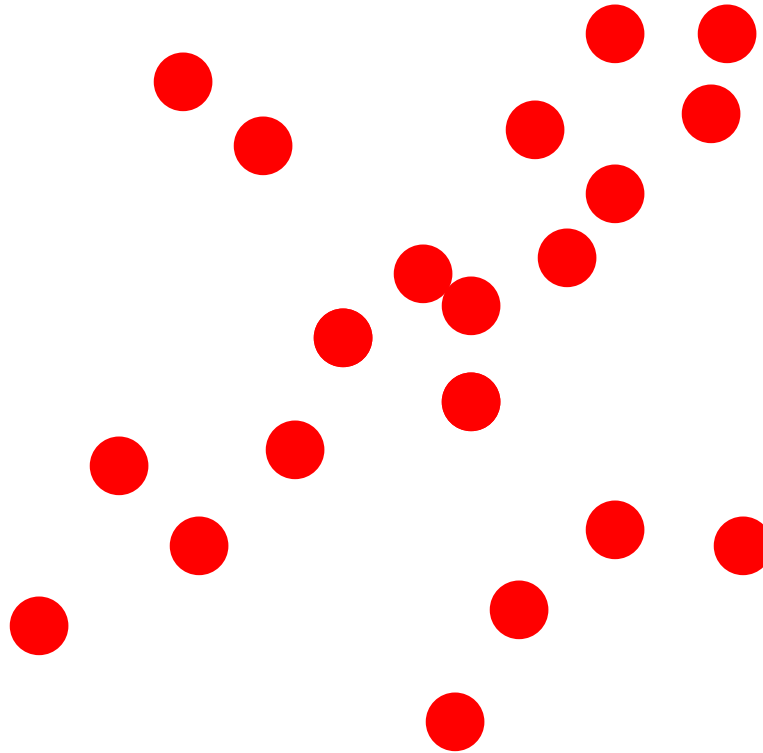
- RANSAC = RANdom Sample Consensus
- Like Hough transform, points “vote” for the model that explains those points
- Iteratively:
 - Sample N points at random (when N is the number of points needed to fit the model)
 - Calculate model parameters
 - Count the number of points that are explained by the model (inlier points)
- Best model = model that explains the most points

Example: RANSAC line fitting

1. Randomly sample the number of points needed to fit a line (2)
2. Find line parameters
3. Count inliers (points that fit this line)

Repeat until max iterations

Final model: model with highest N_{inliers}



Example: RANSAC line fitting

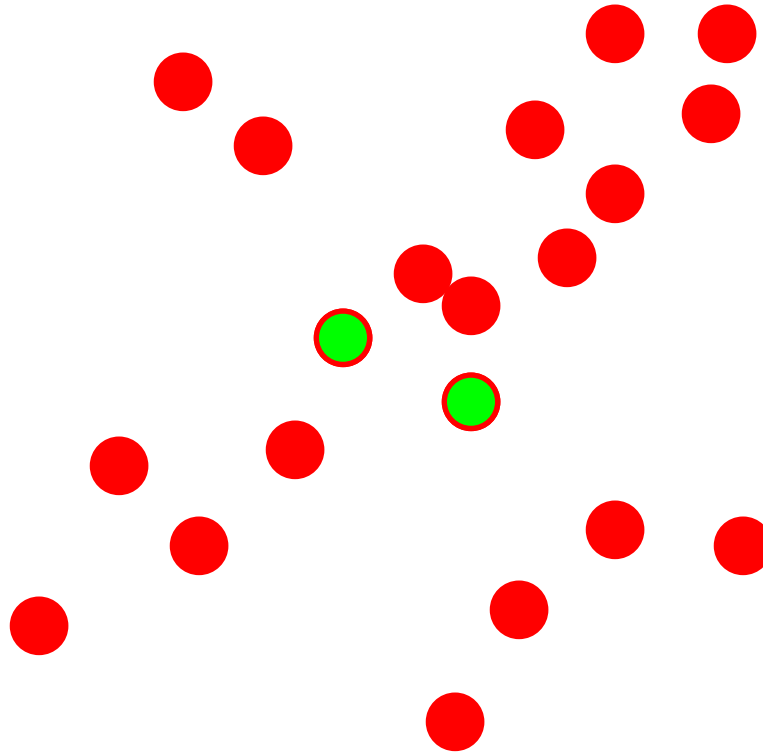
1. Randomly sample the number of points needed to fit a line (2)

2. Find line parameters

3. Count inliers (points that fit this line)

Repeat until max iterations

Final model: model with highest N_{inliers}



Example: RANSAC line fitting

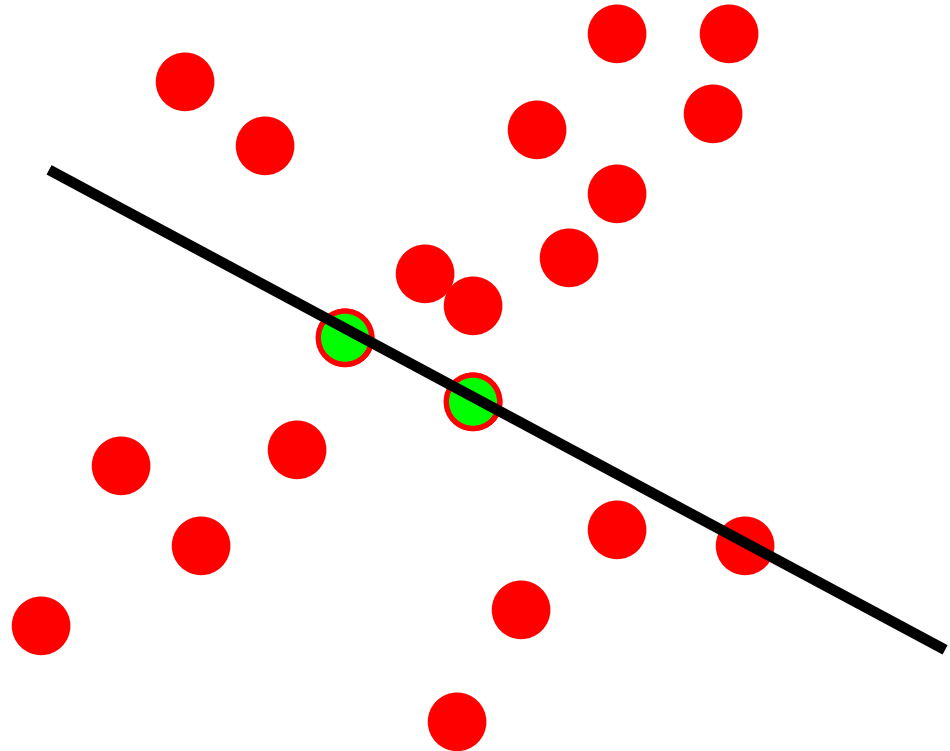
1. Randomly sample the number of points needed to fit a line (2)

2. Find line parameters

3. Count inliers (points that fit this line)

Repeat until max iterations

Final model: model with highest N_{inliers}



Example: RANSAC line fitting

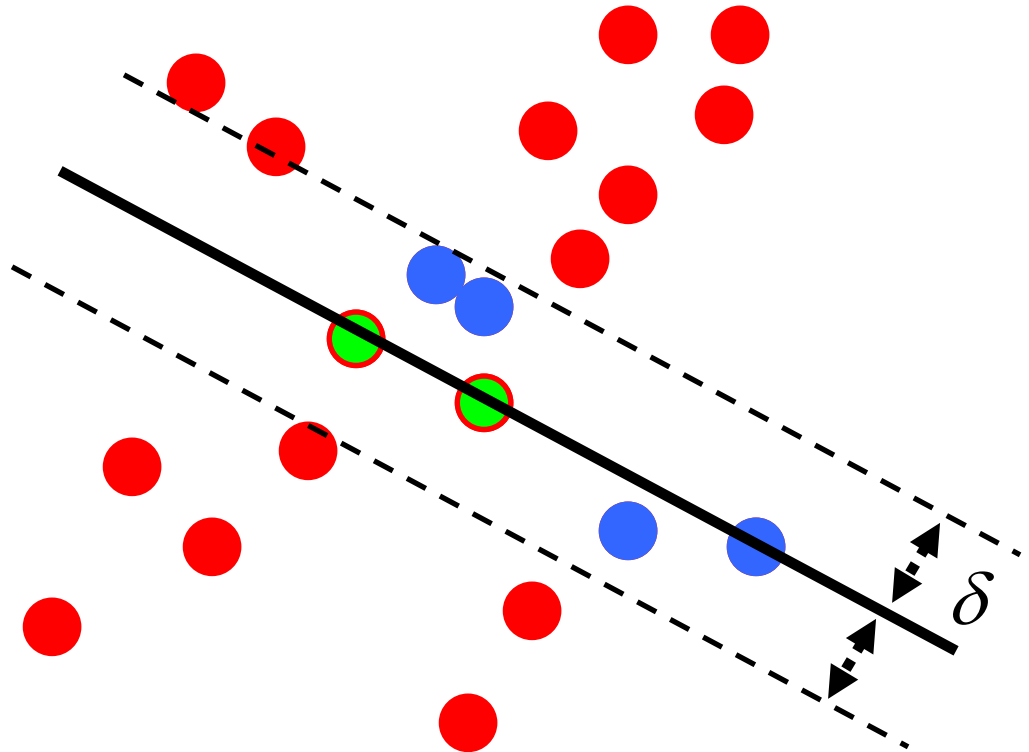
1. Randomly sample the number of points needed to fit a line (2)

2. Find line parameters

3. Count inliers (points that fit this line)

Repeat until max iterations

Final model: model with highest N_{inliers}



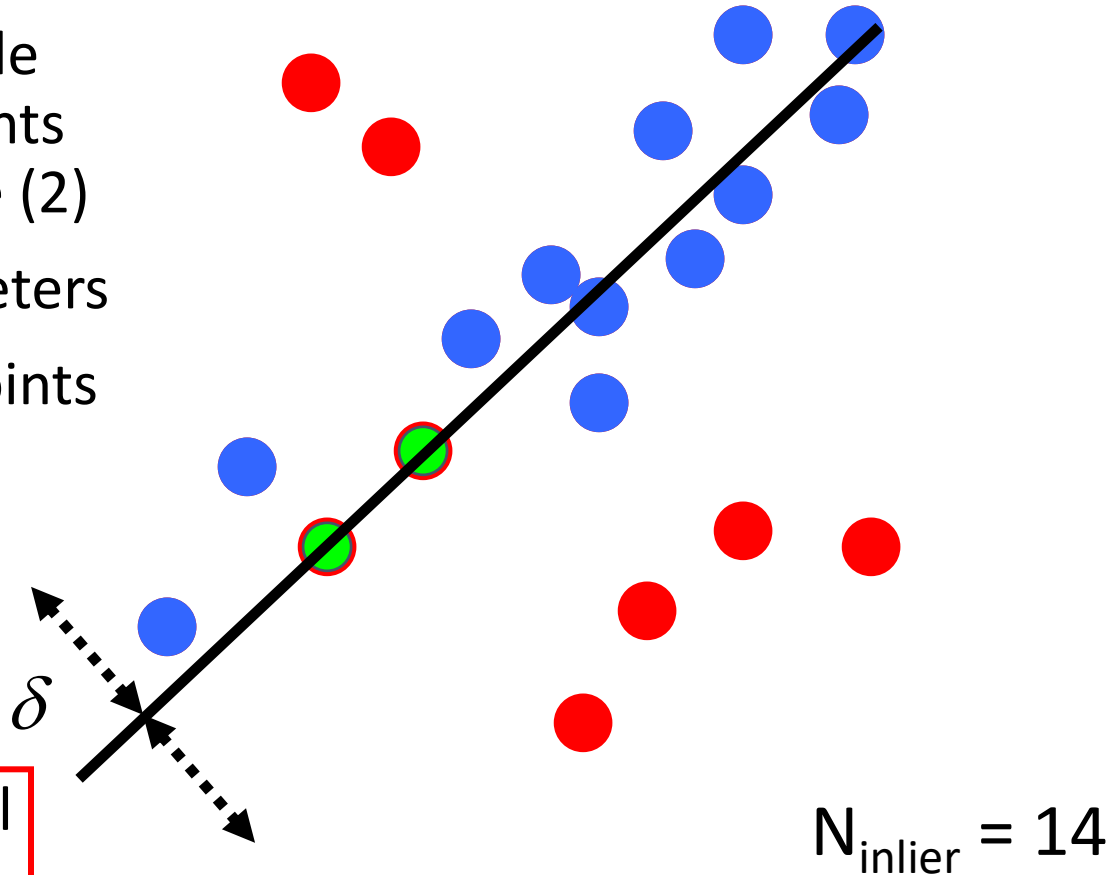
$N_{\text{inlier}} = 6$

Example: RANSAC line fitting

1. Randomly sample the number of points needed to fit a line (2)
2. Find line parameters
3. Count inliers (points that fit this line)

Repeat until max iterations

Final model: model with highest N_{inliers}

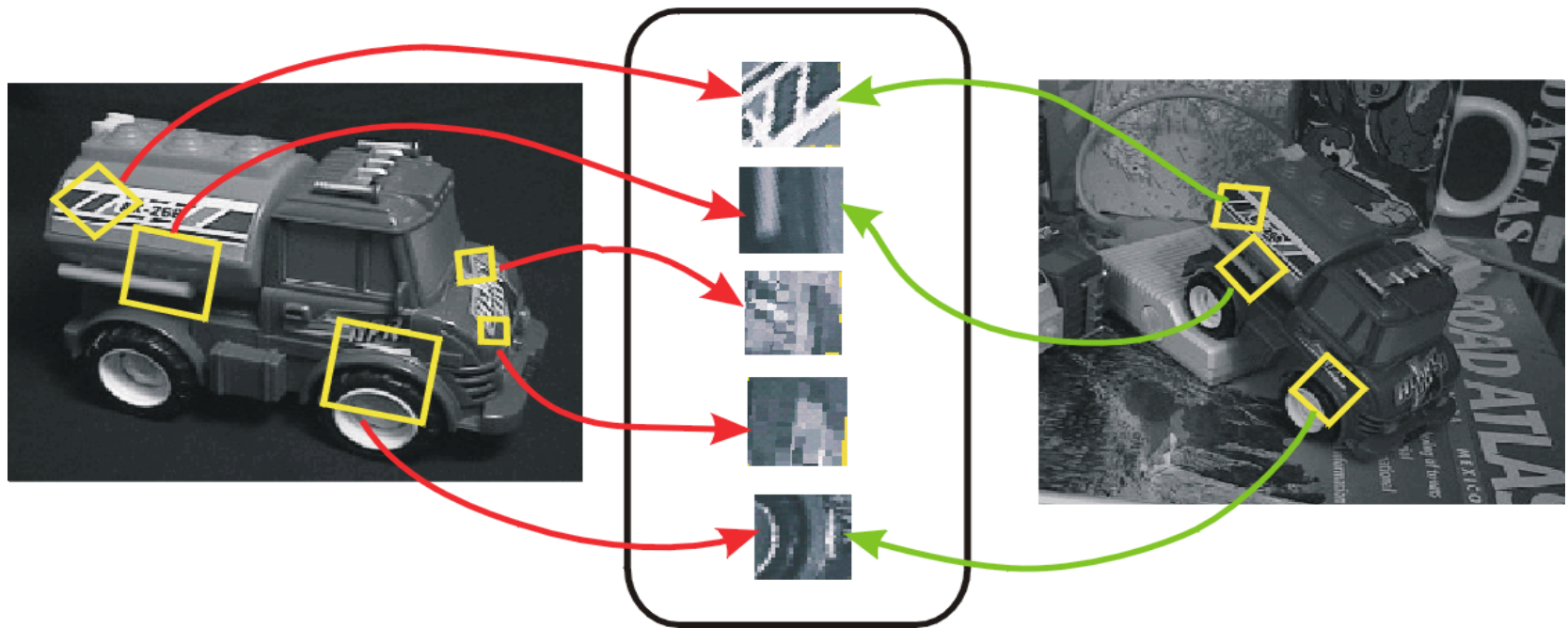


RANSAC parameters

- Number of iterations
 - Should be high enough that it is very likely (e.g., $\text{prob}=0.99$) that you will obtain at least one sample with no outliers
 - Example: Half of your data is outliers, and you want to fit a line. How many samples?
- Threshold for inliers
 - Choose δ so that a good point with noise is likely (e.g., $\text{prob}=0.95$) within threshold

Application: Instance recognition

- Does the second image contain the same features in the *same configuration*?



Feature matching



Step 1. Find keypoints + descriptors in each image

Feature matching



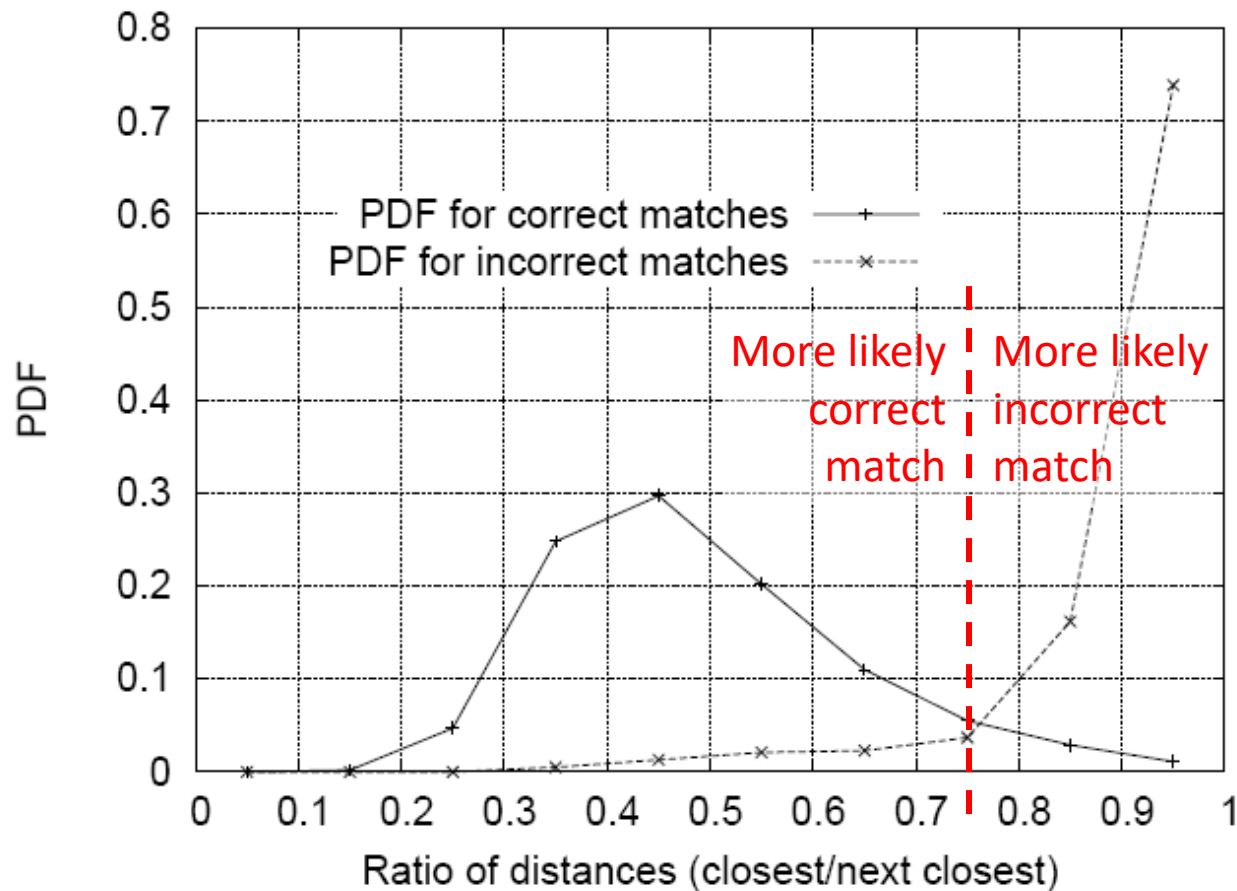
Step 2. Find candidate matches: for each keypoint in image 1, find most similar match in image 2

Feature matching



- Problems:
 - Many keypoints, many similar features – false matches are likely
 - For more robust matching, consider ratio (similarity to best match / similarity to second best)

Reducing false matches



D. G. Lowe (2004)

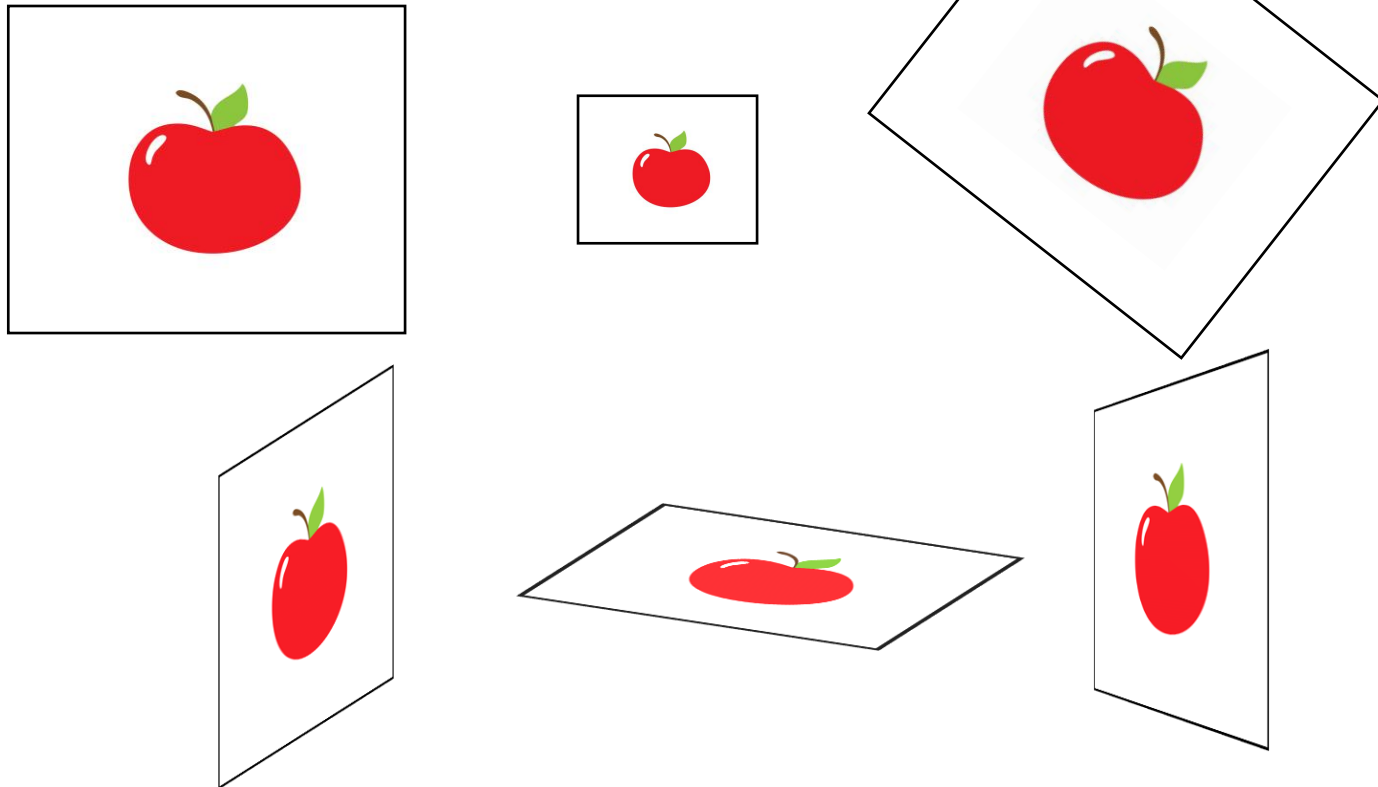
Feature matching

- Exhaustive matching is very slow ($O(m*n)$),
m,n = number of keypoints in each image
- More efficient option: approximate nearest neighbours
 - Faster, but may not find the closest match
- Even with ratio matching, proportion of false matches is likely to be very high

Step 3. Use RANSAC to find subset of matches that can be explained by a transformation model

What model?

- Possible transformations:




Affine transformations

- Affine transformation is any combination of translation, scale, rotation, and shear


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine



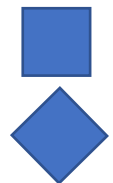
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale




$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \Theta & -\sin \Theta \\ \sin \Theta & \cos \Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

Projective transformation

- Projective transformation combines affine transformation with a projective warp

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

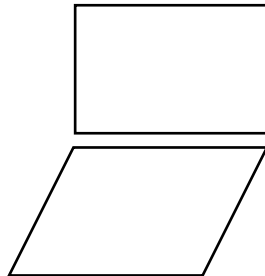
Projective

Affine transformation

Lines map to lines

Parallel lines remain parallel

Ratios are preserved

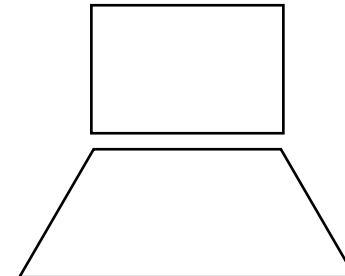


Projective transformation

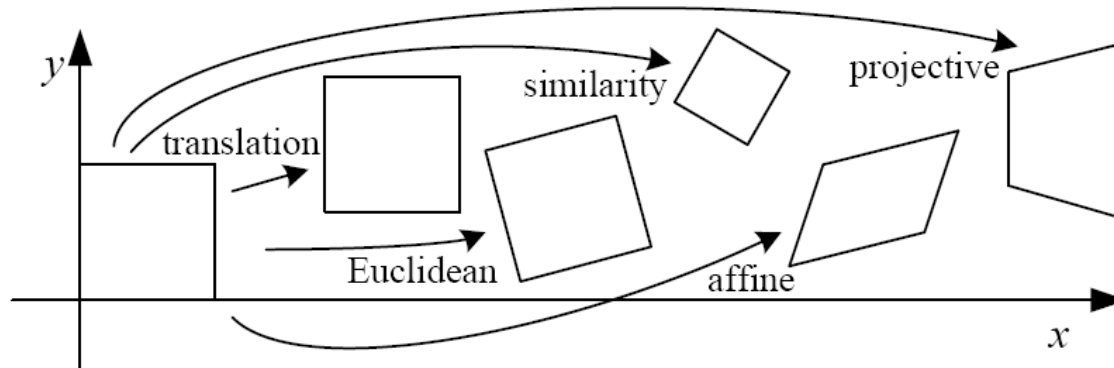
Lines map to lines

Parallel lines may not remain parallel

Ratios are not preserved



2D planar transformations



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

Feature matching with RANSAC

1. Randomly sample the number of points needed to fit model (e.g., 4 for projective)
2. Compute model parameters from points
3. Count inlier points

Repeat until max iterations, take model with most inliers



Summary

- Feature matching with RANSAC finds matching keypoints across images
- Finds matching points across views, and computes the transformation that relates those points
- Applications:
 - Instance recognition
 - Video stabilisation
 - Panorama stitching
 - Finding planes, vanishing points, etc. in images
 - 3D reconstruction