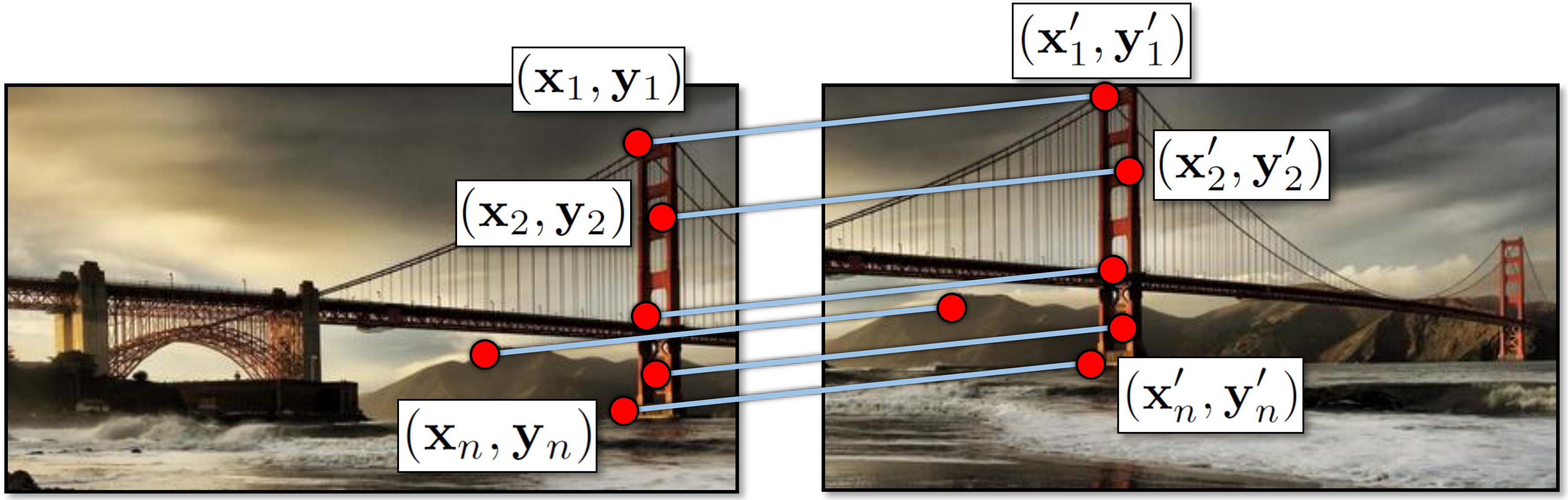
CAP 4453
Robot Vision

Dr. Gonzalo Vaca-Castaño
gonzalo.vacacastano@ucf.edu

Short Review from last class

Image warping



How do we find point correspondences automatically?

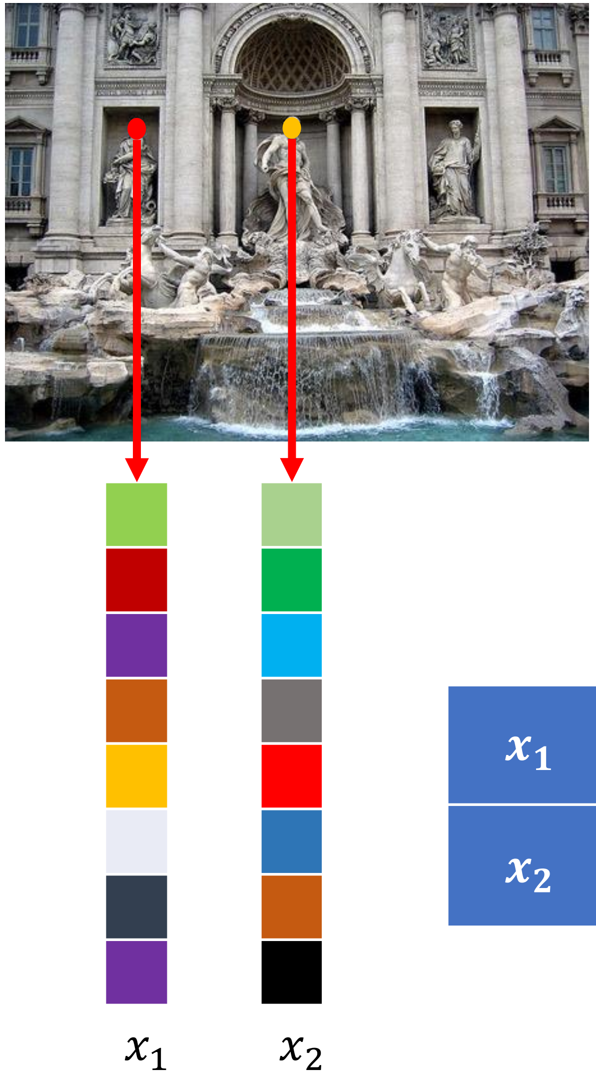
Robot Vision

11. Feature points description

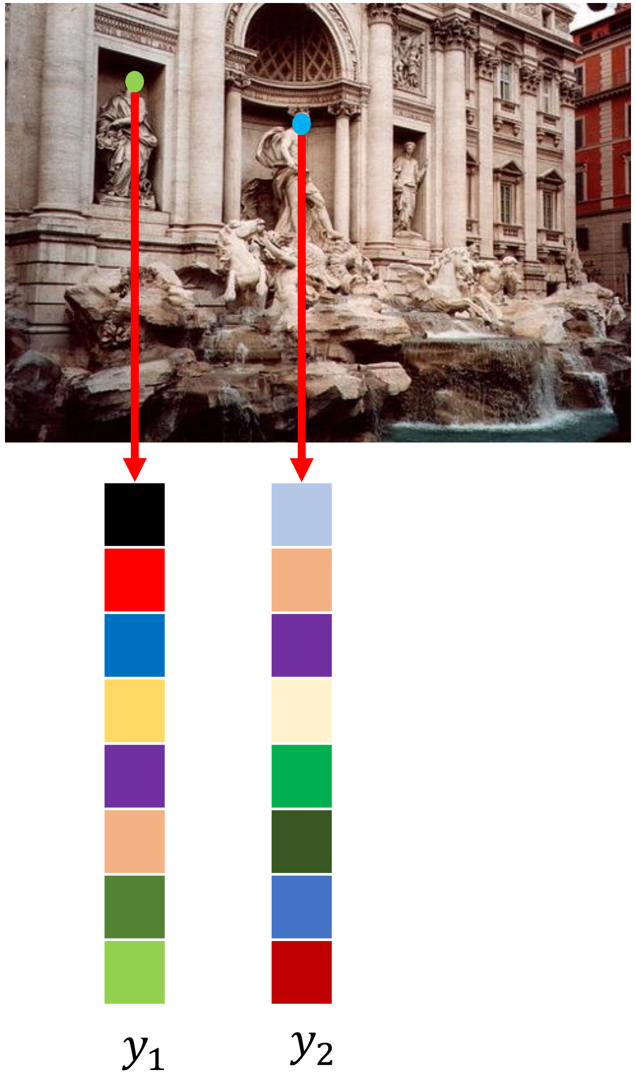
Outline

- Motivation
- Detecting key points
    - Harris corner detector
    - Blob detection
- Feature descriptors
    - HOG
    - MOPS
- SIFT

Feature matching



	y_1	y_2
x_1	$d(x_1, y_1)$	$d(x_1, y_2)$
x_2	$d(x_2, y_1)$	$d(x_2, y_2)$



Scale Invariant Feature Transform (SIFT)

- Lowe., D. 2004, IJCV

cited > 58K

Distinctive Image Features from Scale-Invariant Keypoints

DAVID G. LOWE
Computer Science Department, University of British Columbia, Vancouver, B.C., Canada
Lowe@cs.ubc.ca

Received January 10, 2003; Revised January 7, 2004; Accepted January 22, 2004

Abstract. This paper presents a method for extracting distinctive invariant features from images that can be used to perform reliable matching between different views of an object or scene. The features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. This paper also describes an approach to using these features for object recognition. The recognition proceeds by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. This approach to recognition can robustly identify objects among clutter and occlusion while achieving near real-time performance.

Keywords: invariant features, object recognition, scale invariance, image matching

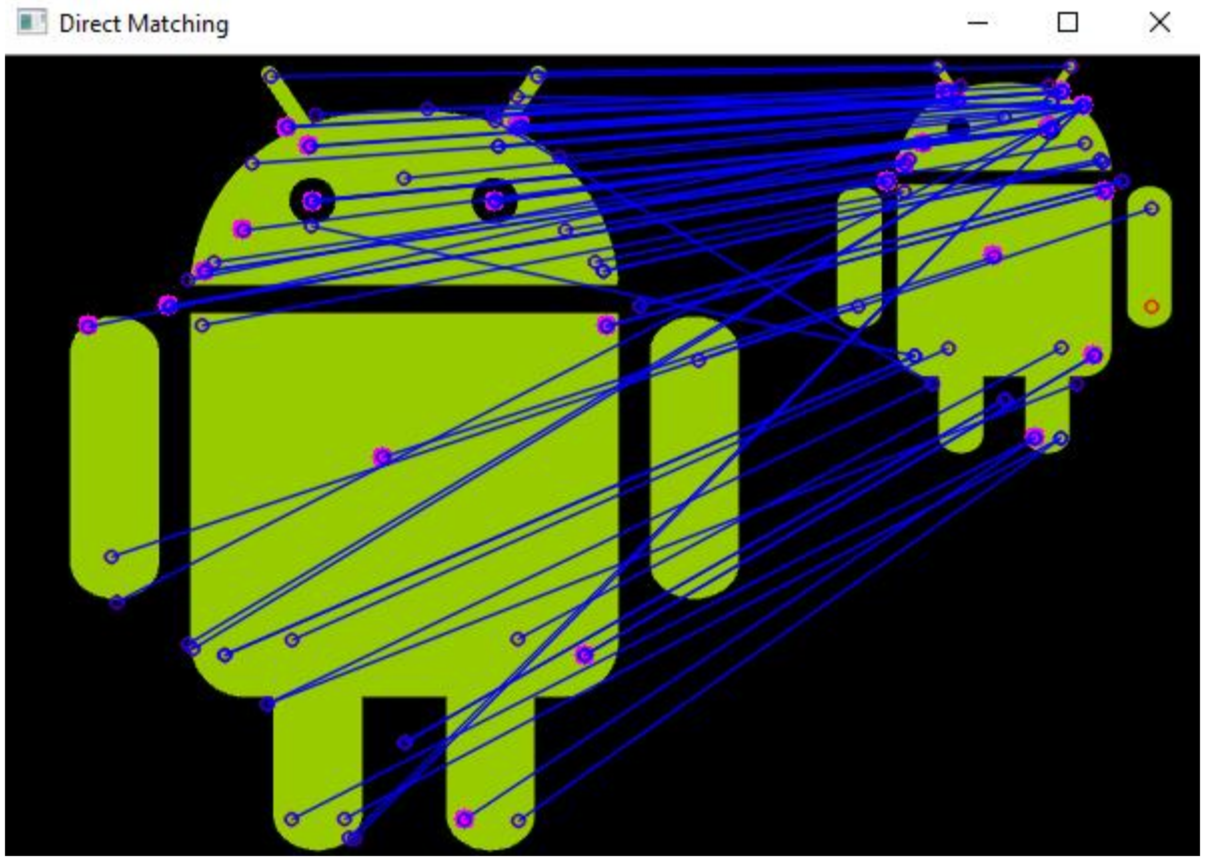
1. Introduction

Image matching is a fundamental aspect of many problems in computer vision, including object or scene recognition, solving for 3D structure from multiple images, stereo correspondence, and motion tracking. This paper describes image features that have many properties that make them suitable for matching differing images of an object or scene. The features are invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. They are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for object and scene recognition.

The cost of extracting these features is minimized by taking a cascade filtering approach, in which the more expensive operations are applied only at locations that pass an initial test. Following are the major stages of computation used to generate the set of image features:

1. *Scale-space extrema detection:* The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation.
2. *Keypoint localization:* At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
3. *Orientation assignment:* One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.

SIFT Matching based on distance



Lowe's Ratio Test

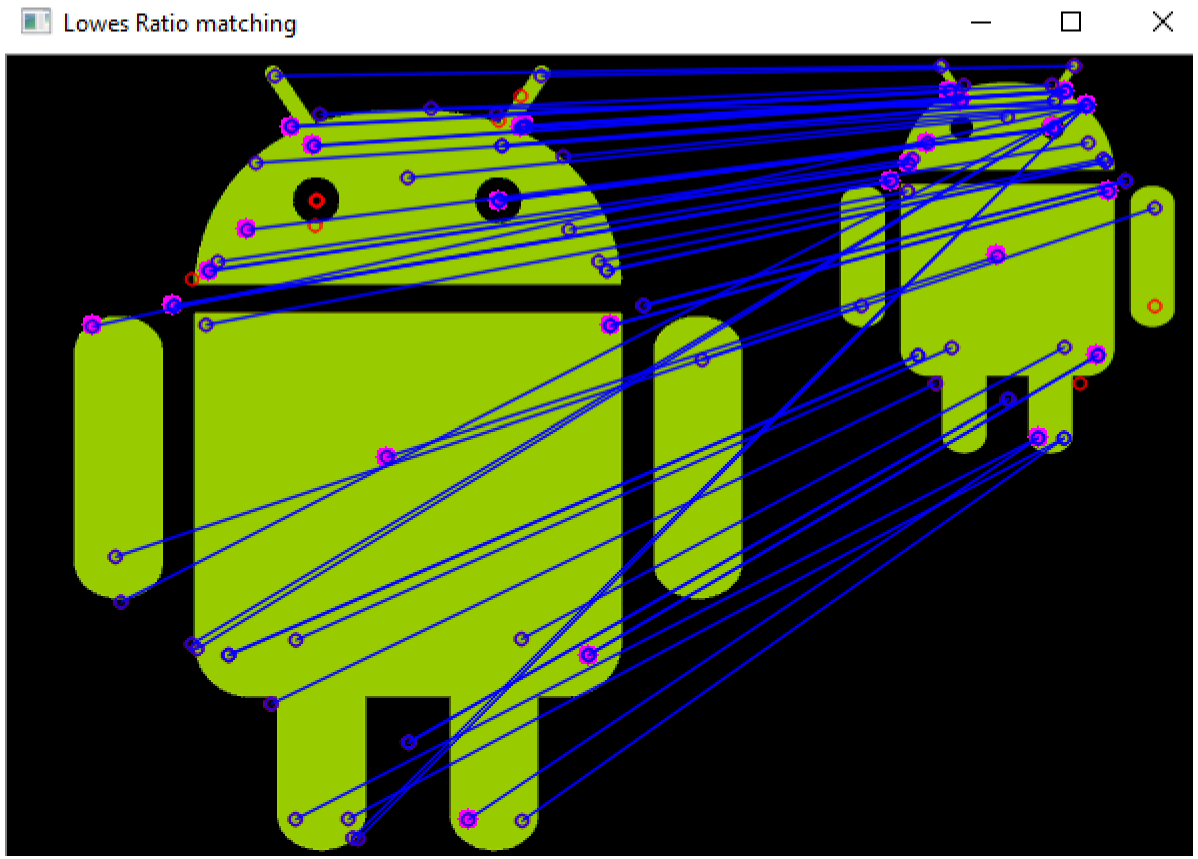
Algorithm:

1. First, we compute the distance between feature f_i in image one and all the features f_j in image two.
2. We choose feature f_c in image two with the minimum distance to feature f_i in image of one as our closest match.
3. We then proceed to get feature f_s the feature in image two with the second closest distance to the feature f_i.
4. Then we find how much nearer our closest match f_c is over our second closest match f_s through the distance ratio.
5. Finally we keep the matches with *distance ratio* < *distance ratio threshold*.

Lowe’s ratio

```
## Ratio test
matches = matcher.knnMatch(descs1, descs2, 2)     #each feature descriptor is matched to n number of closest neighbors. (n=2 in this case)
matchesMask = [[0,0] for i in range(len(matches))]
src_pts = []
dst_pts = []
good=[]
for i, (m1,m2) in enumerate(matches):  # each keypoint of the first image is matched with a number of keypoints from the second image.
    # We keep the 2 best matches for each keypoint (best matches = the ones with the smallest distance measurement).
    # Lowe's test checks that the two distances are sufficiently different.
    # If they are not, then the keypoint is eliminated and will not be used for further calculations.
    if m1.distance < 0.7 * m2.distance:
        good.append(m1)
        matchesMask[i] = [1,0]
        ## Notice: How to get the index
        pt1 = kpts1[m1.queryIdx].pt
        pt2 = kpts2[m1.trainIdx].pt
        src_pts.append(pt1)
        dst_pts.append(pt2)
```

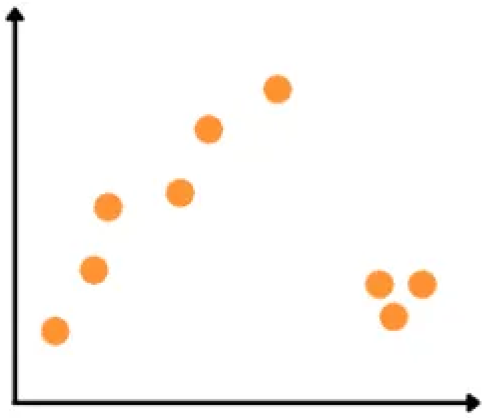
Lowe's Ratio



Ransac

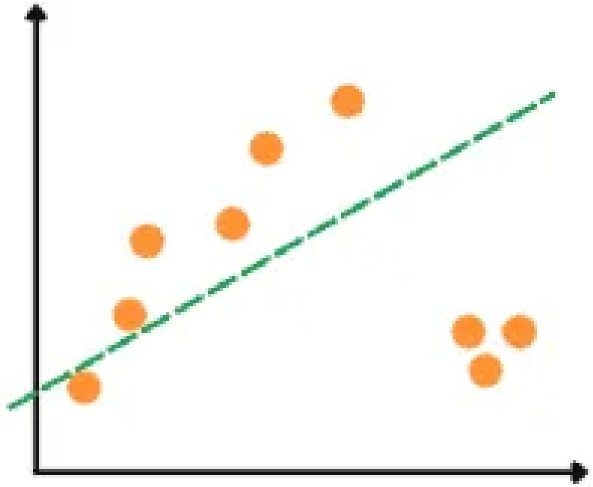
RANdom **SA**mple **C**onsensus

RANSAC (example applied to linear regression)



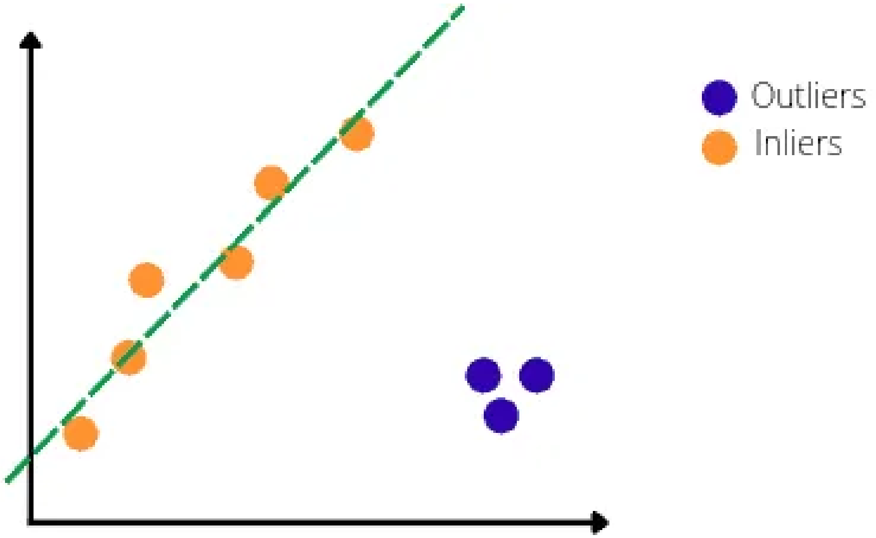
Problem: Find the best fitting line

RANSAC (example applied to linear regression)



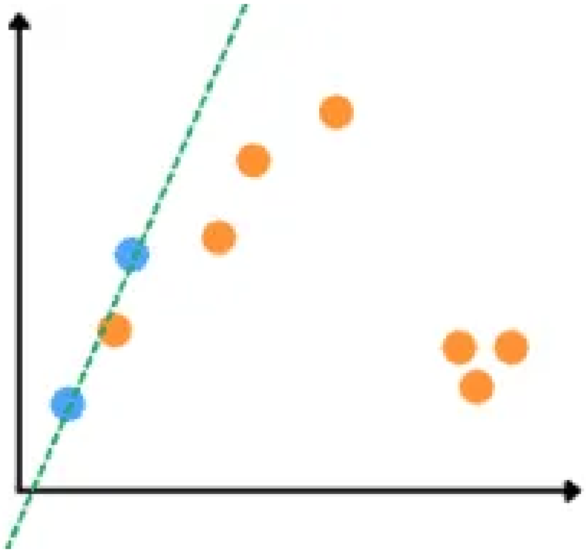
Problem: Find the best fitting line

RANSAC (example applied to linear regression)



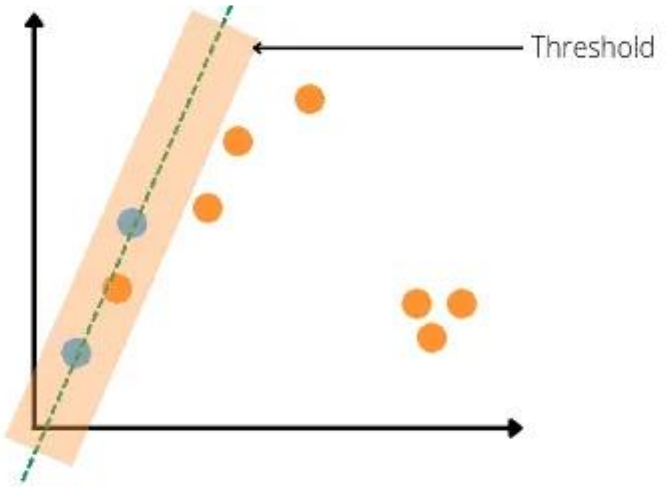
Problem: Find the best fitting line

RANSAC (example applied to linear regression)



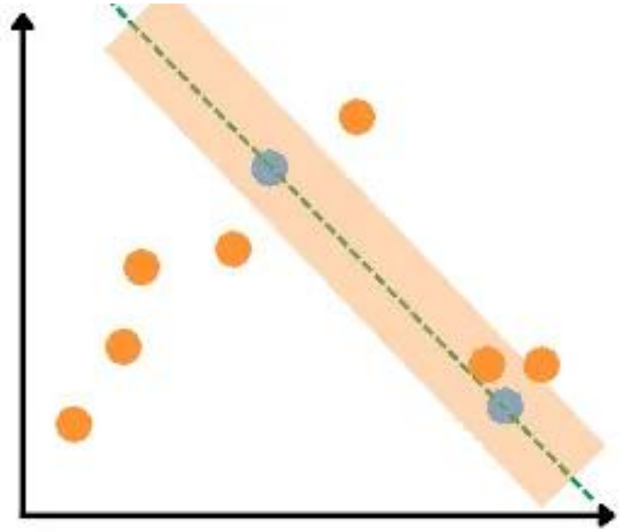
Let's pick those 2 points to make a line

RANSAC (example applied to linear regression)



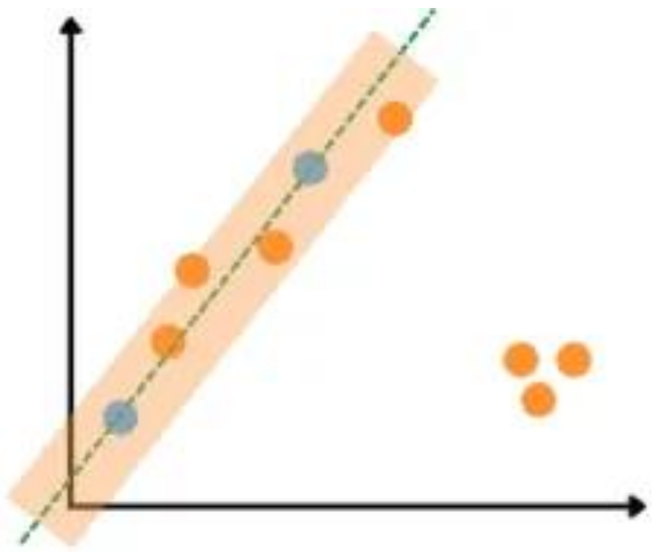
For those 2 points, 3 points are inliers

RANSAC (example applied to linear regression)



Randomly pick other 2 points, 3 points are inliers

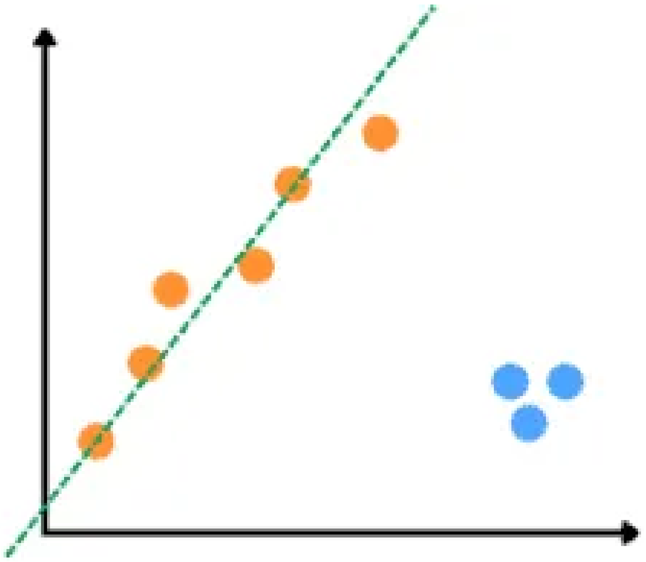
RANSAC (example applied to linear regression)



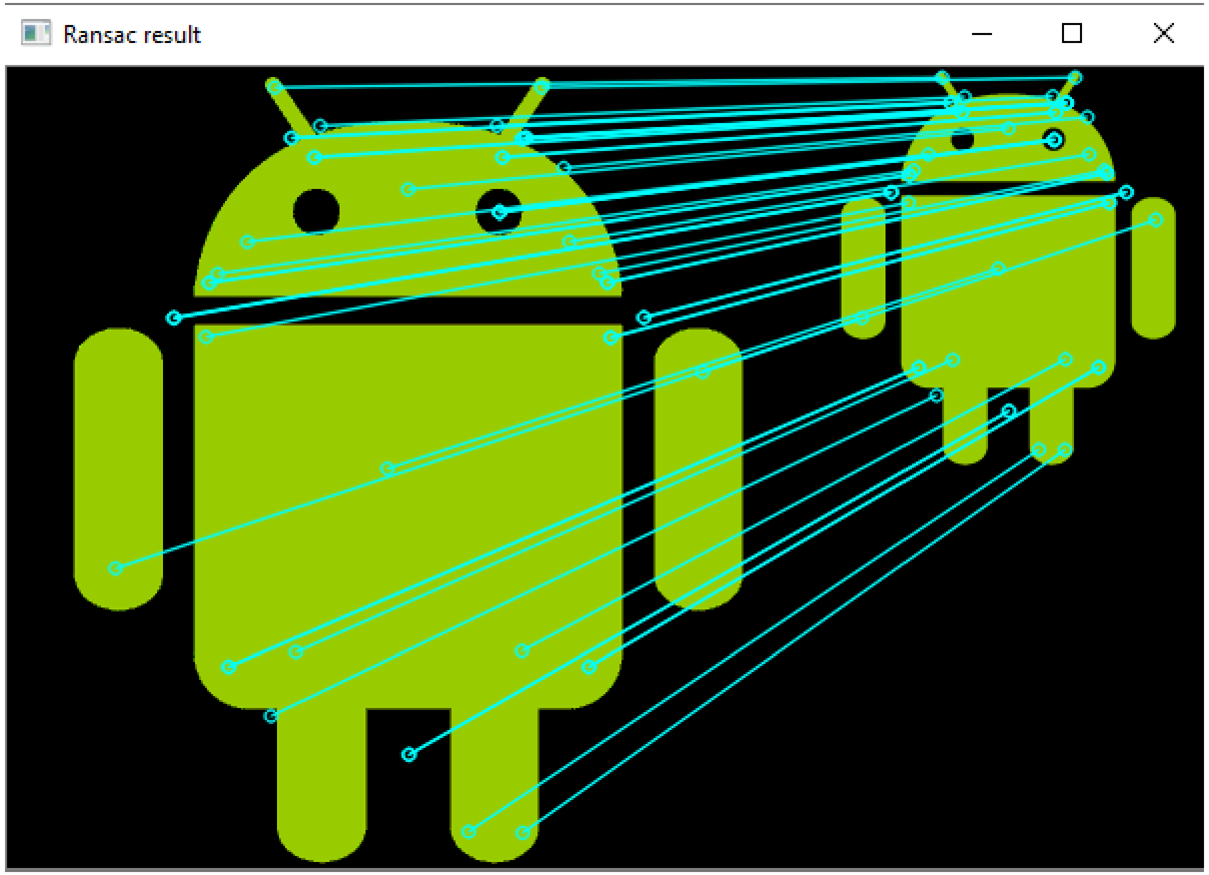
Randomly pick other 2 points, 6 points are inliers

RANSAC (example applied to linear regression)

We pick the set that have the largest set of inliers



RANSAC



Questions?