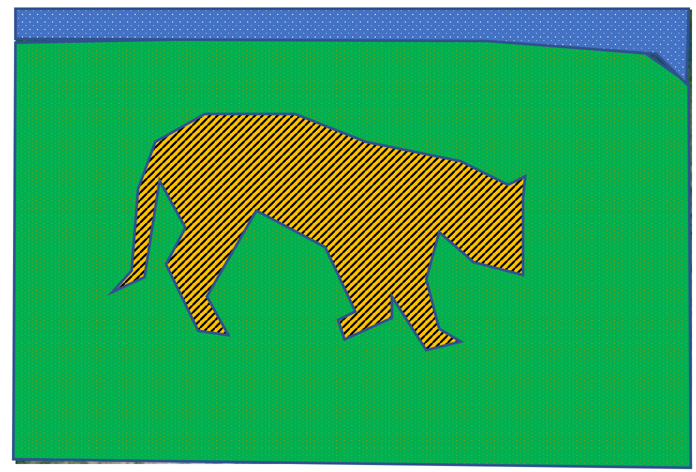
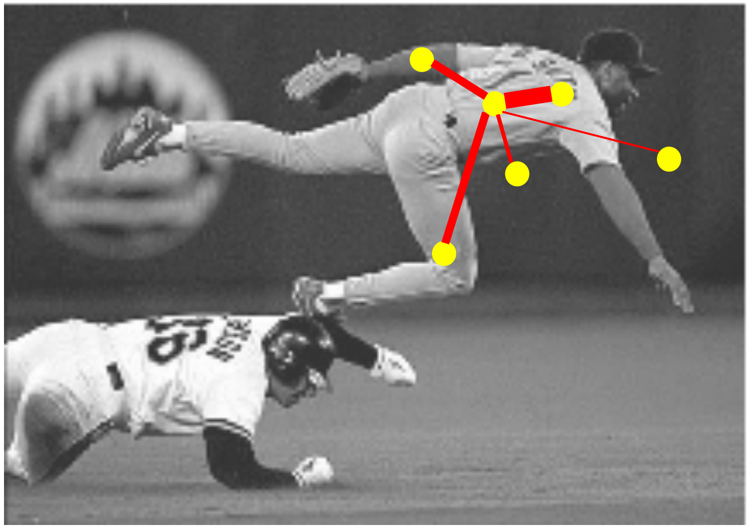
Grouping

What is grouping?

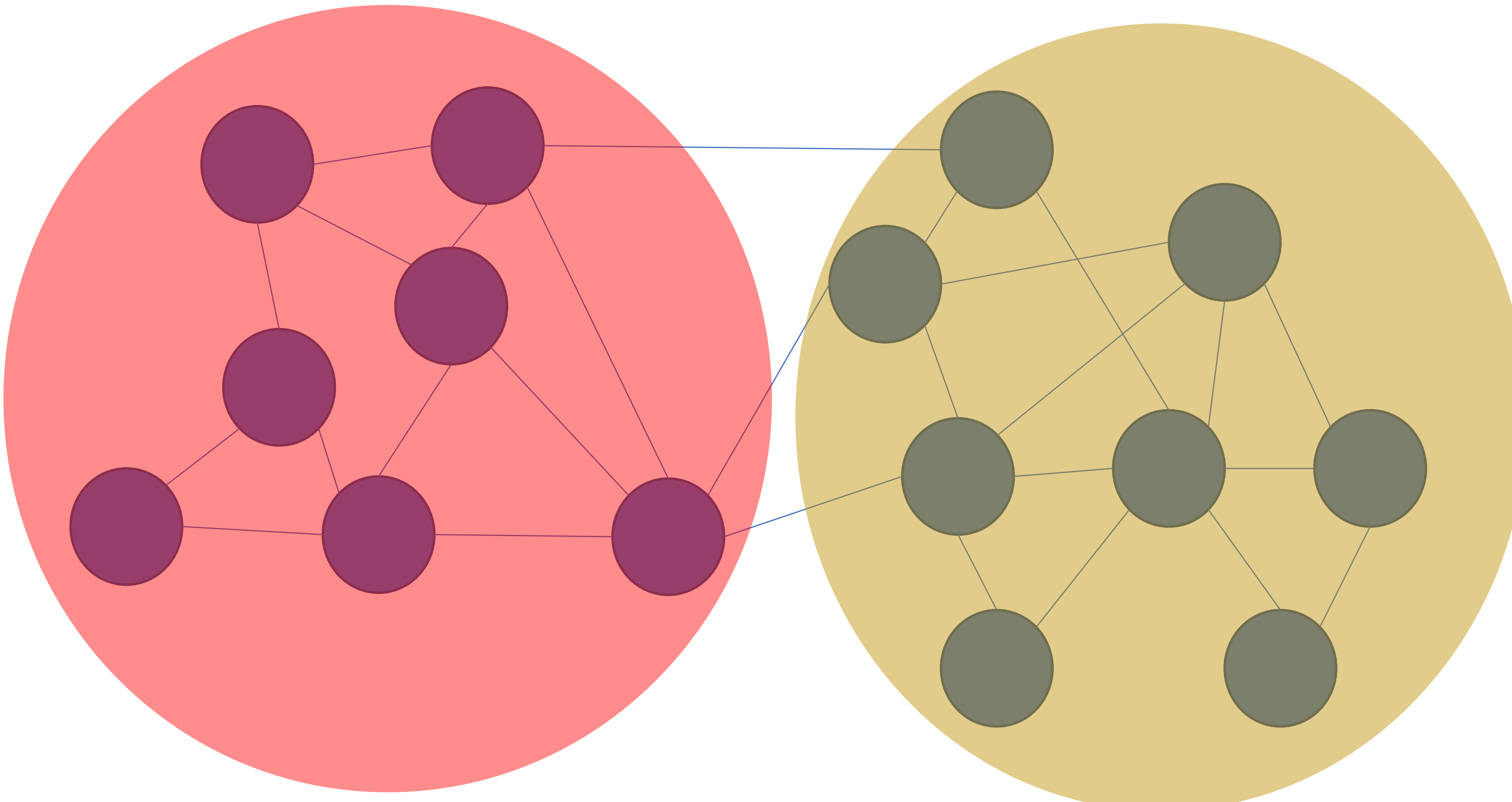


Images as graphs

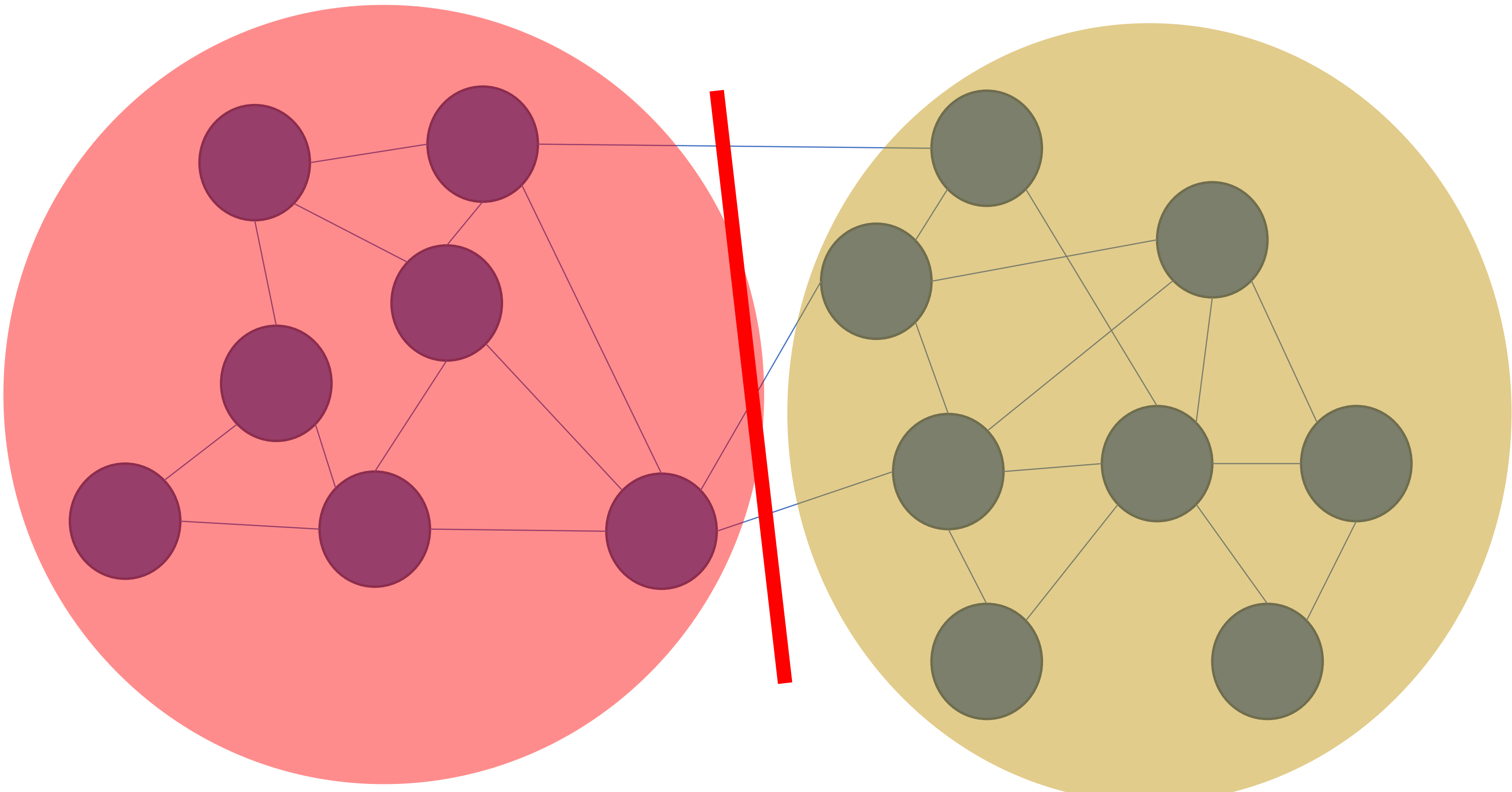
- Each pixel is node
- Edge between "similar pixels"
  - *Proximity:* nearby pixels are more similar
  - *Similarity:* pixels with similar color are more similar
- Weight of edge = similarity



Segmentation is graph partitioning

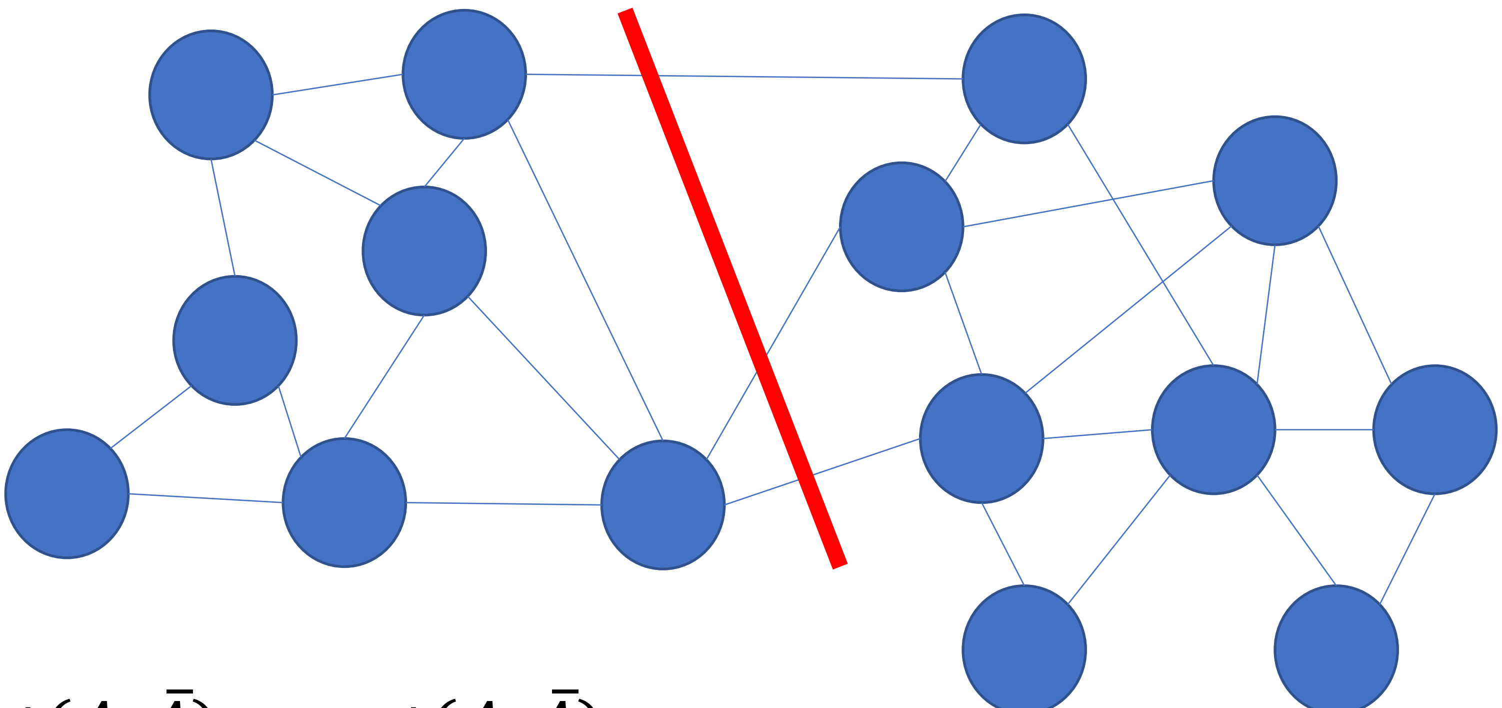


Segmentation is graph partitioning



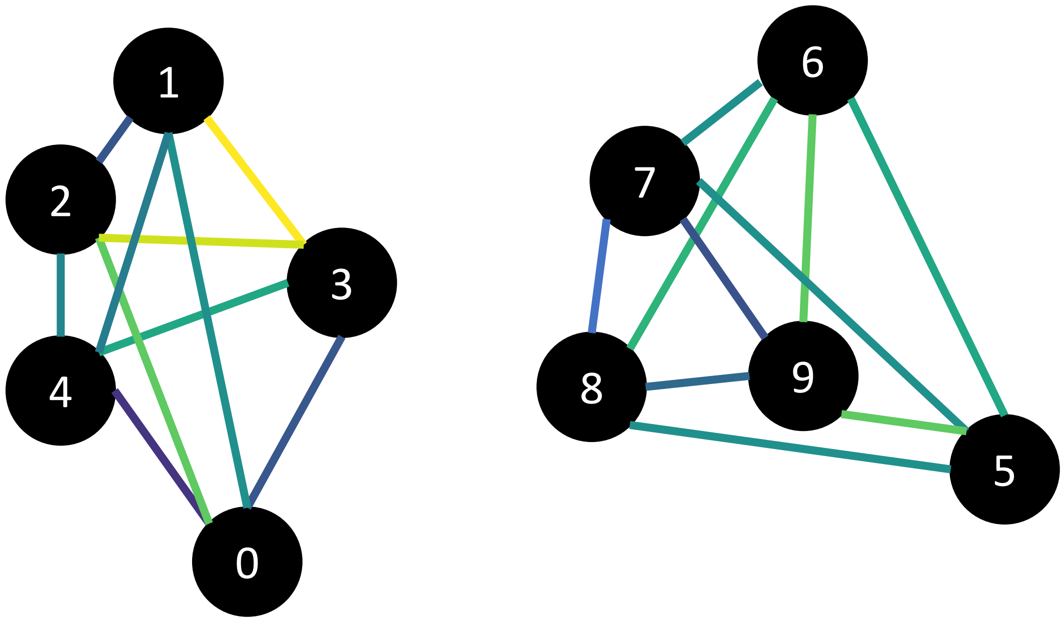
- Every partition “cuts” some edges

Normalized cut

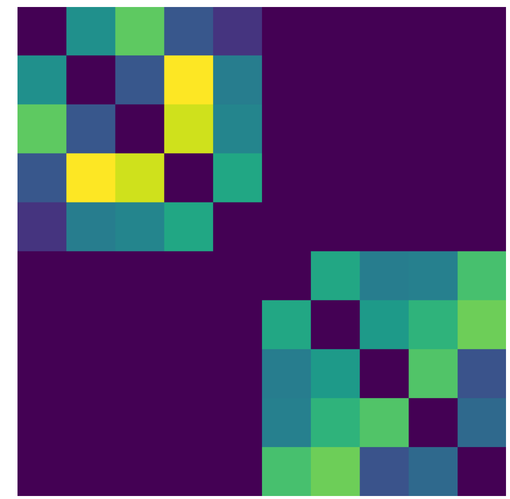


$$\frac{cut(A,\bar{A})}{vol(A)} + \frac{cut(A,\bar{A})}{vol(\bar{A})}$$

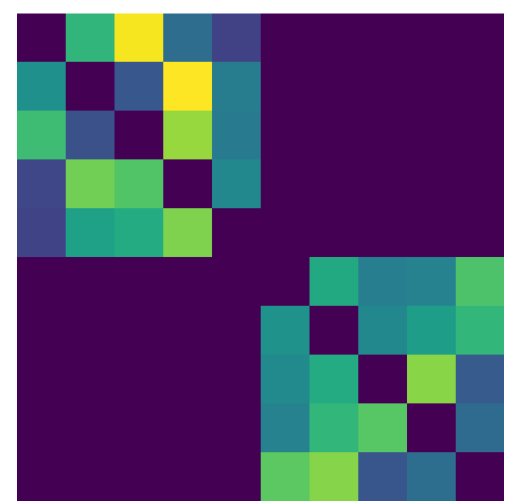
Graphs and matrices



$$E_{ij} = \frac{w_{ij}}{\sum_k w_{ik}}$$

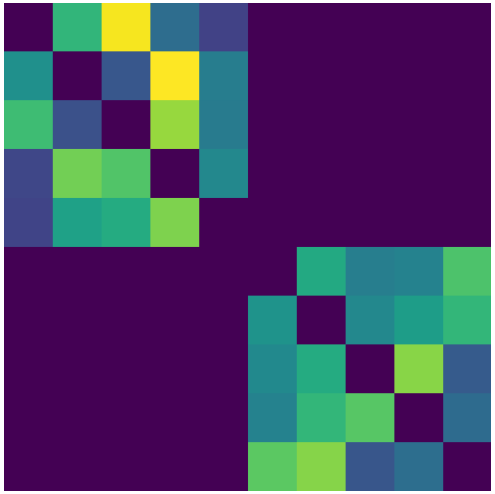


W



E = $D^{-1}W$

Graphs and matrices

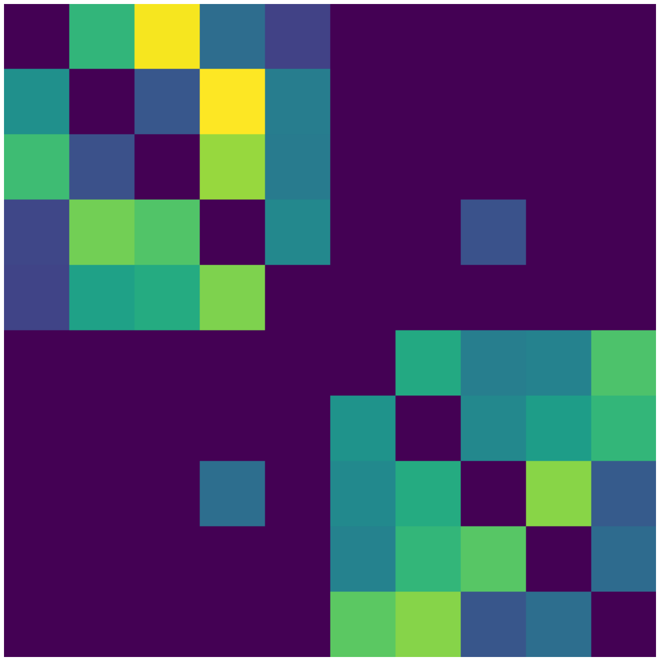


E = D^{-1}W

$$E_{ij} = \frac{w_{ij}}{\sum_k w_{ik}}$$

	v_1	**Ev_1**
0:	1	1
1:	1	1
2:	1	1
3:	1	1
4:	1	1
5:	0	0
6:	0	0
7:	0	0
8:	0	0
9:	0	0

Graphs and matrices



$E = D^{-1}W$

$$E_{ij} = \frac{w_{ij}}{\sum_k w_{ik}}$$

	v_1	$\mathbf{Ev_1}$
0:	1	1
1:	1	1
2:	1	1
3:	1	1
4:	1	1
5:	0	0
6:	0	0
7:	0	0.2
8:	0	0
9:	0	0

Graphs and matrices

$$D^{-1}Wy \approx y$$

Define z so that $y = D^{-\frac{1}{2}}z$

$$D^{-1}WD^{-\frac{1}{2}}z \approx D^{-\frac{1}{2}}z$$

$$\Rightarrow D^{-\frac{1}{2}}WD^{-\frac{1}{2}}z \approx z$$

$$\Rightarrow (I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}})z \approx 0$$

Graphs and matrices

$$\Rightarrow (I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}) z \approx 0$$

$$\Rightarrow \mathcal{L} z \approx 0$$

$$\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

is called the
Normalized Graph
Laplacian

Graphs and matrices

$$\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

- We want $\mathcal{L}z \approx 0$
- Trivial solution: all nodes of graph in one cluster, nothing in the other
- To avoid trivial solution, look for the *eigenvector with the second smallest eigenvalue*

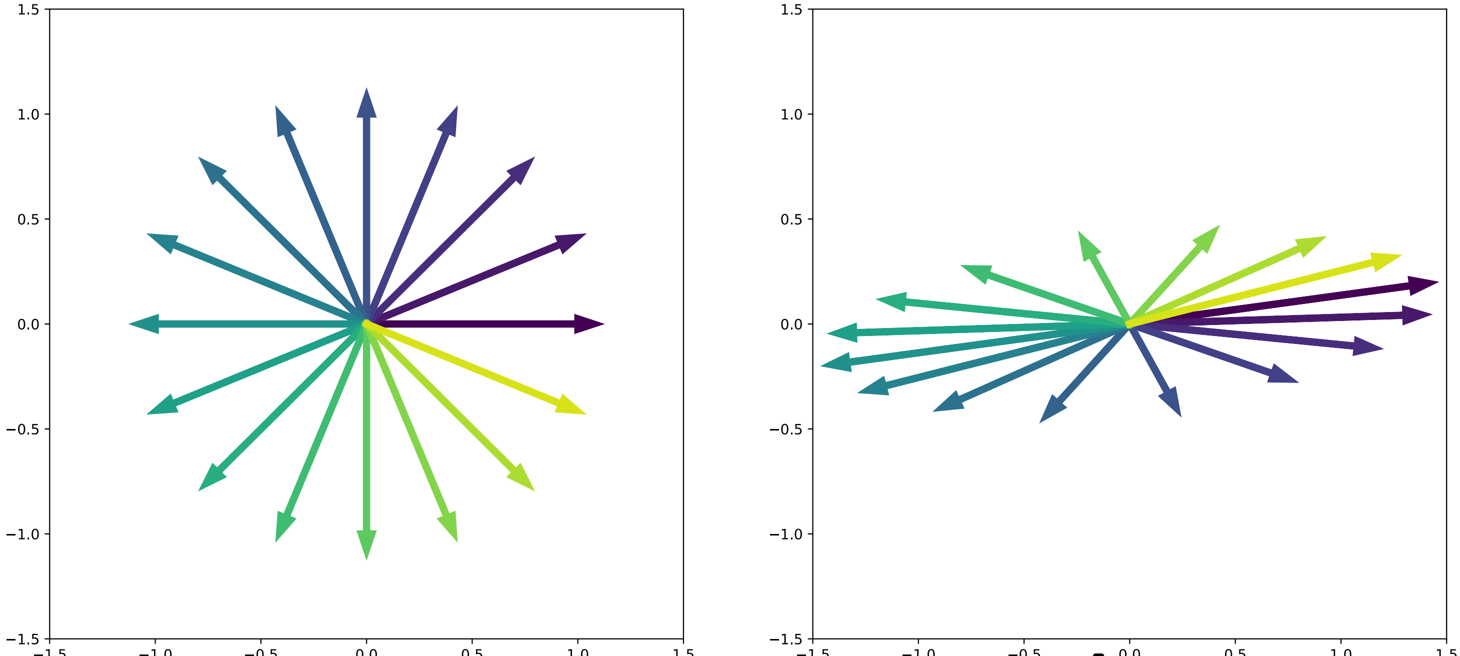
$$\mathcal{L}z = \lambda z$$

$$\lambda_1 < \lambda_2 < \ldots < \lambda_N$$

- Find z s.t. $\mathcal{L}z = \lambda_2 z$

A quick detour into eigenvalues and eigenvectors

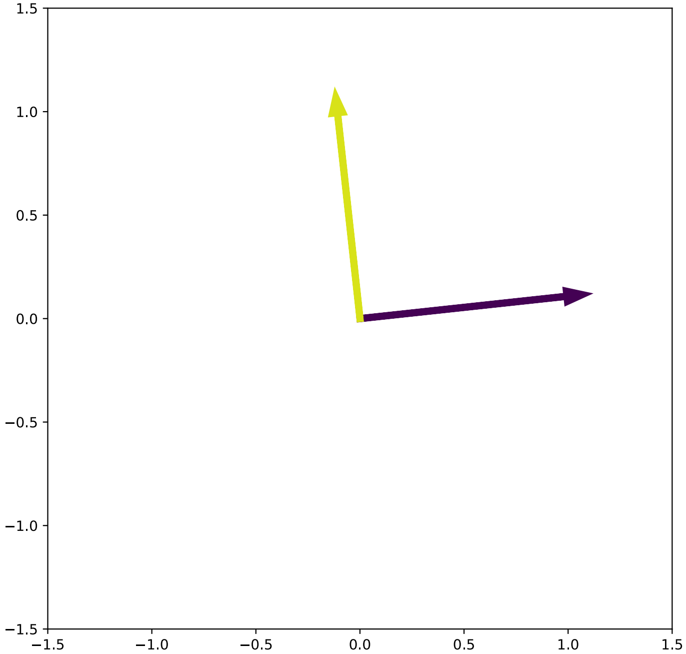
- Given vector x and matrix A, what does Ax look like?



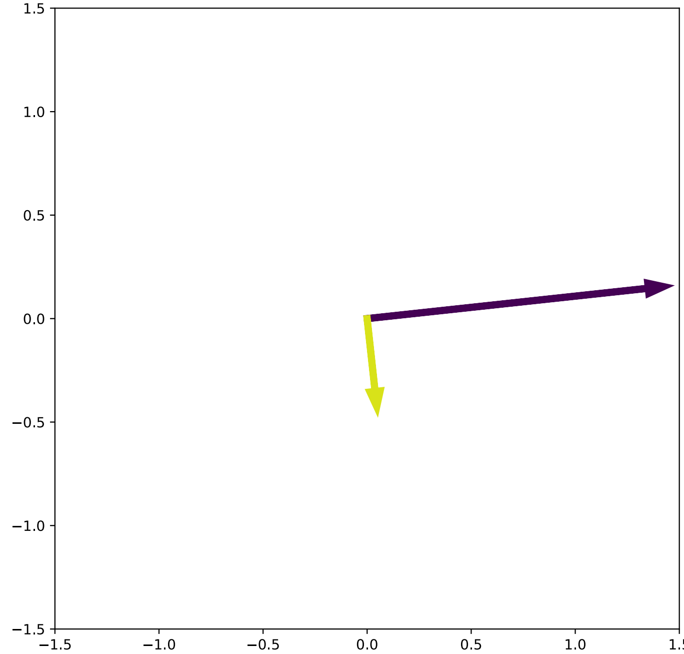
x Ax

A quick detour into eigenvalues and eigenvectors

- Given a matrix A, x is an eigenvector with eigenvalue λ if $Ax = \lambda x$



x



Ax

A quick detour into eigenvalues and eigenvectors

- Given a matrix A, x is an eigenvector with eigenvalue λ if $Ax = \lambda x$
- Any square real symmetric n x n matrix has n eigenvalues
- For symmetric mats, eigenvectors corresponding to two different eigenvalues are orthogonal

$$\lambda_1 = \min_x \frac{\|Ax\|}{\|x\|}$$

$$\lambda_2 = \min_x \frac{\|Ax\|}{\|x\|} \text{ s.t. } x^T v_1 = 0$$

Eigenvectors and the graph laplacian

- We want $\mathcal{L}z \approx 0$
- Trivial solution z_0: all nodes of graph in one cluster, nothing in the other
  - $\mathcal{L}z_0 = 0$
  - z0 is eigenvector with 0 eigenvalue
- We want $z^T z_0 = 0$ and $\mathcal{L}z \approx 0$
- Look for eigenvector with *second-smallest* eigenvalue

Normalized cuts

- Approximate solution to normalized cuts
- Construct matrix W and D
- Construct normalized graph laplacian

$$\mathcal{L} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

- Look for the second smallest eigenvector

$$\mathcal{L}z = \lambda_2 z$$

- Compute $y = D^{-\frac{1}{2}} z$
- *Threshold y to get clusters*
  - Ideally, sweep threshold to get lowest N-cut value

More than 2 clusters

- Given graph, use N-cuts to get 2 clusters
- Each cluster is a sub-graph
  - Re-run N-cuts on each sub-graph

Normalized cuts

- NP Hard
- But approximation using *eigenvector of normalized graph laplacian*
    - Smallest eigenvector : trivial solution
    - *Second smallest eigenvector: good partition*
    - *Other eigenvectors: other partitions*

- An instance of “Spectral clustering”
    - Spectrum = set of eigenvalues
    - Spectral clustering = clustering using eigenvectors of (various versions of) graph laplacian

Images as graphs

- Each pixel is a node
- What is the edge weight between two nodes / pixels?
  - F(i): intensity / color of pixel i
  - X(i): position of pixel i

$$w_{ij} = e^{\frac{-\|\boldsymbol{F}(i)-\boldsymbol{F}(j)\|_2^2}{\sigma_I}} * \begin{cases} e^{\frac{-\|\boldsymbol{X}(i)-\boldsymbol{X}(j)\|_2^2}{\sigma_X}} & \text{if } \|\boldsymbol{X}(i)-\boldsymbol{X}(j)\|_2 < r \\ 0 & \text{otherwise,} \end{cases}$$

Computational complexity

- A 100 x 100 image has 10K pixels
- A graph with 10K pixels has a 10K x 10K affinity matrix
- Eigenvalue computation of an N x N matrix is $O(N^3)$
- Very very expensive!

Eigenvectors of images

The eigenvector has as many components as pixels in the image

2^{nd} Eigenvector

2^{nd} Eigenvector

Recursive N-cuts

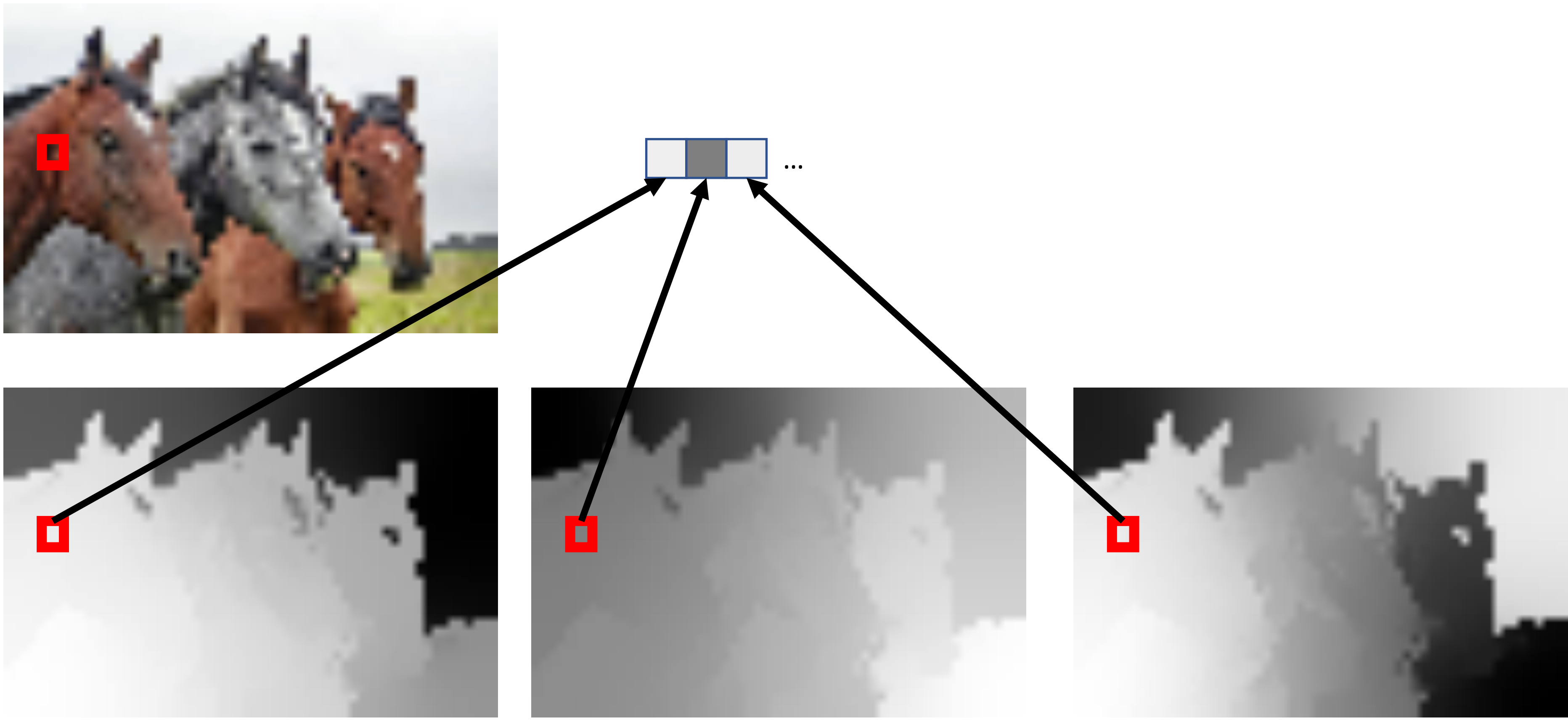
2^{nd} eigenvector

First partition

2^{nd} eigenvector of 1^{st} subgraph

recursive partition

Eigenvectors as pixel representations



2^{nd} eigenvector

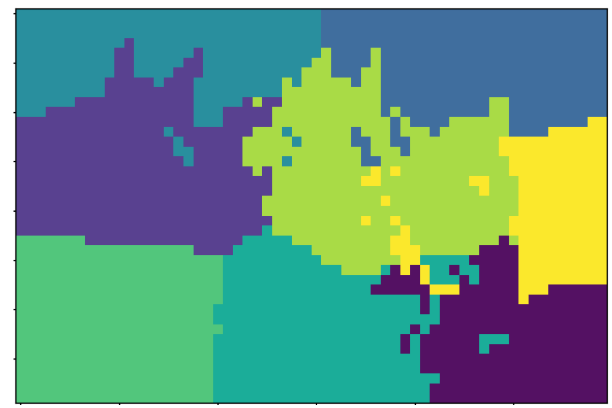
3^{rd} eigenvector

4^{th} eigenvector

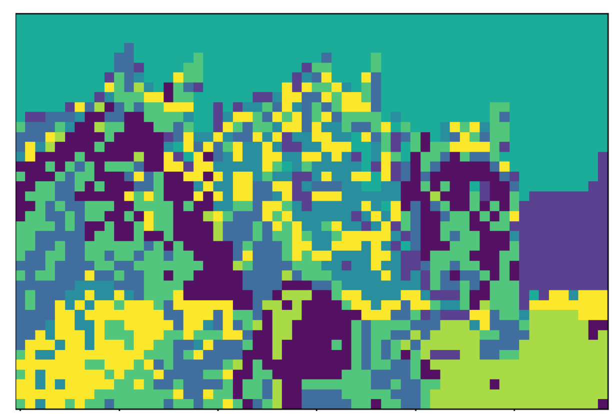
K-means

K-Means: Pixel represented using top 10 eigenvectors

K-Means: RGB + X,Y

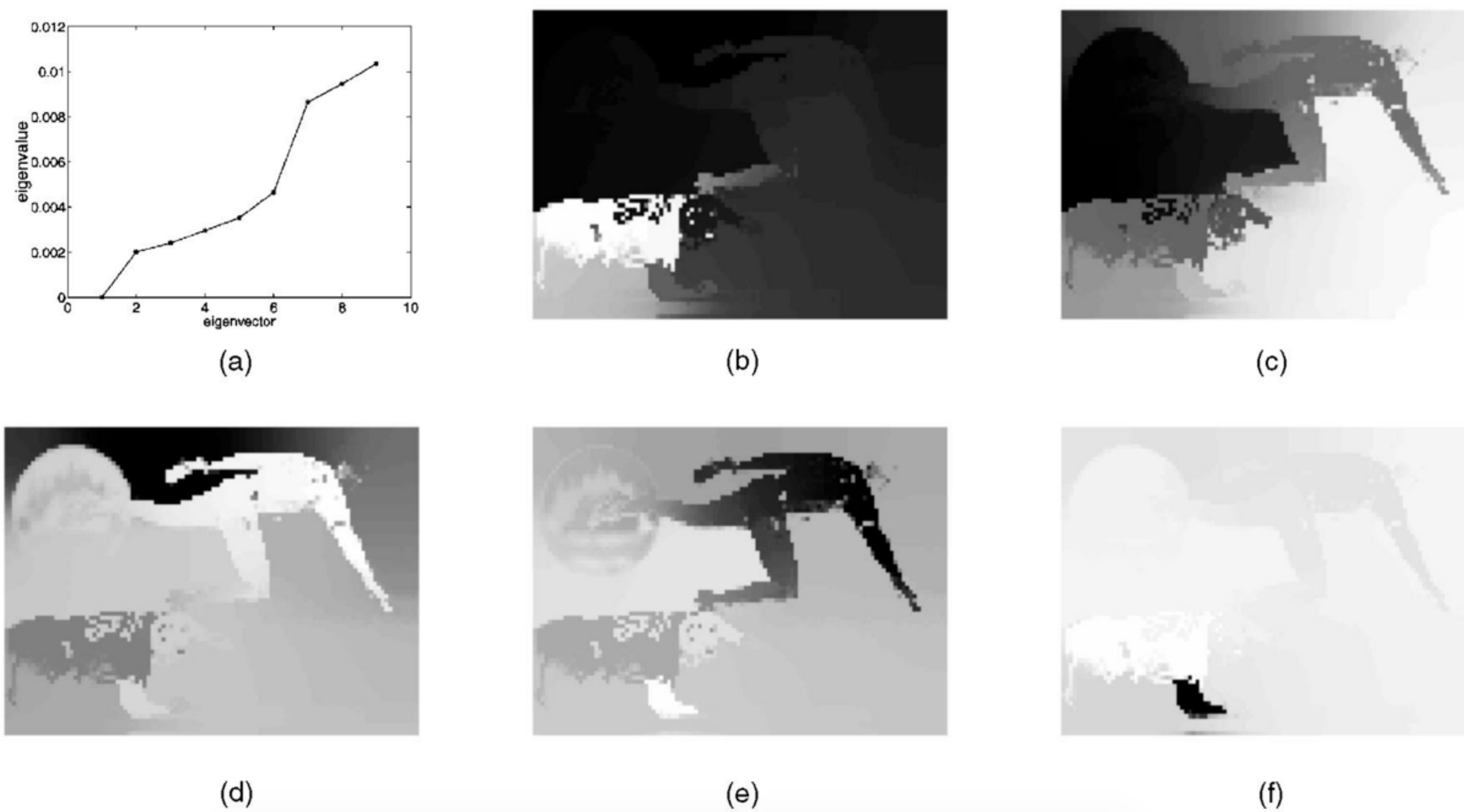


K-Means: RGB



Another example

Eigenvectors of images

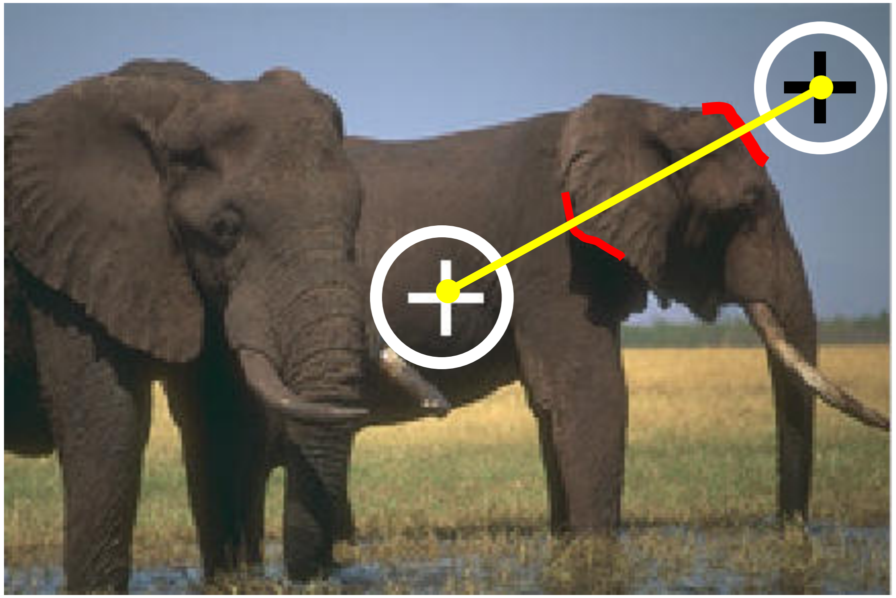


N-Cuts resources

- http://scikit-learn.org/stable/modules/clustering.html#spectral-clustering
- https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf

Images as graphs

- Enhancement: edge between far away pixel, weight = 1 – magnitude of *intervening contour*



Eigenvectors of images

Grouping: a summary

- Goal: group pixels into objects
- Simple baselines based on color similarity and local reasoning: Canny, k-means
- Complex solution to exploit contour continuity and global reasoning: N-Cuts
- Challenges:
  - Texture
  - What is k?
- Grouping still a research problem!

The correspondence problem

Why?

- Multiple images can give a clue about 3D structure

Why? Reconstruction

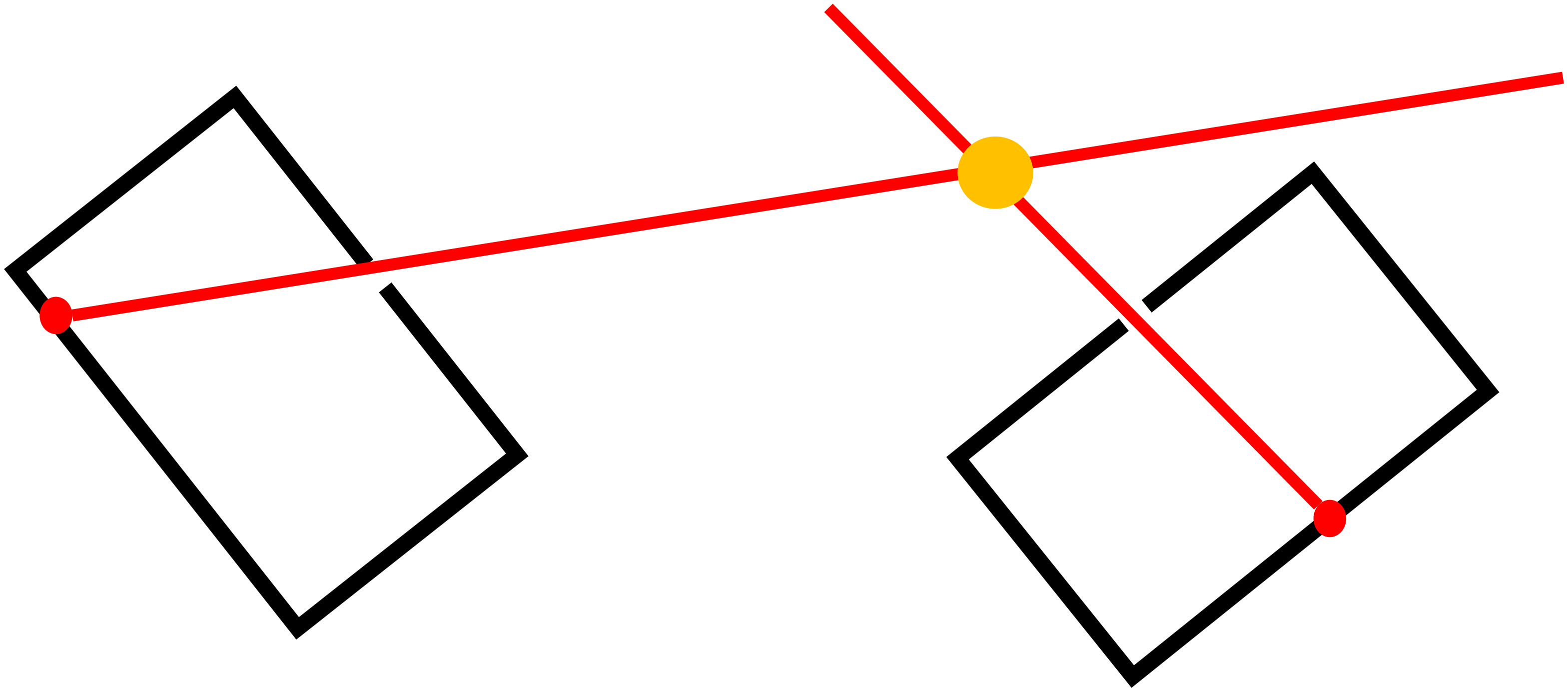
- Multiple images can give a clue about 3D structure

Why? Reconstruction

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

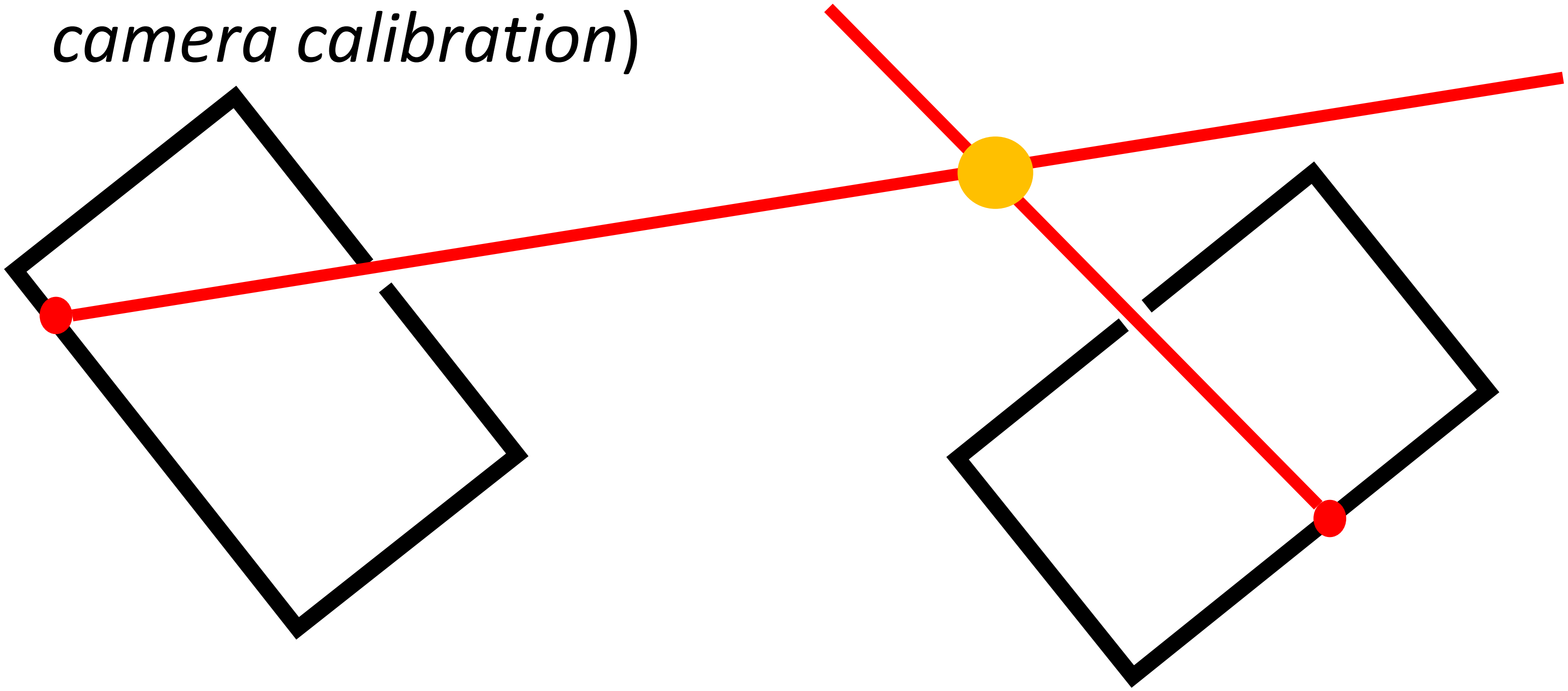
Reconstruction from correspondence

- 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*)

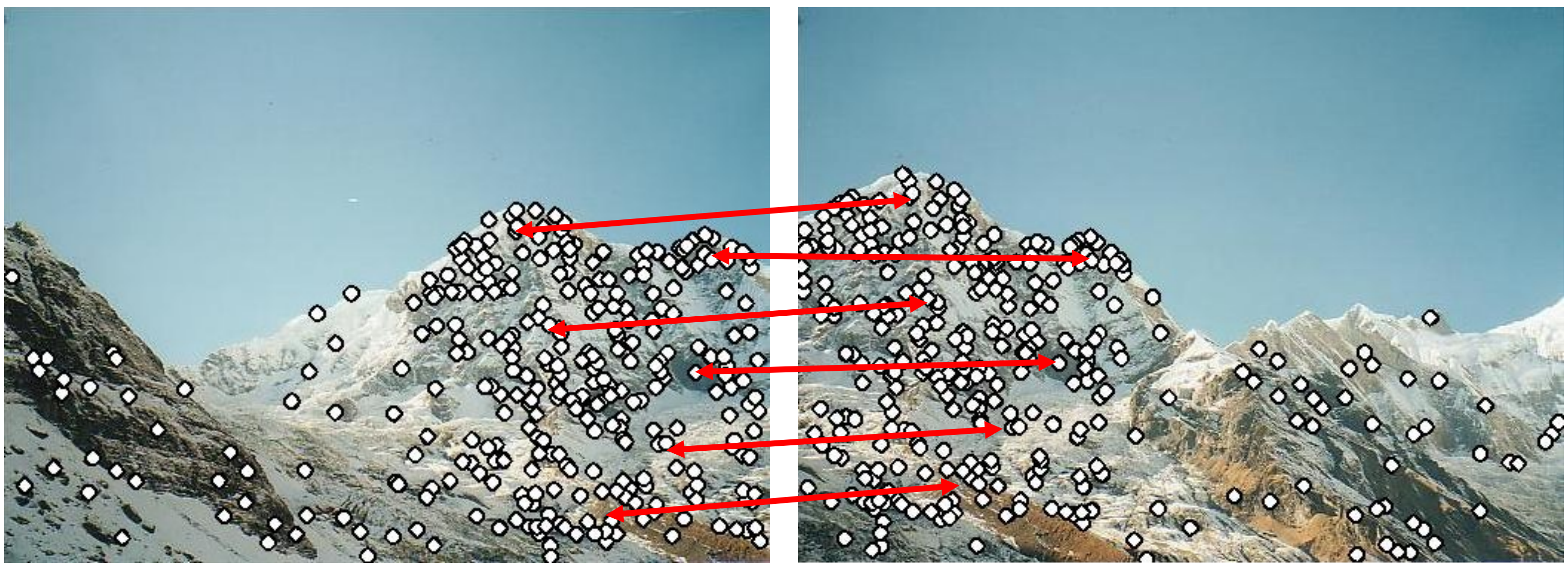


Pose-estimation / Camera calibration

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Pose-estimation / Camera calibration

- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract correspondence

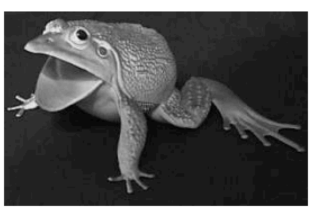
Pose-estimation / Camera calibration

- Motivation: panorama stitching
  - We have two images – how do we combine them?

Step 1: extract correspondence
Step 2: align images

Why correspondence?

- Recognition: Match image to product view



Lowe, IJCV 2004

Other applications of correspondence

- Image alignment
- Motion tracking
- Robot navigation

Correspondence can be challenging

Fei-Fei Li

Correspondence

by [Diva Sian]

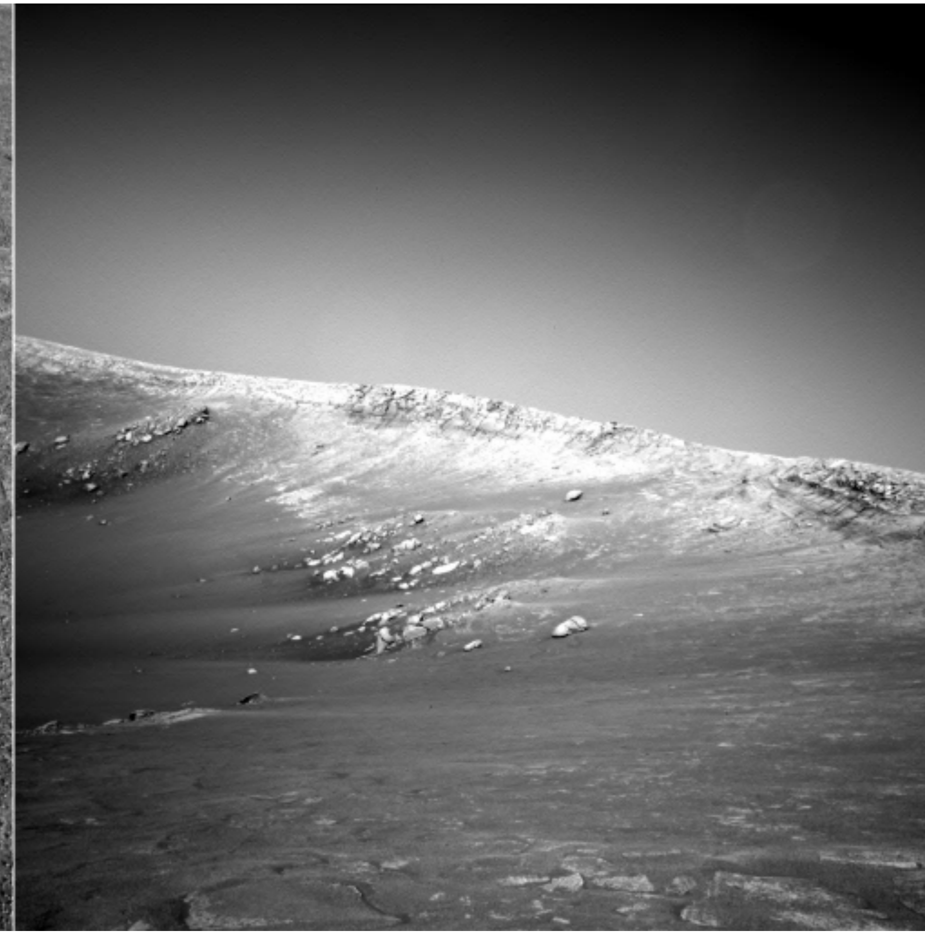
by [swashford]

Harder case

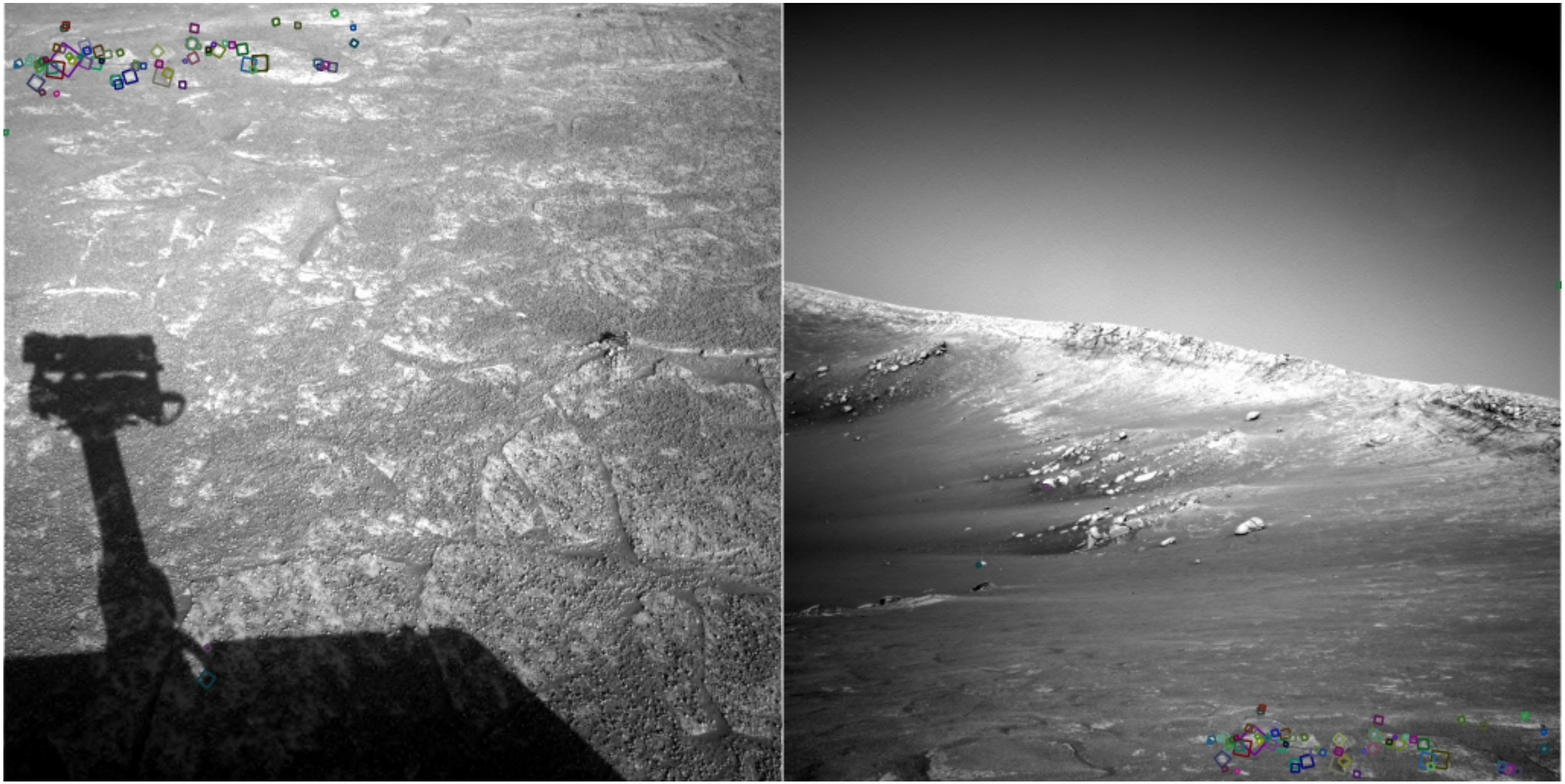
by Diva Sian

by scgbt

Harder still?



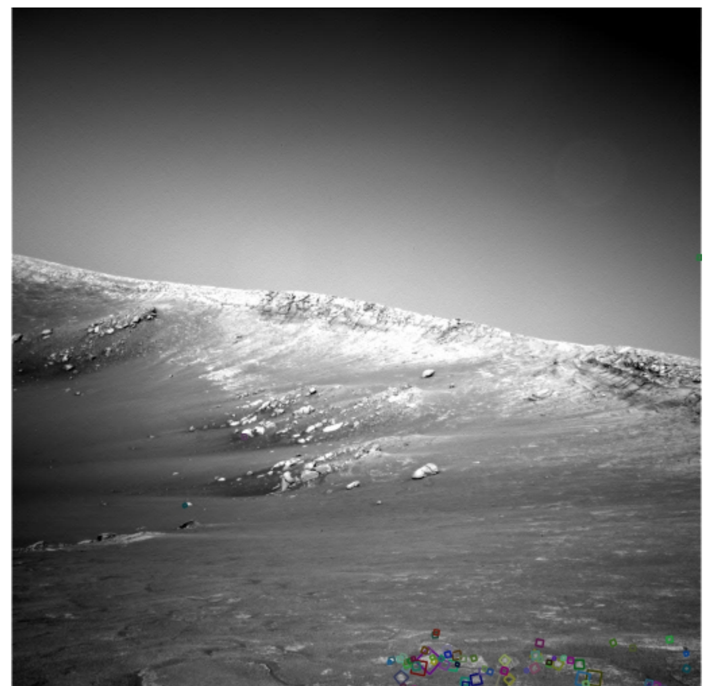
Answer below (look for tiny colored squares…)



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
- Dense correspondence: try to match every pixel
  - Needed if we want 3D location of every pixel

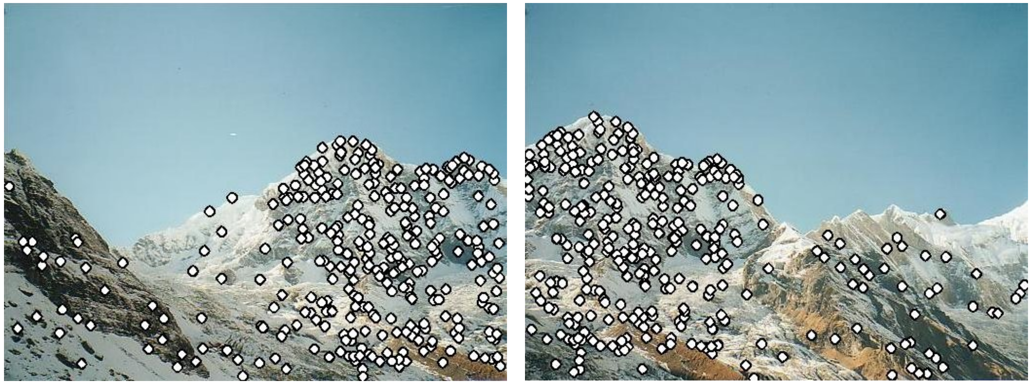


Sparse correspondence

- Which pixels should be searching correspondence for?
  - *Feature points / keypoints*

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

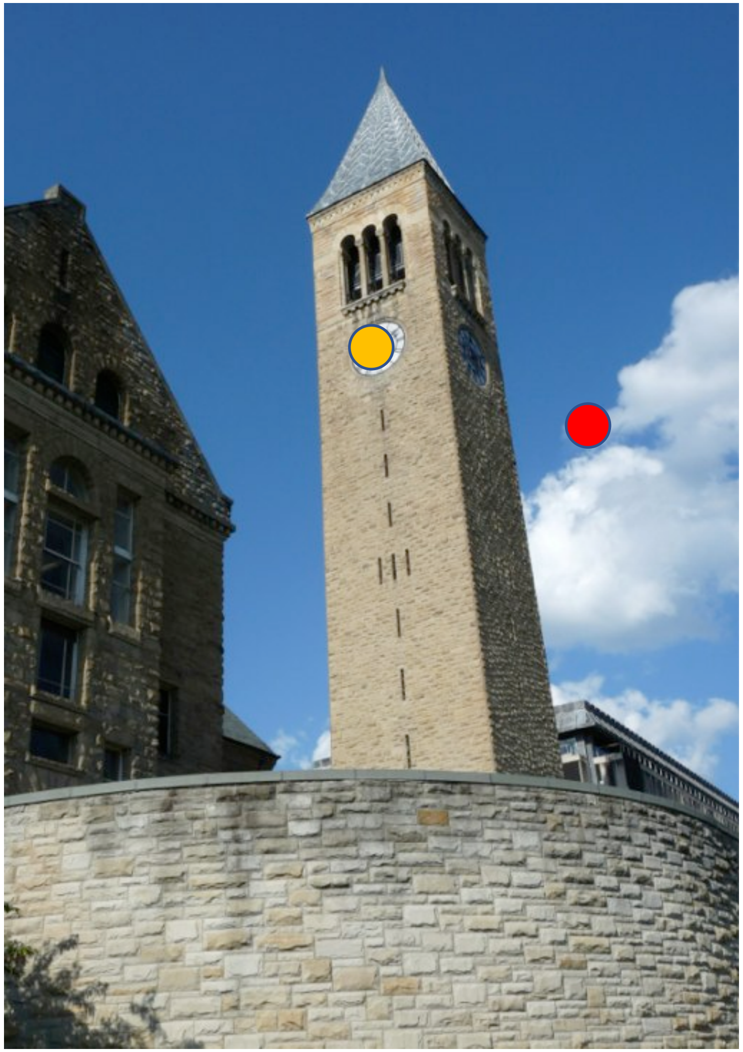
- We want to detect (at least some of) the same points in both images.

No chance to find true matches!

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

Kristen Grauman

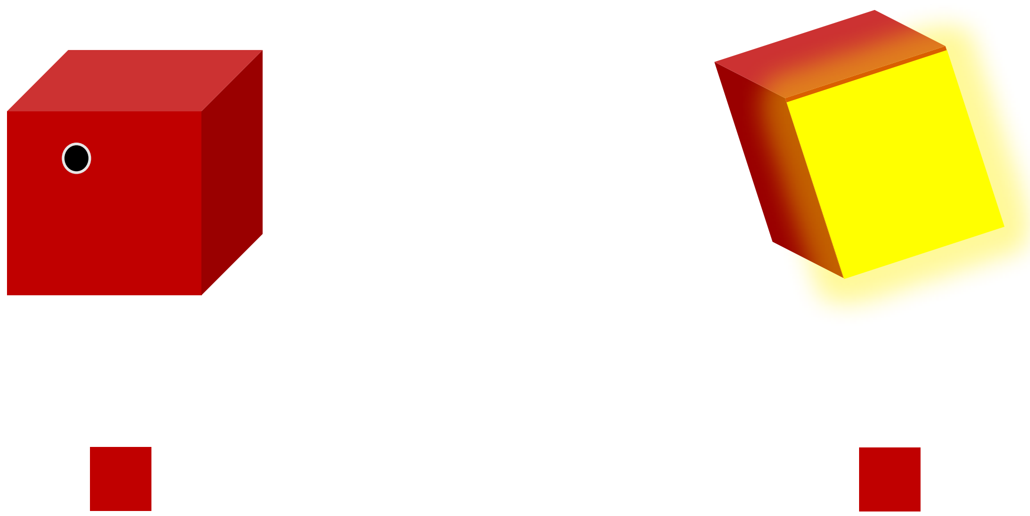
Repeatability / invariance



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

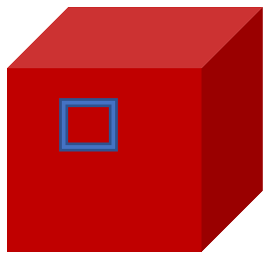
- Individual pixels are ambiguous
- Idea: Look at whole patches!

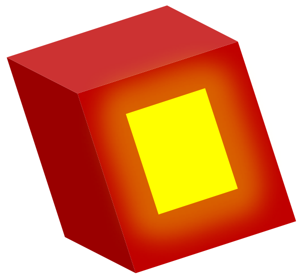
The aperture problem

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

The aperture problem

- *Some local neighborhoods* are ambiguous





The aperture problem