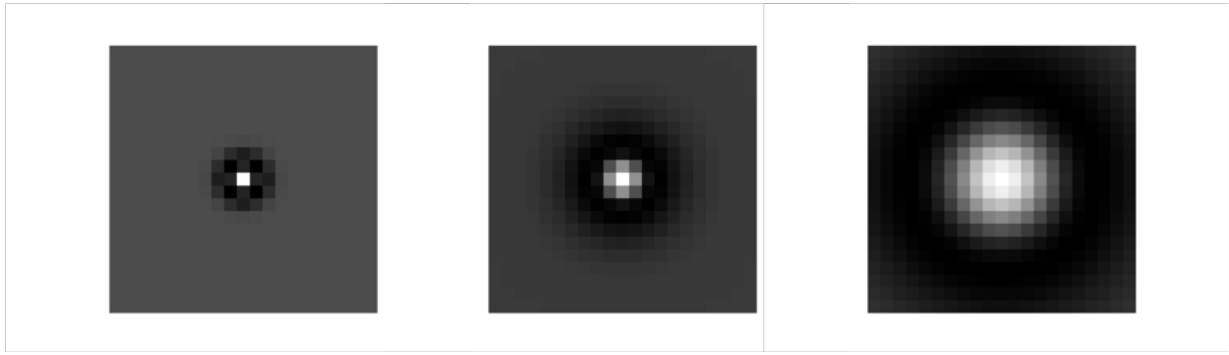
Grouping / segmentation

Texture gradient pipeline

- Step 0: Create set of filters (called *filter bank*)
  - Usually oriented edge detectors

  - And Difference of Gaussians

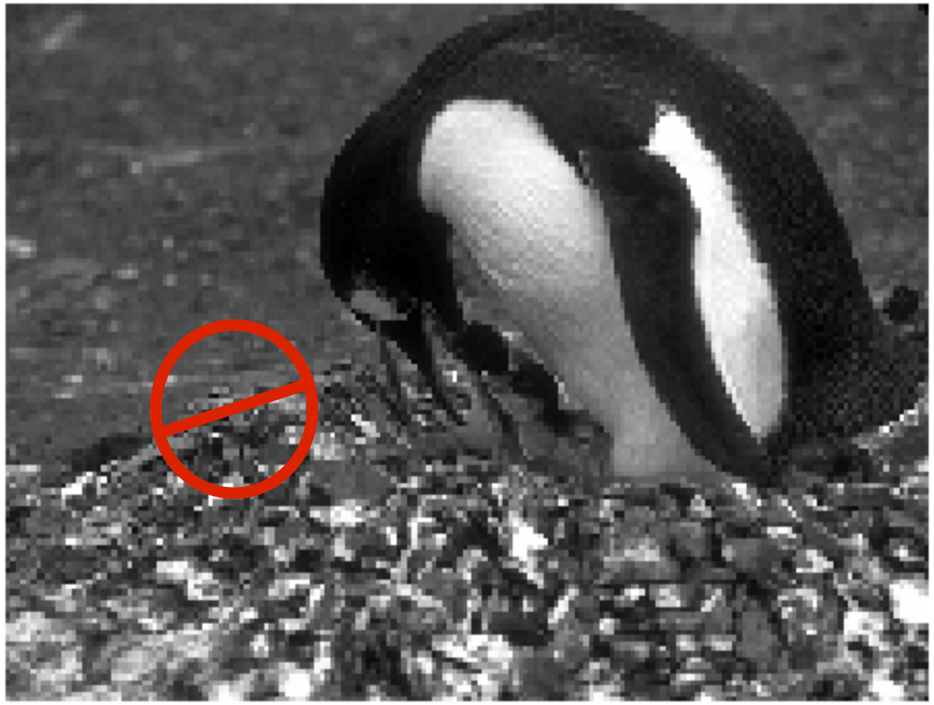


Texture gradient pipeline

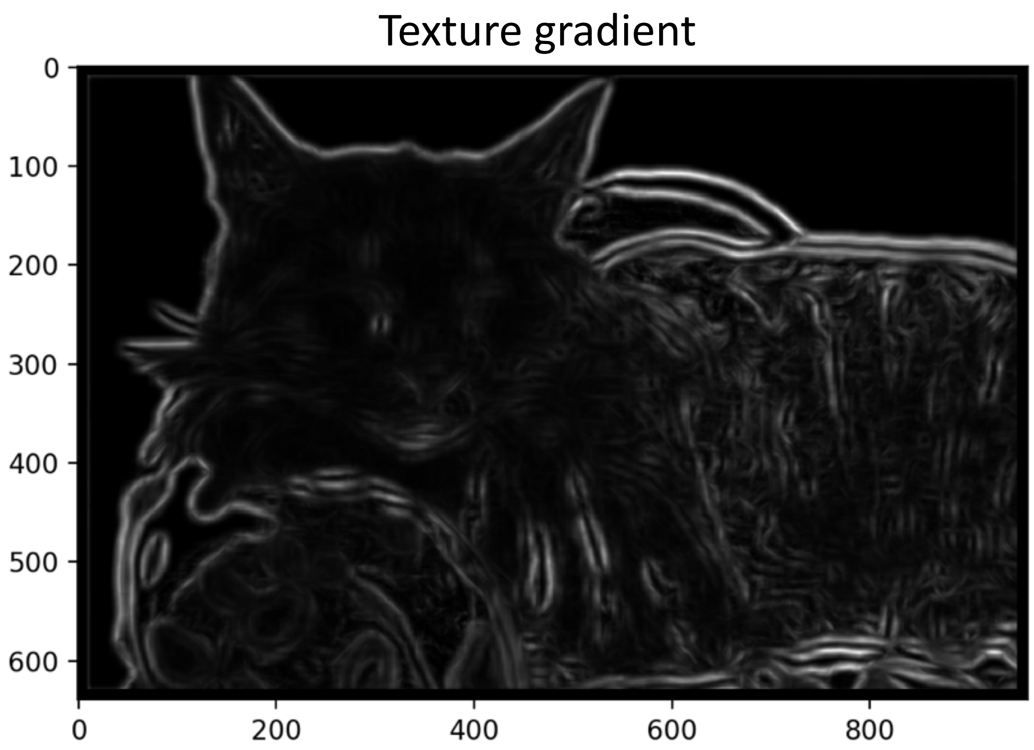
- Step 1: Convolve image with all filters in filter bank
  - If filter bank has n filters, end up with n outputs per pixel
- Step 2: Use n outputs per pixel as pixel representation to perform k-means
  - K-means centers = “textons”
- Step 3: Assign each pixel to its nearest texton
  - Nearest measured based on Euclidean distance in n-dimensional pixel space

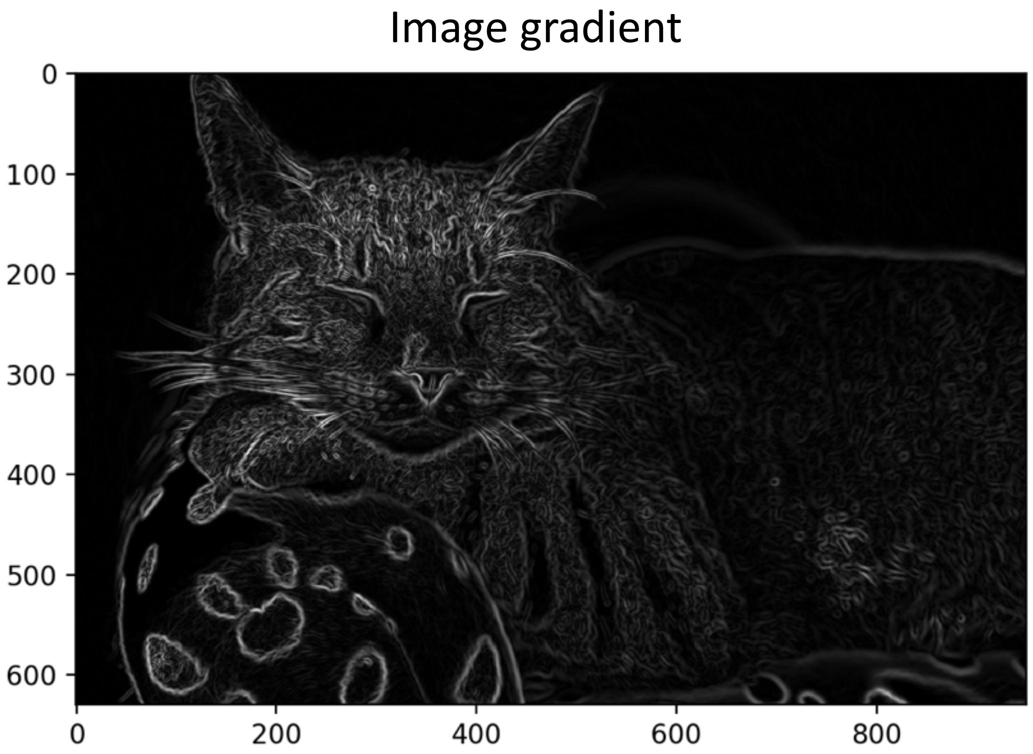
Texture gradient pipeline

- Step 5: At every pixel (x,y)
  - For every orientation θ
    - Place two half disks at that orientation
    - In each half disk, count the number of occurrences of each texton to find histogram
    - Compute the distance (e.g., L2 distance) between the two histograms
    - This gives score $T(x, y, \theta)$ for this pixel for this orientation
  - Compute the maximum score over all orientations: $T(x, y) = \max_{\theta} T(x, y, \theta)$



Texture gradient





Other techniques for grouping / segmentation

- Better contour detection
  - Learning-based edge detection (random forests, neural networks)
  - Contour completion and forming closed boundaries
- Better clustering
  - Graph-based clustering techniques (spectral clustering)
  - Clustering techniques that take contour information into account

Grouping/Segmentation: a summary

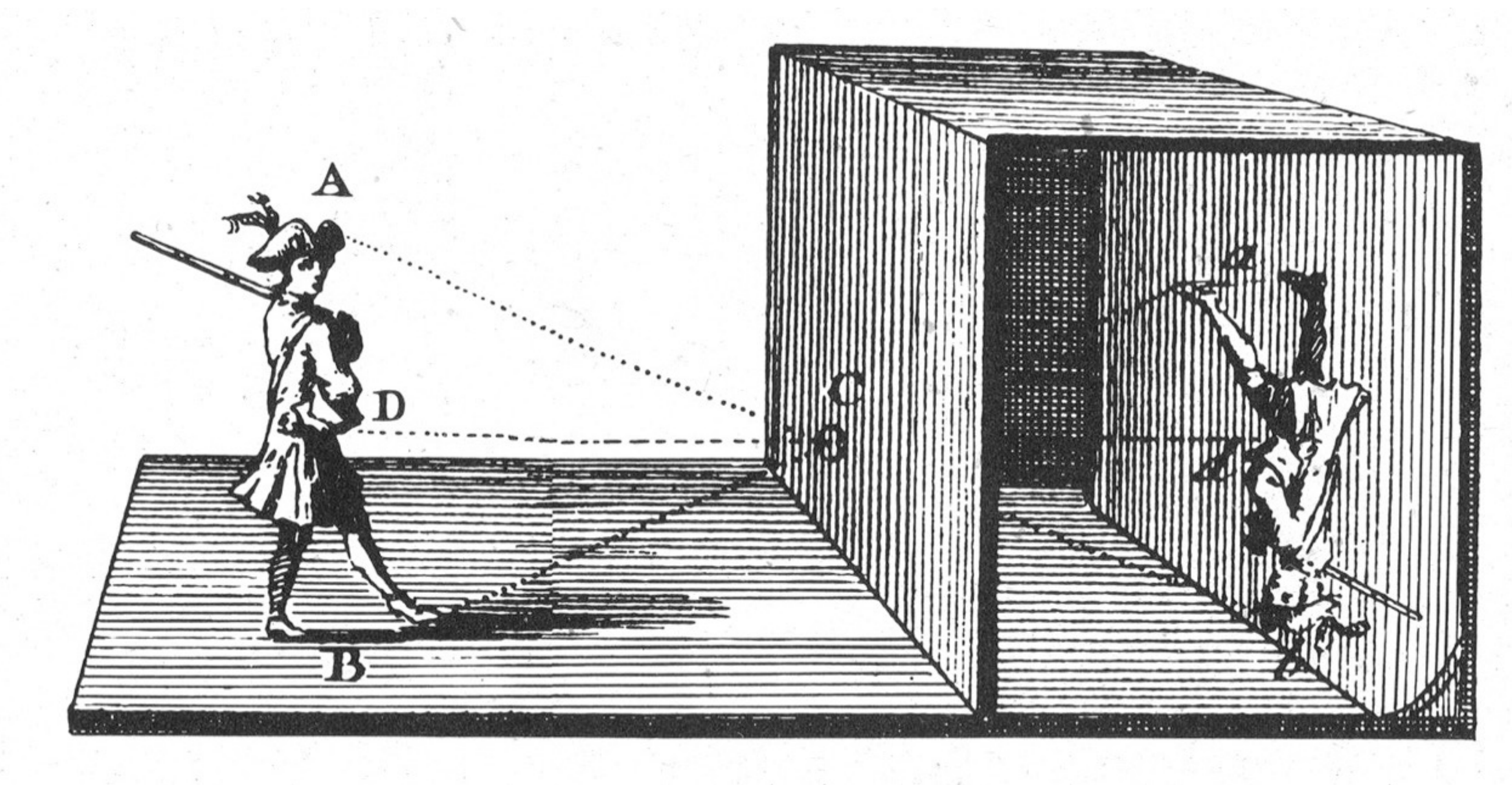
- Goal: group pixels into objects
- Simple solutions: edge detection, k-means
- Challenges:
  - Texture: Possible solution: texture gradient
  - What is k?
- Grouping still a research problem!

Reconstruction

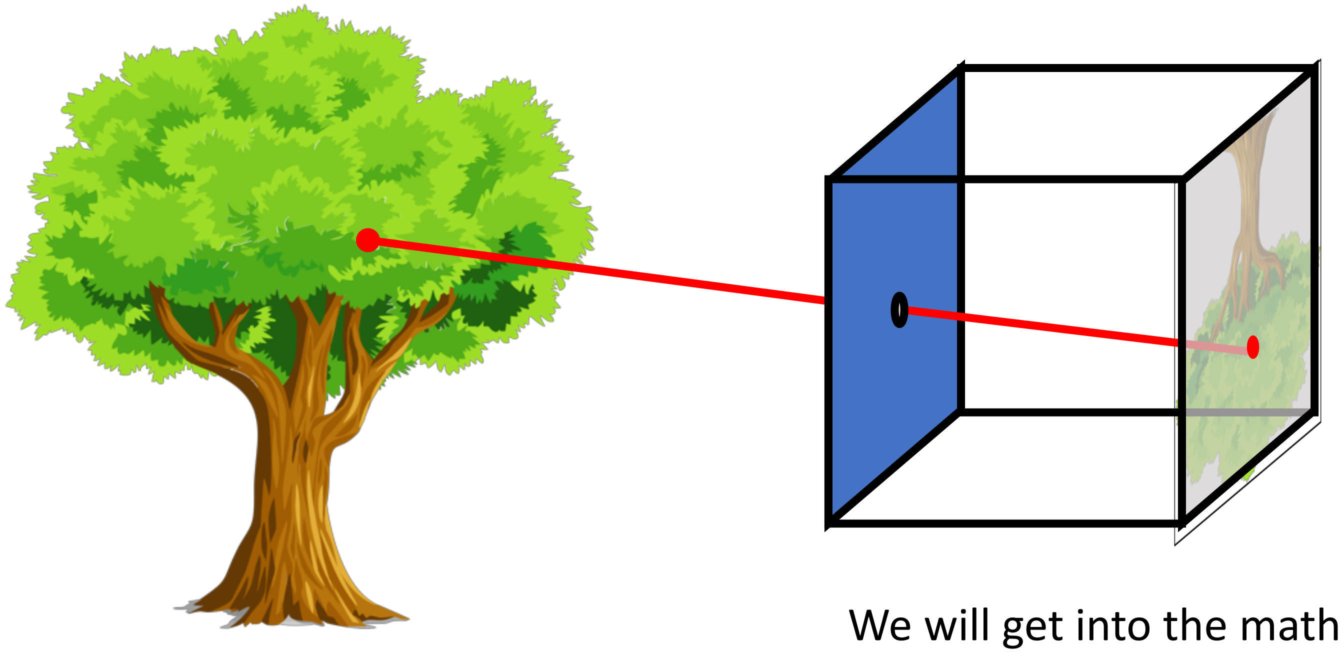
The reconstruction problem

- Camera is in 3D, taking a picture of the 3D world.
- Given an image / multiple images
  - Where is each pixel in 3D?
  - Where is the camera in 3D?
- Objects in 3D are made up of different materials, painted in different colors, illuminated under different lights
  - What is the “true color” of the object?
  - What is its “true material”?
- Need to understand the geometry and physics of image formation!

The pinhole camera - *Camera Obscura*



The pinhole camera

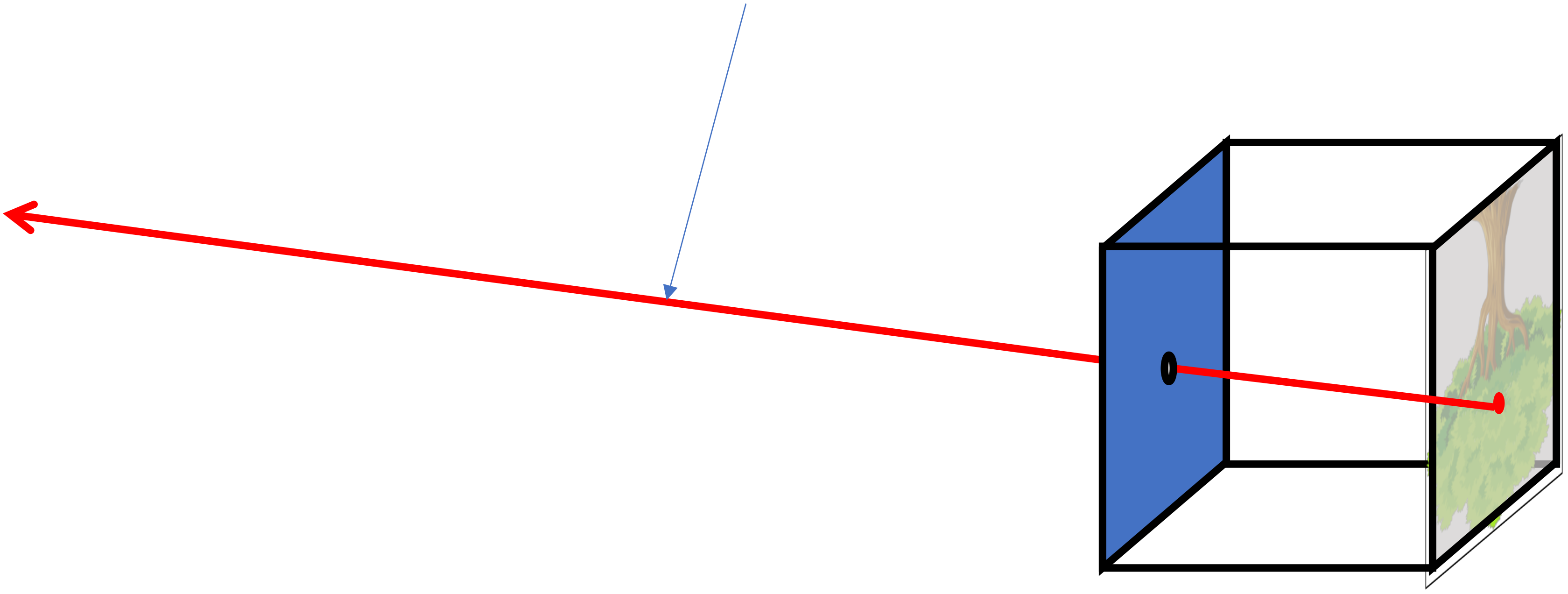


We will get into the math later

The pinhole camera

3D Reconstruction is an ill-posed problem

Actual 3D point can be
anywhere along this line



One way out: multiple images

- Multiple images can give a clue about 3D structure

One way out: multiple images

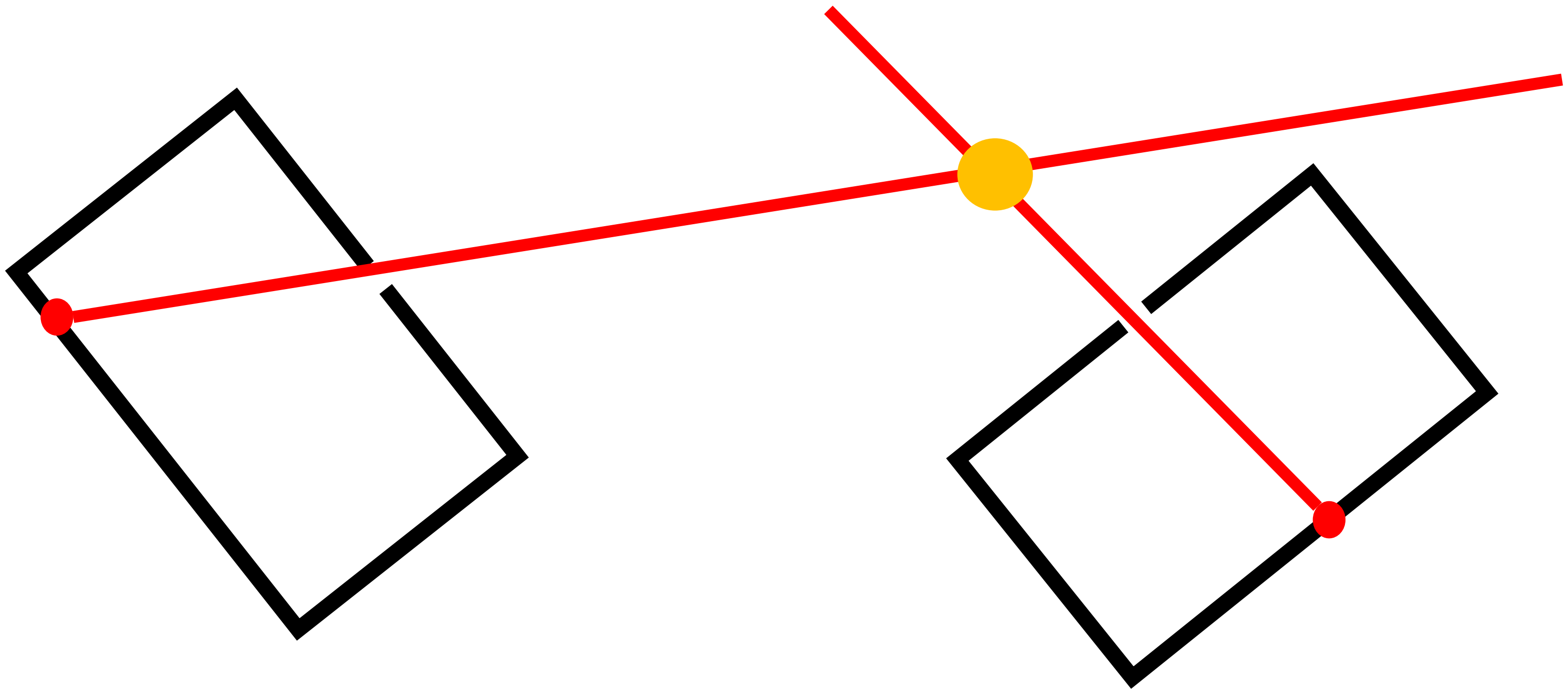
- Parallax: nearby objects move more than far away objects

One way out: multiple images

- Need to find which pixel in image 2 matches which in image 1 - the *correspondence* problem

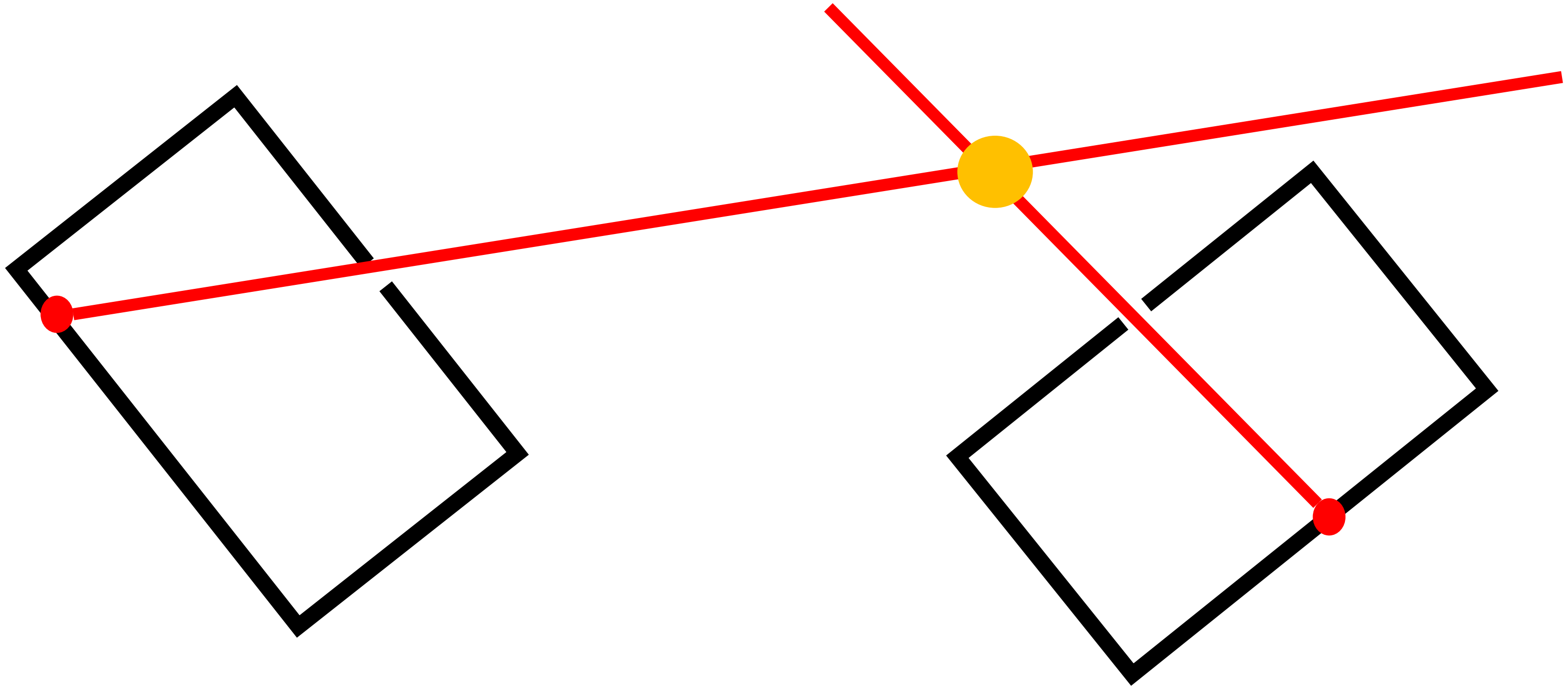
Reconstruction from correspondence

- Given known cameras, correspondence gives the location of 3D point (*Triangulation*)



Reconstruction from correspondence

- Given a 3D point, correspondence gives relationship between cameras (*Pose estimation / camera calibration*)



Next few classes

- How do we find correspondences?
- How do we use correspondences to reconstruct 3D?

Other applications of correspondence

- Image alignment
- Motion tracking
- Robot navigation

Correspondence can be challenging

Fei-Fei Li

Correspondence

by Diva Sian

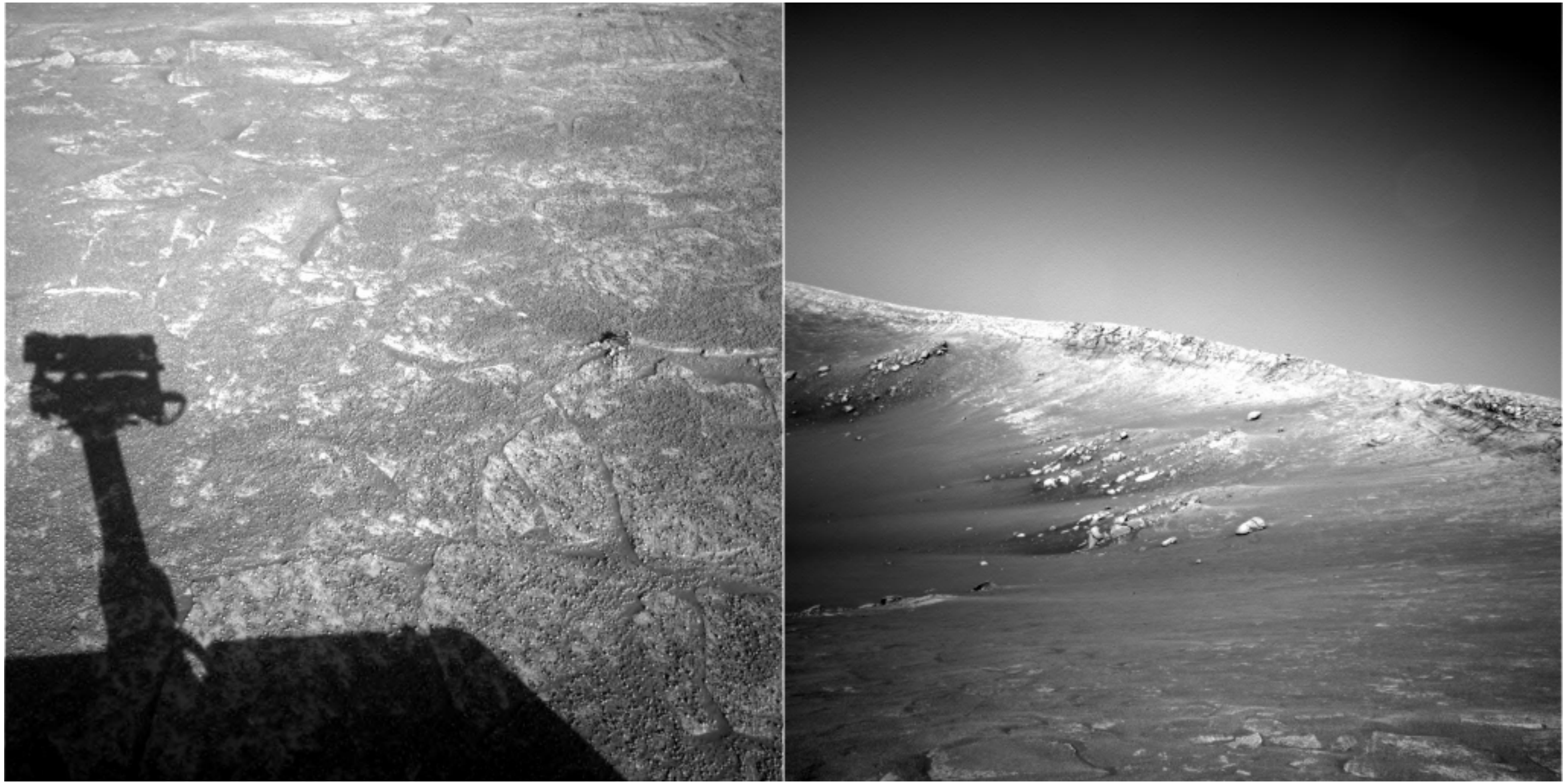
by swashford

Harder case

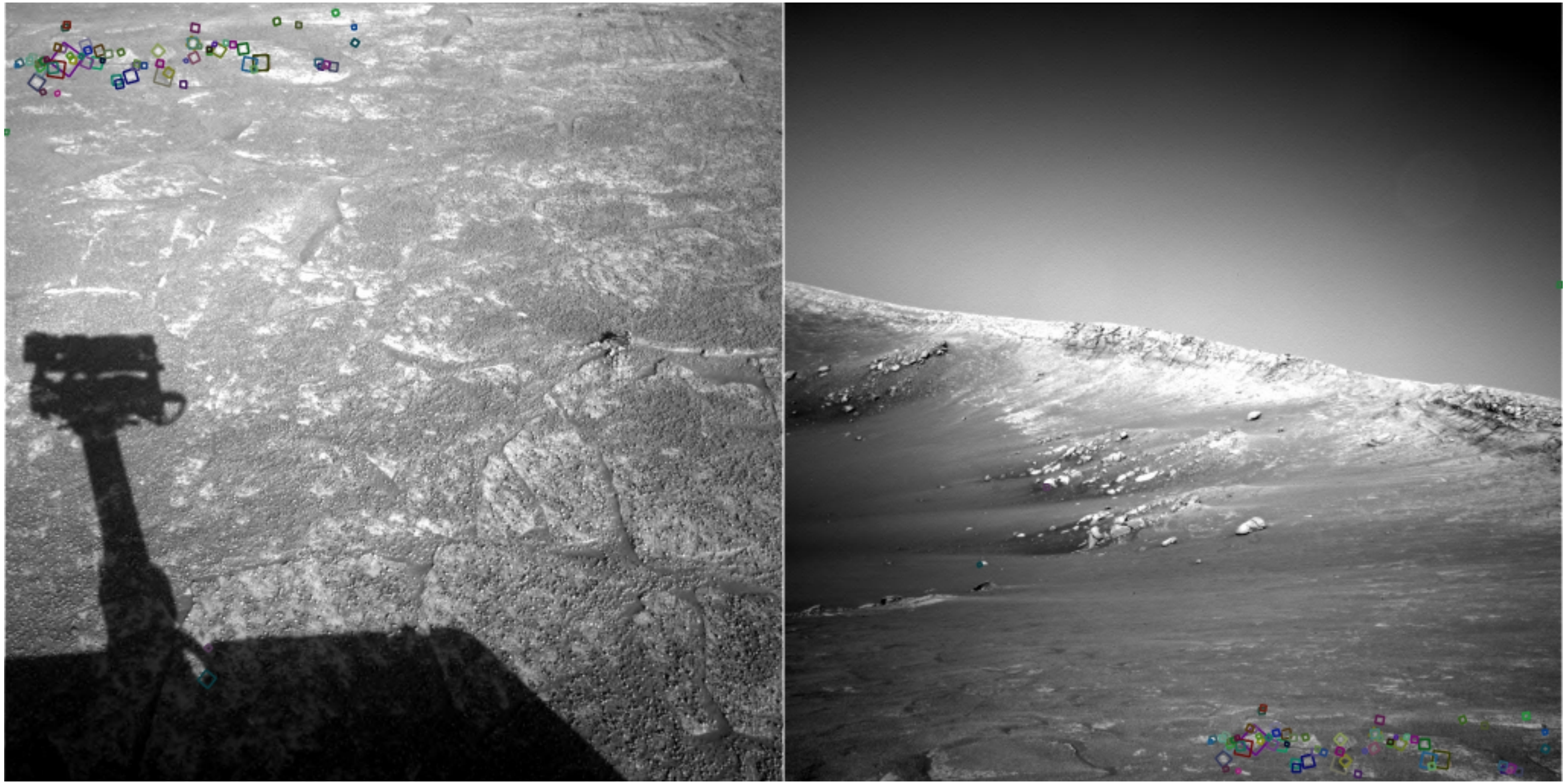
by Diva Sian

by scgbt

Harder still?



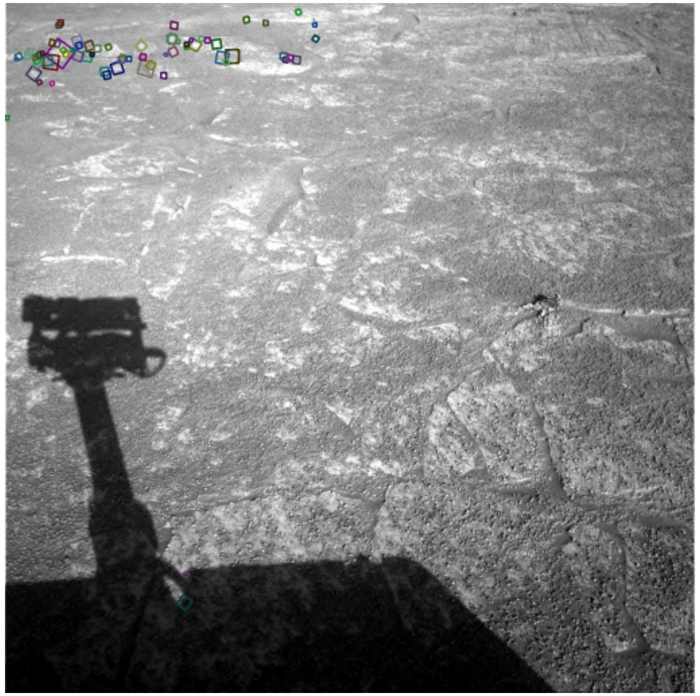
Answer below (look for tiny colored squares…)

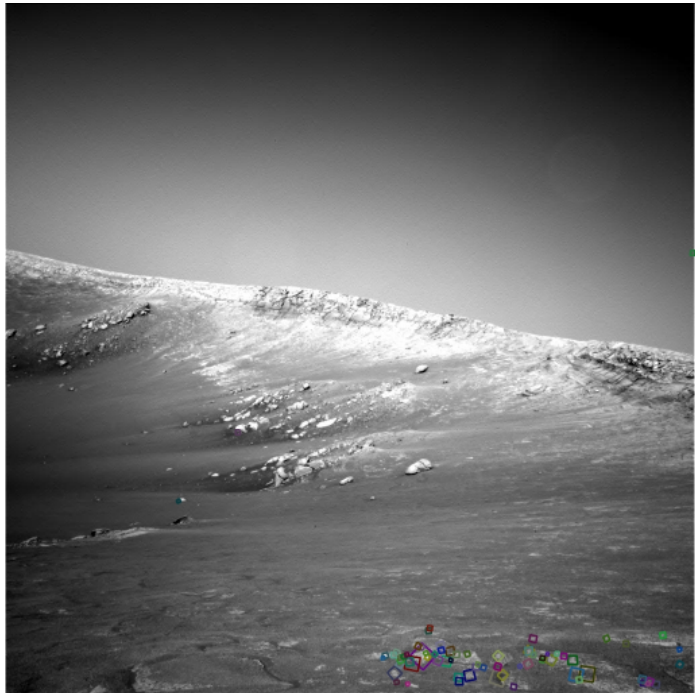


NASA Mars Rover images
with SIFT feature matches

Sparse vs dense correspondence

- Sparse correspondence: produce a few, high confidence matches
  - Good enough for estimating pose or relationship between cameras
  - Easier
- Dense correspondence: try to match every pixel
  - Needed if we want 3D location of every pixel





Sparse correspondence

- How do we do sparse correspondence?
- Step 1: In each image, separately identify a few key pixels
  - These pixels are called *Feature points / keypoints*
  - This step is called *feature detection*
- Step 2: Try to find matching pairs of keypoints in the two images
  - This step is called *feature description and matching*

What makes a good feature point?

Characteristics of good feature points

- Repeatability / invariance
  - The same feature point can be found in several images despite geometric and photometric transformations
- Saliency / distinctiveness
  - Each feature point is distinctive
  - Fewer ”false” matches

Goal: repeatability

- We want to detect (at least some of) the same points in both images.

No chance to find true matches!

- Yet we have to be able to run the detection procedure *independently* per image.

Kristen Grauman

Goal: distinctiveness

- The feature point should be distinctive enough that it is easy to match
  - Should *at least* be distinctive from other patches nearby

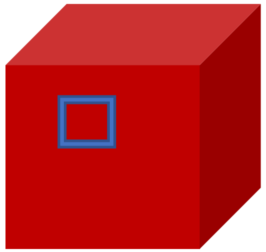
The aperture problem

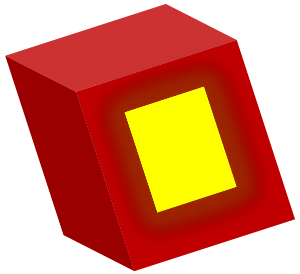
The aperture problem

- Individual pixels are ambiguous
- Idea: Look at whole patches!

The aperture problem

- Individual pixels are ambiguous
- Idea: Look at whole patches!





The aperture problem

- *Some local neighborhoods* are ambiguous

The aperture problem