CS 558: Computer Vision
7th Set of Notes

Instructor: Philippos Mordohai
Webpage: www.cs.stevens.edu/~mordohai
E-mail: Philippos.Mordohai@stevens.edu
Office: Lieb 215

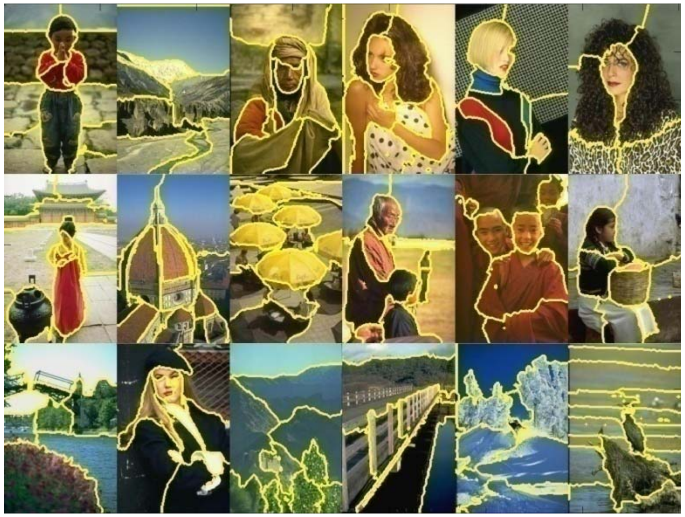
Overview

- Segmentation and Grouping
  - Based on slides by K. Grauman and D. Hoiem
- Camera geometry
  - Based on slides by M. Pollefeys, R. Hartley and A. Zisserman

Segmentation and Grouping

Slides by Derek Hoiem and
Kristen Grauman

Examples of grouping in vision

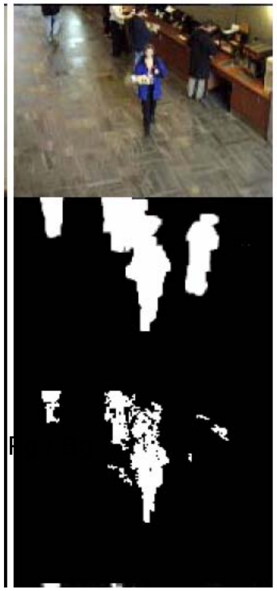


[Figure by J. Shi]

Determine image regions

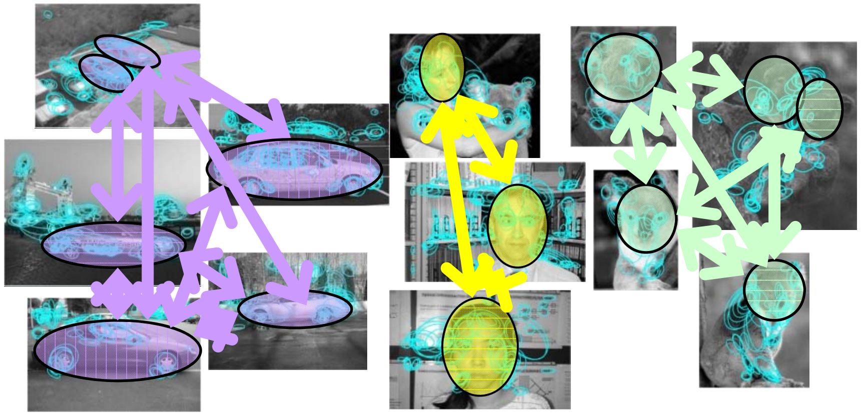
[Figure by I. Pitas et al.]

Group video frames into shots



[Figure by Wang & Suter]

Figure-ground



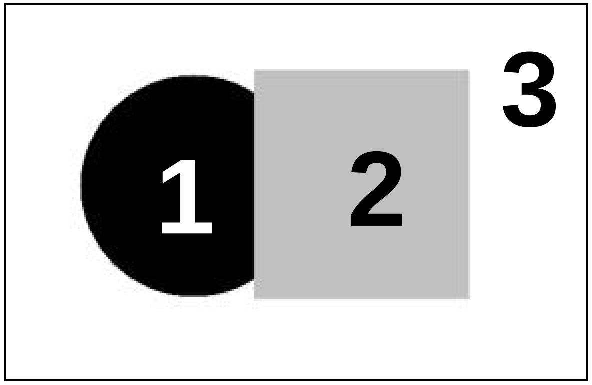
[Figure by Grauman & Darrell]

Object-level grouping

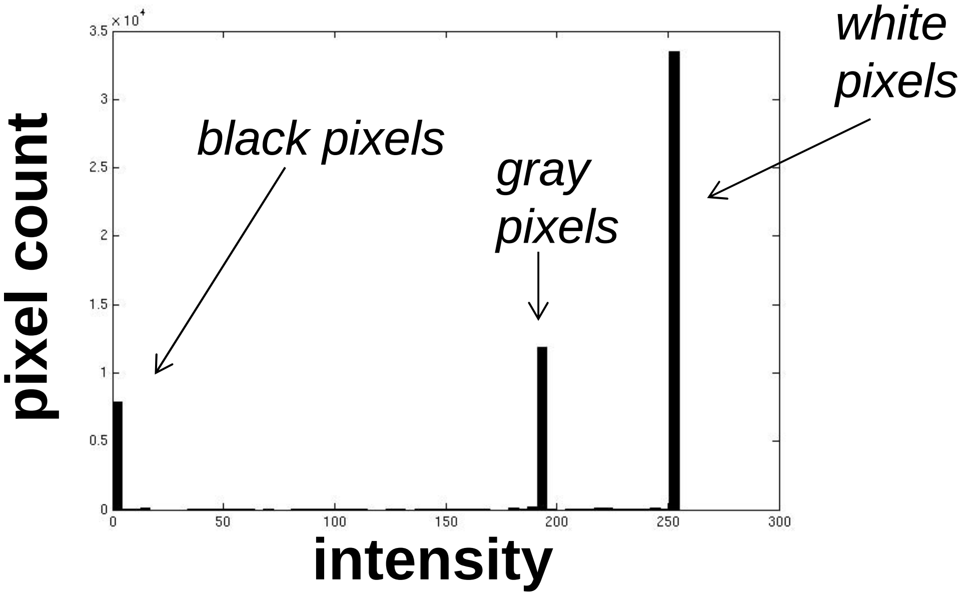
Grouping in vision

- Goals:
  - Gather features that belong together
  - Obtain an intermediate representation that compactly describes key image (video) parts
- Top down vs. bottom up segmentation
  - Top down: pixels belong together because they are from the same object
  - Bottom up: pixels belong together because they look similar
- Hard to measure success
  - What is interesting depends on the application

Image segmentation: toy example

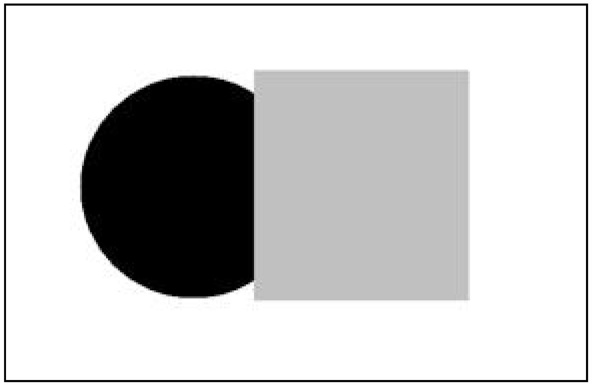


input image

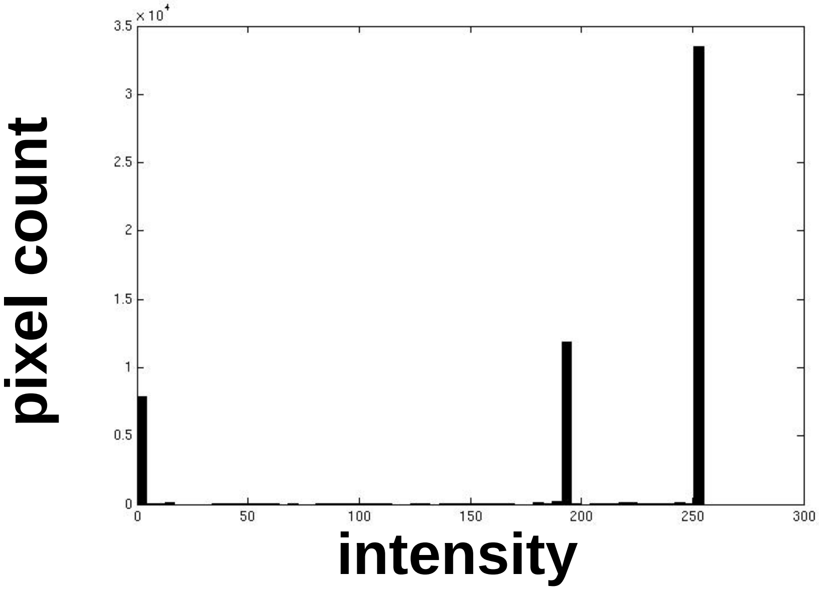


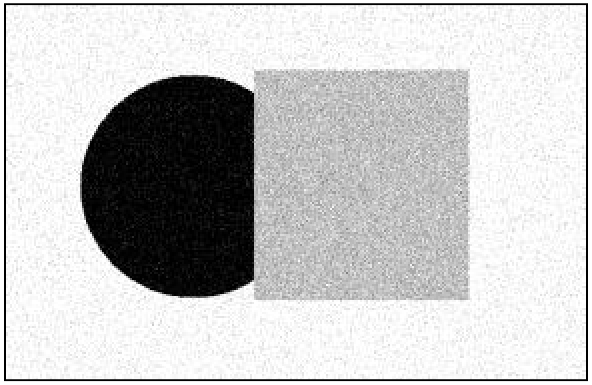
- These intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
  - i.e., *segment* the image based on the intensity feature.
- What if the image isn’t quite so simple?

Kristen Grauman

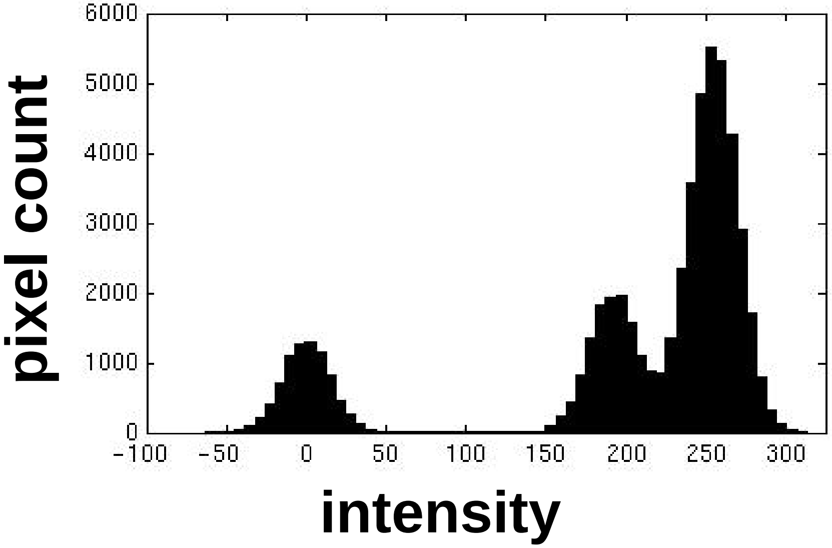


input image

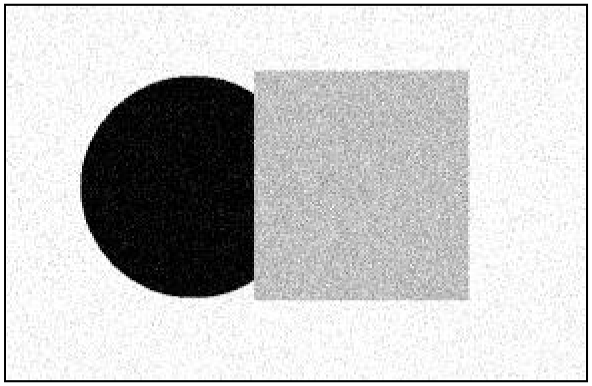




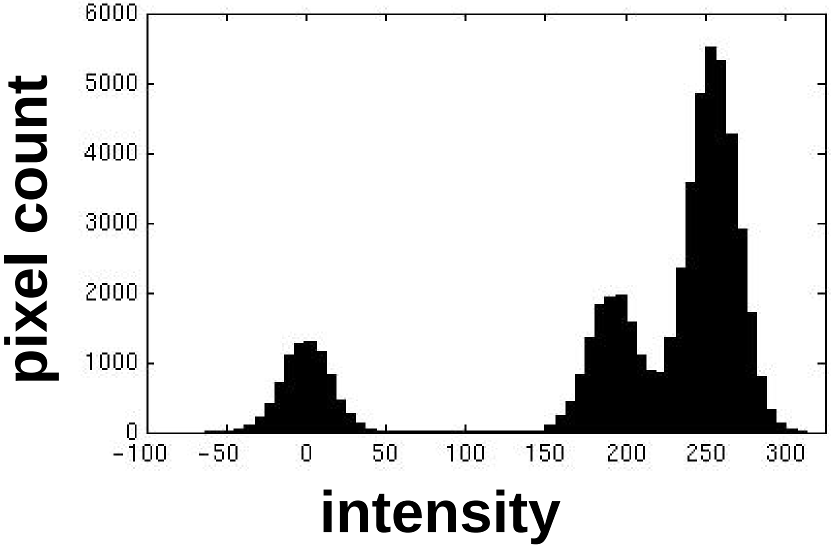
input image



Kristen Grauman

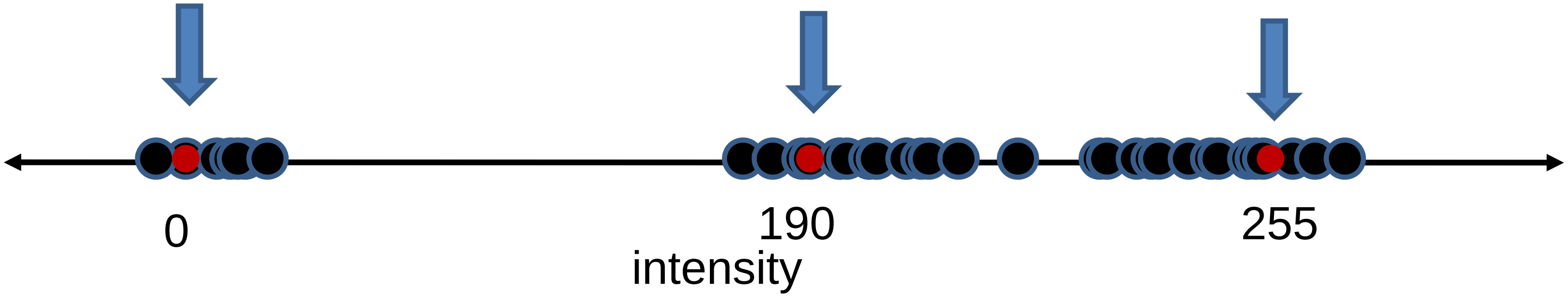


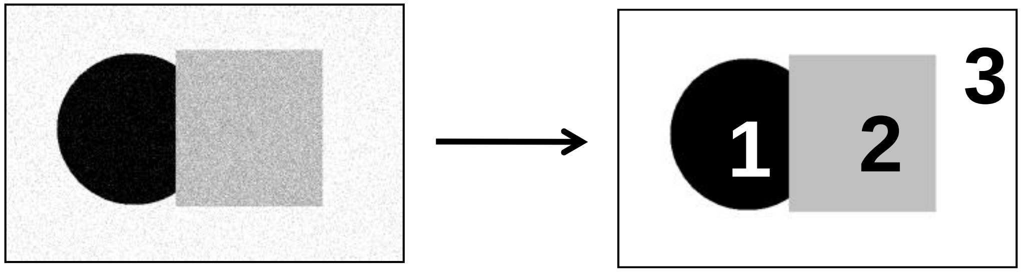
input image



- Now how to determine the three main intensities that define our groups?
- We need to ***cluster.***

Kristen Grauman





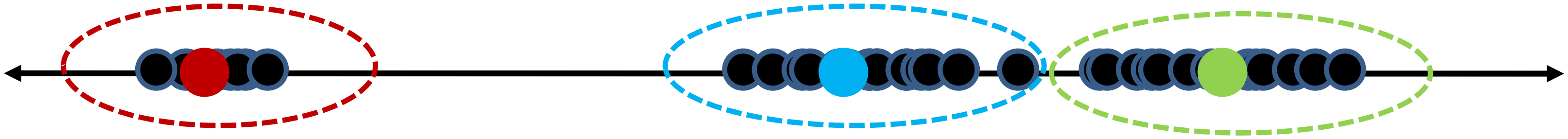
- Goal: choose three “centers” as the representative intensities, and label every pixel according to which of these centers it is nearest to.
- Best cluster centers are those that minimize SSD between all points and their nearest cluster center c_i:

$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

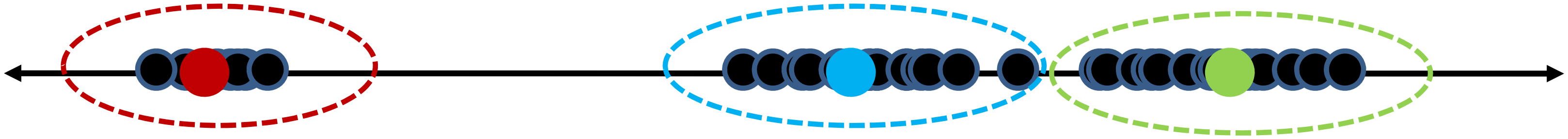
Kristen Grauman

Clustering

- With this objective, it is a “chicken and egg” problem:
  - If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.



  - If we knew the **group memberships**, we could get the centers by computing the mean per group.

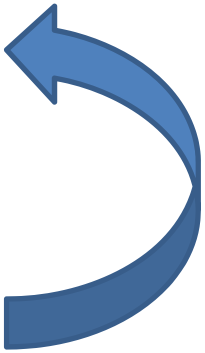


Kristen Grauman

K-means clustering

- Basic idea: randomly initialize the *k* cluster centers, and iterate between the two steps we just saw.

1. Randomly initialize the cluster centers, c_1, ..., c_K
2. Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put p into cluster i
3. Given points in each cluster, solve for c_i
   - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2



Properties

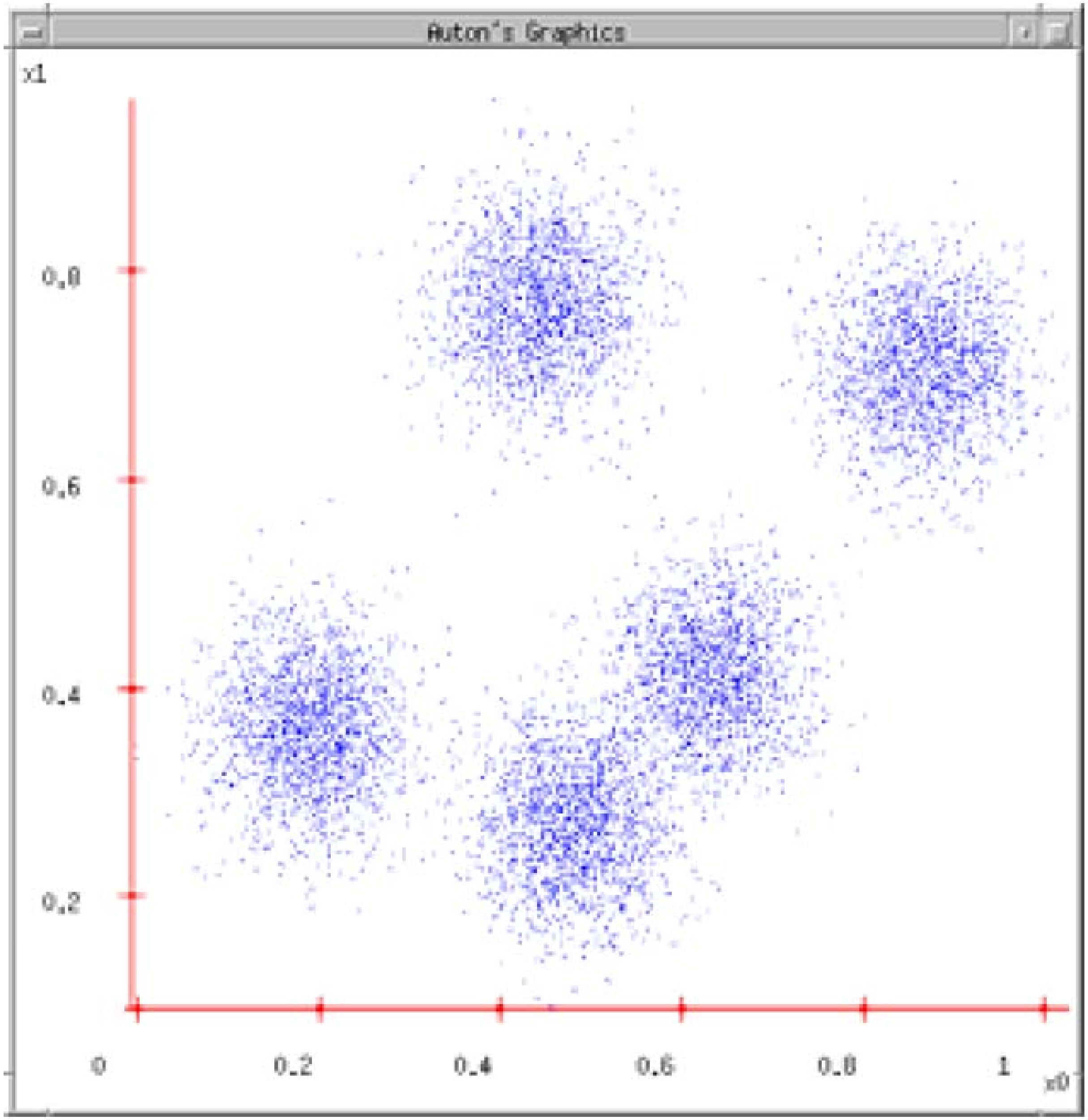
- Will always converge to *some* solution
- Can be a “local minimum”
  - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

Source: Steve Seitz

K-means

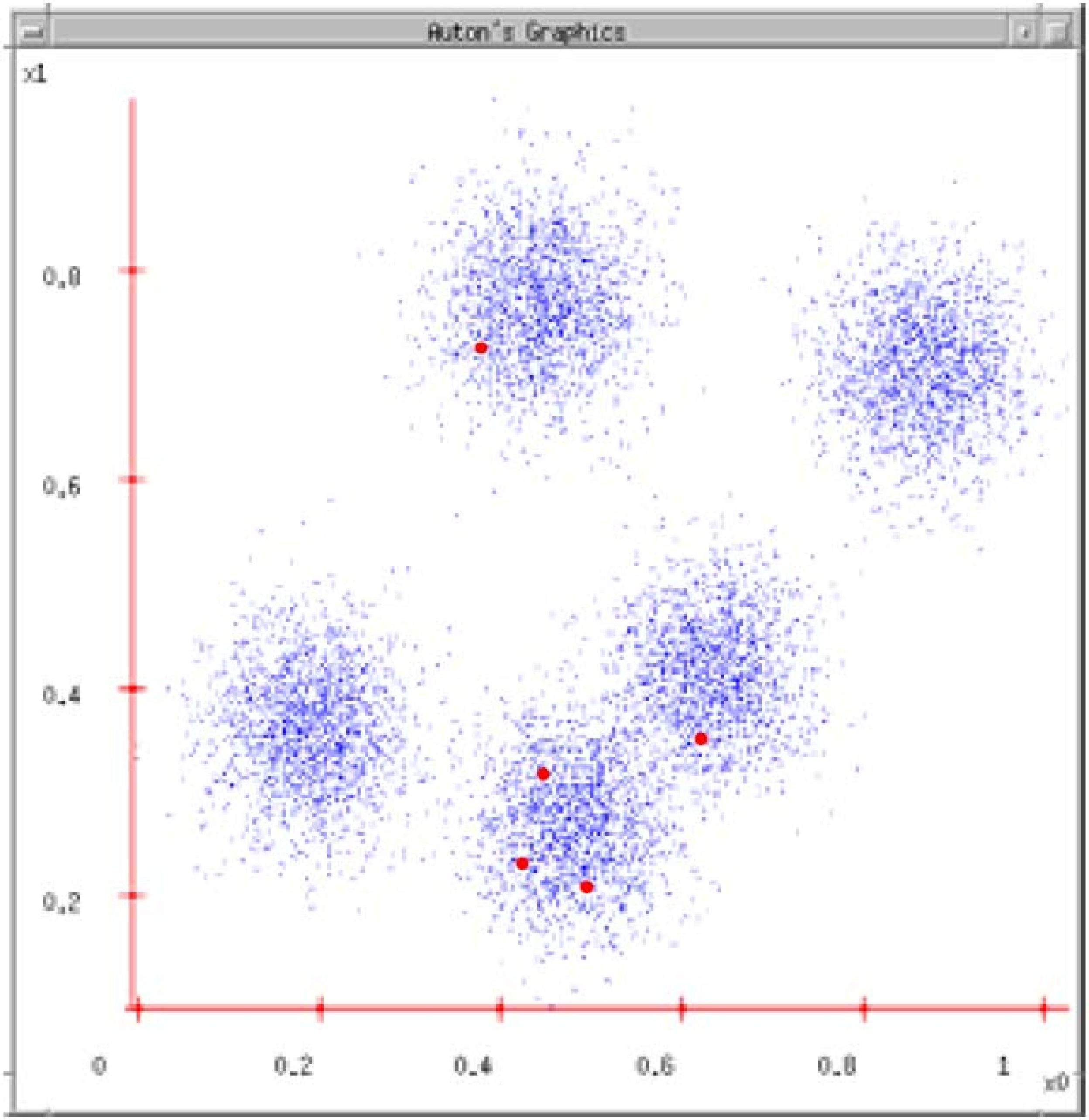
1. Ask user how many clusters they'd like. *(e.g. k=5)*



Andrew Moore

K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations



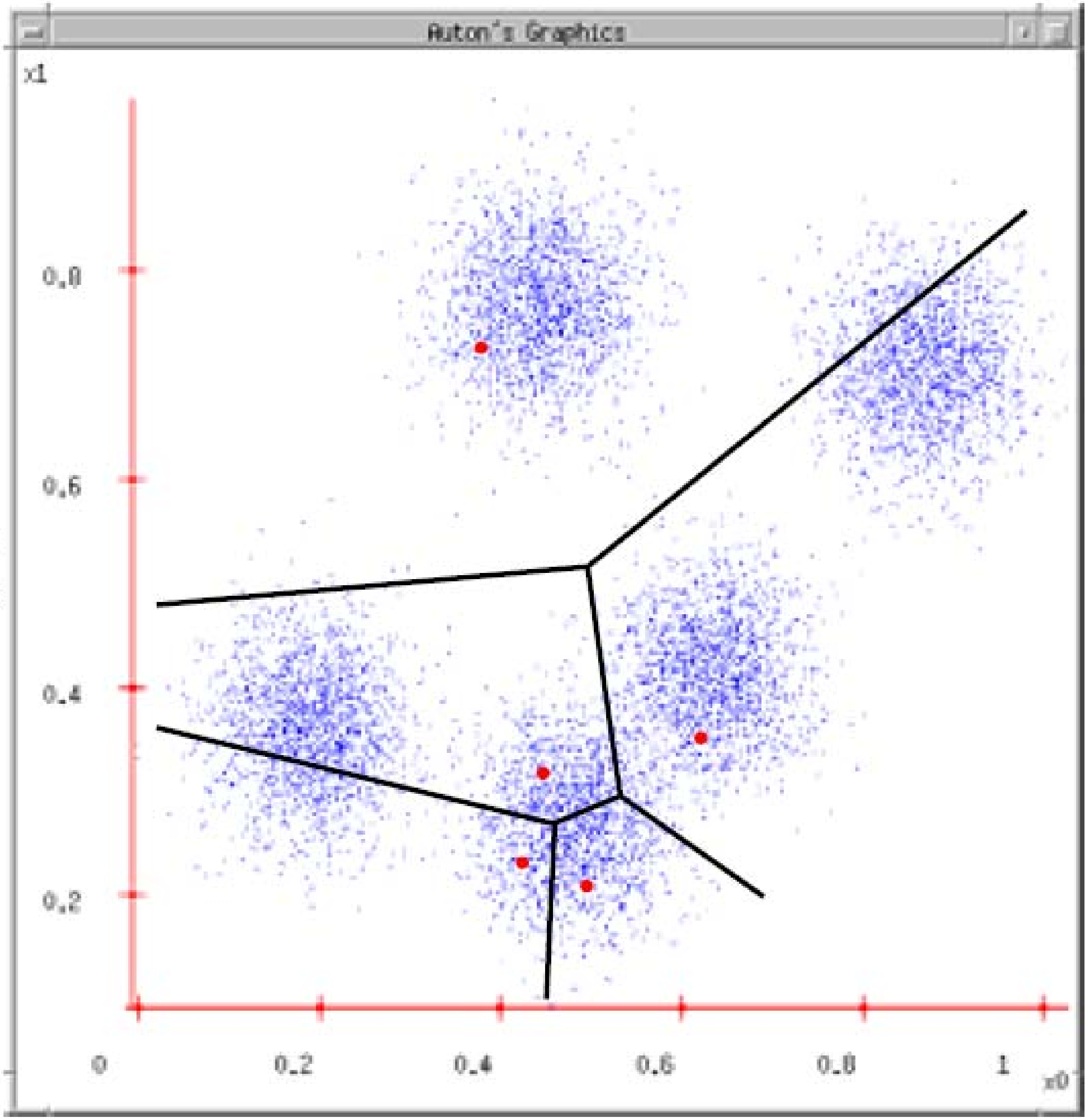
Andrew Moore

K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*

2. Randomly guess k cluster Center locations

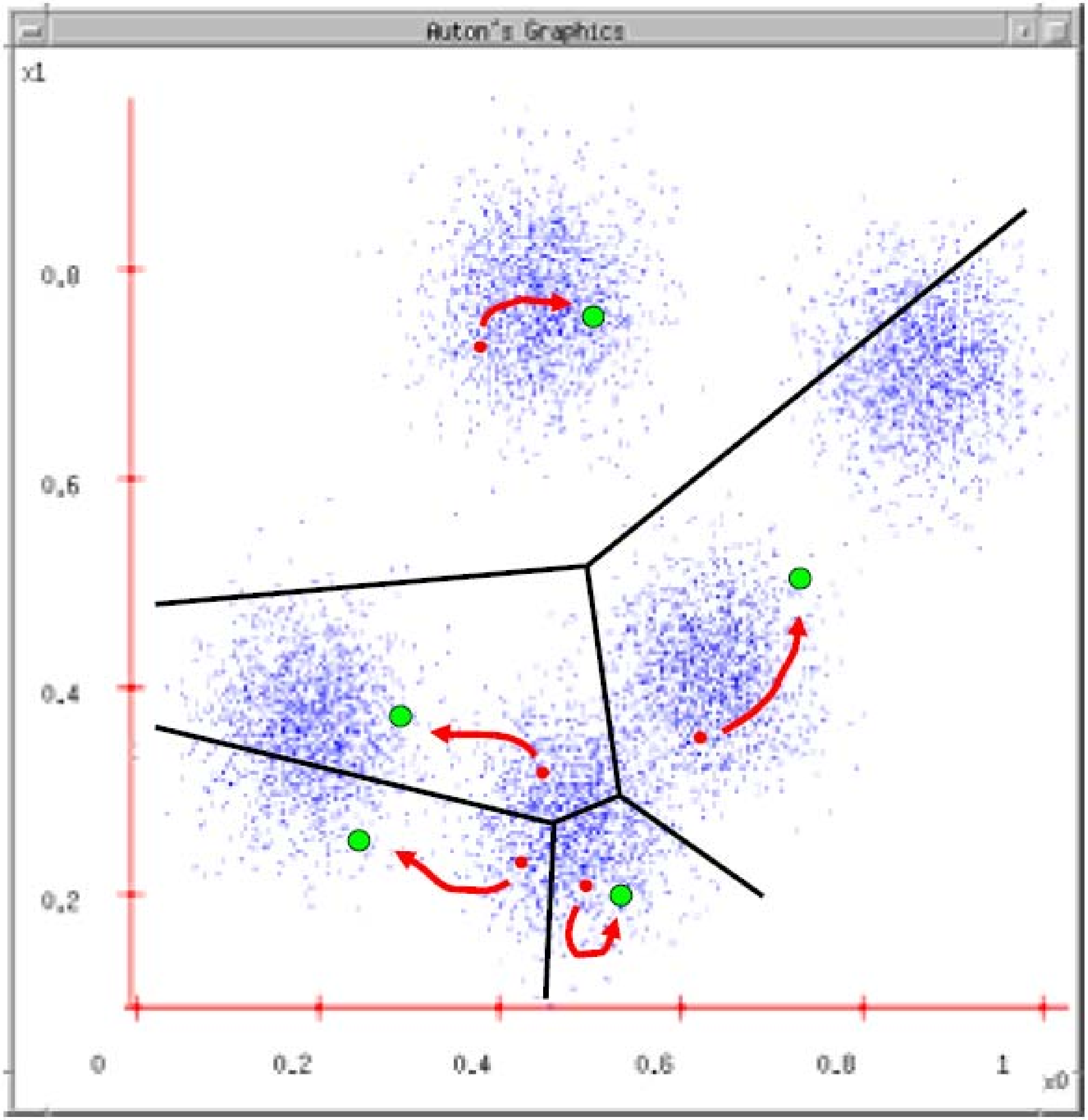
3. Each datapoint finds out which Center it's closest to. (Thus each Center "owns" a set of datapoints)



Andrew Moore

K-means

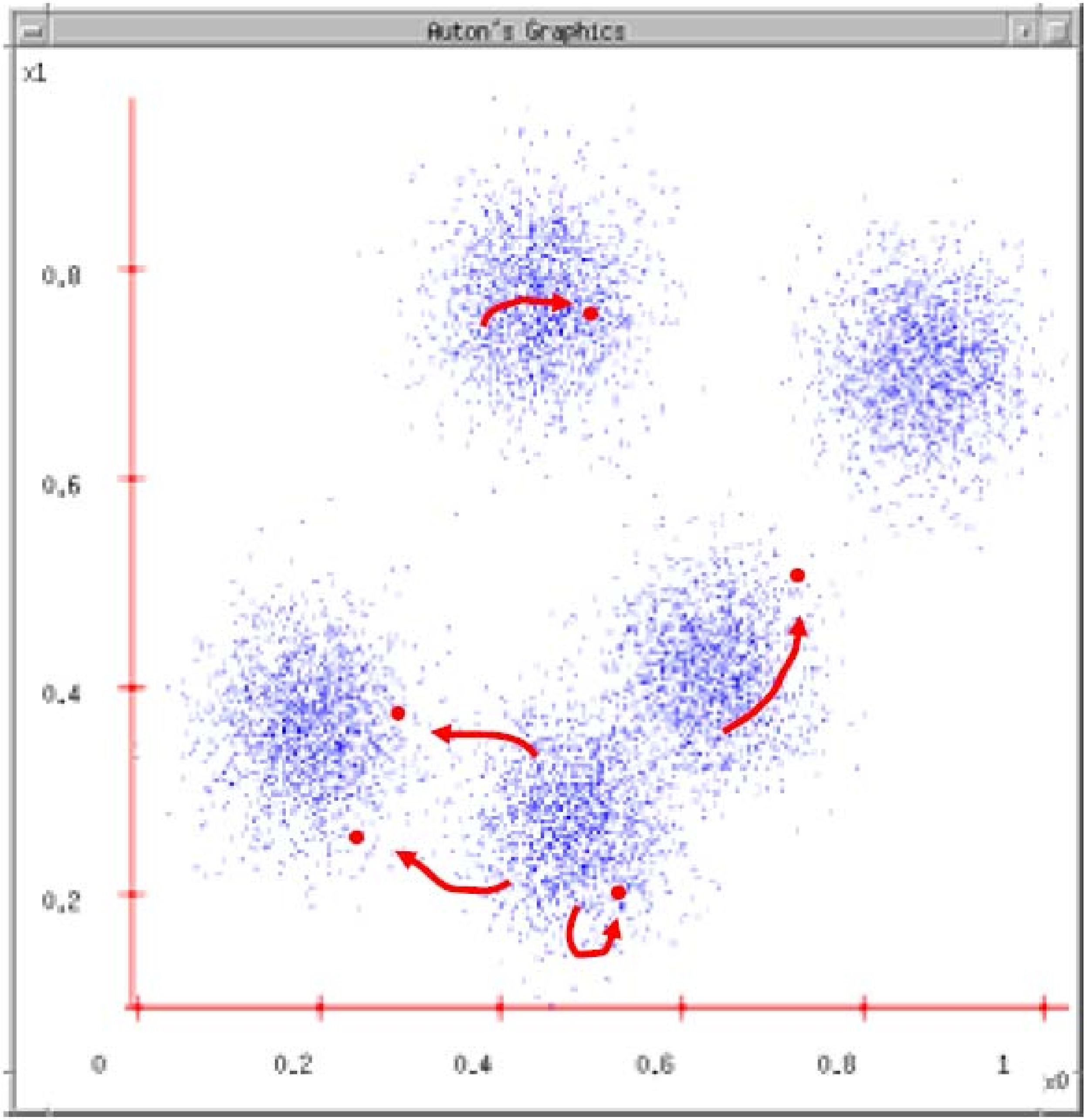
1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



Andrew Moore

K-means

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!



Andrew Moore

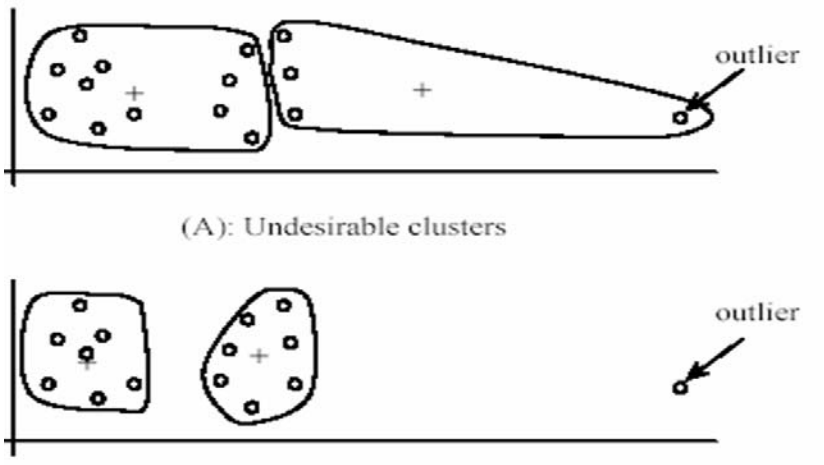
K-means: pros and cons

Pros

- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

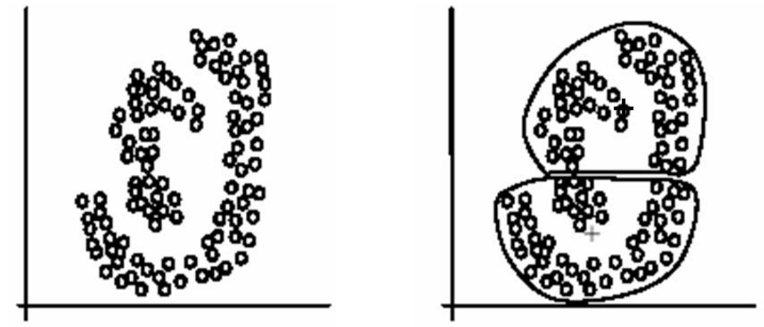
Cons/issues

- Setting k?
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters
- Assumes means can be computed



(A): Undesirable clusters

(B): Ideal clusters



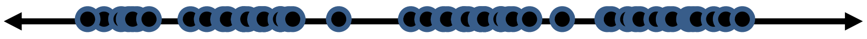
(A): Two natural clusters

(B): *k*-means clusters

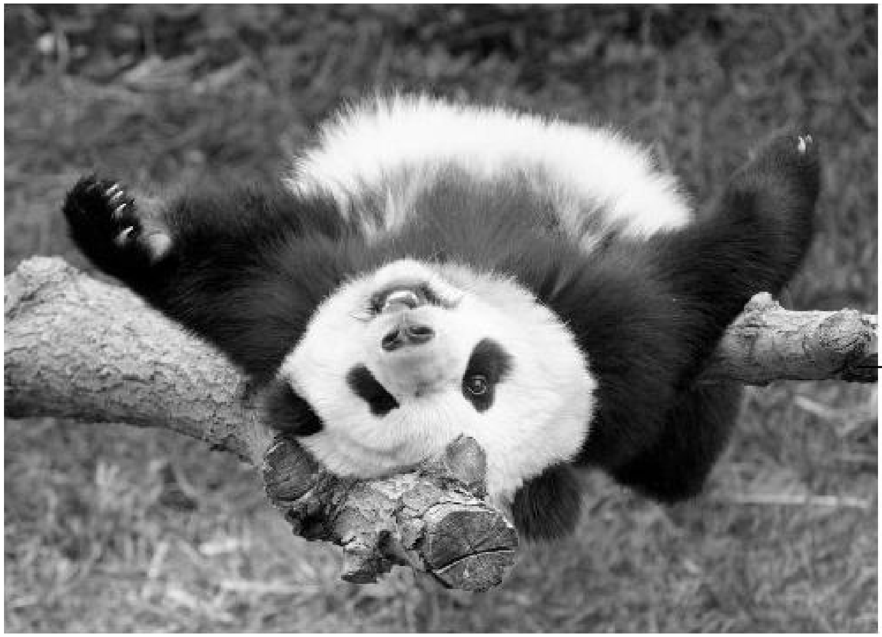
Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

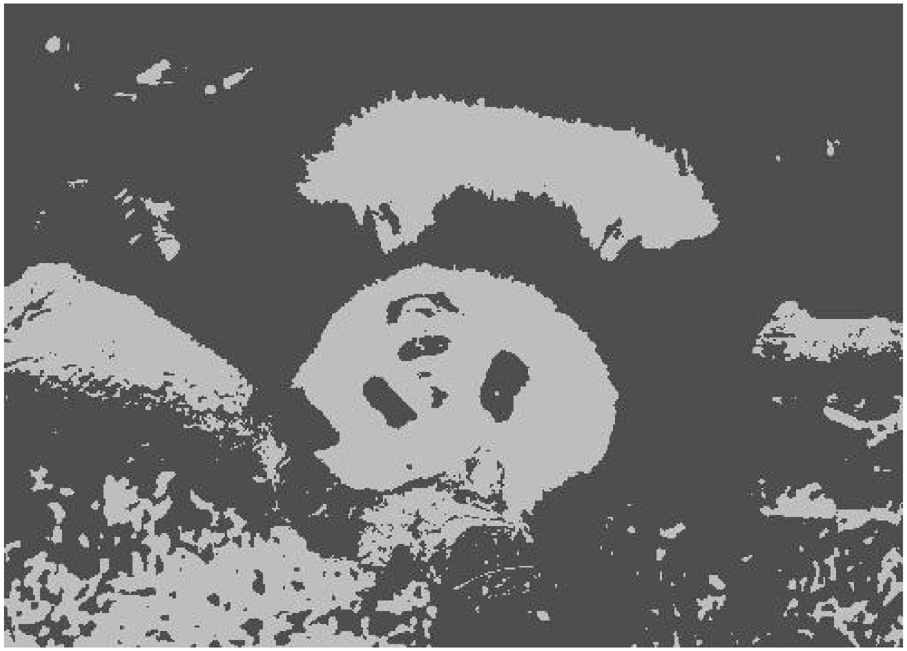
Grouping pixels based on **intensity** similarity



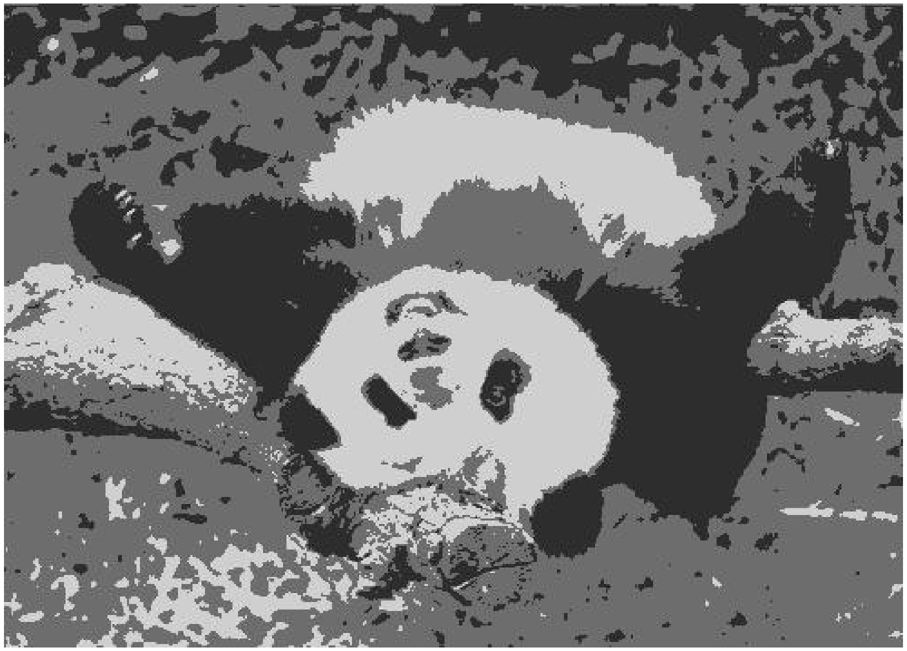
Feature space: intensity value (1-d)



K=2



K=3

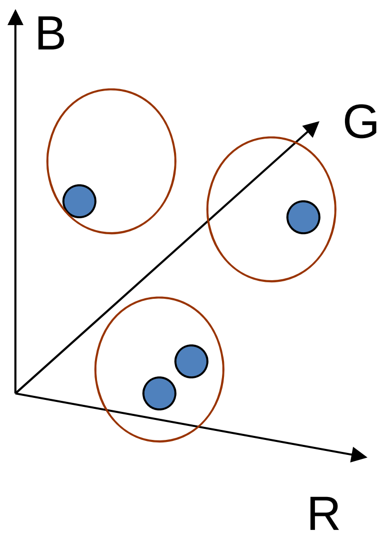


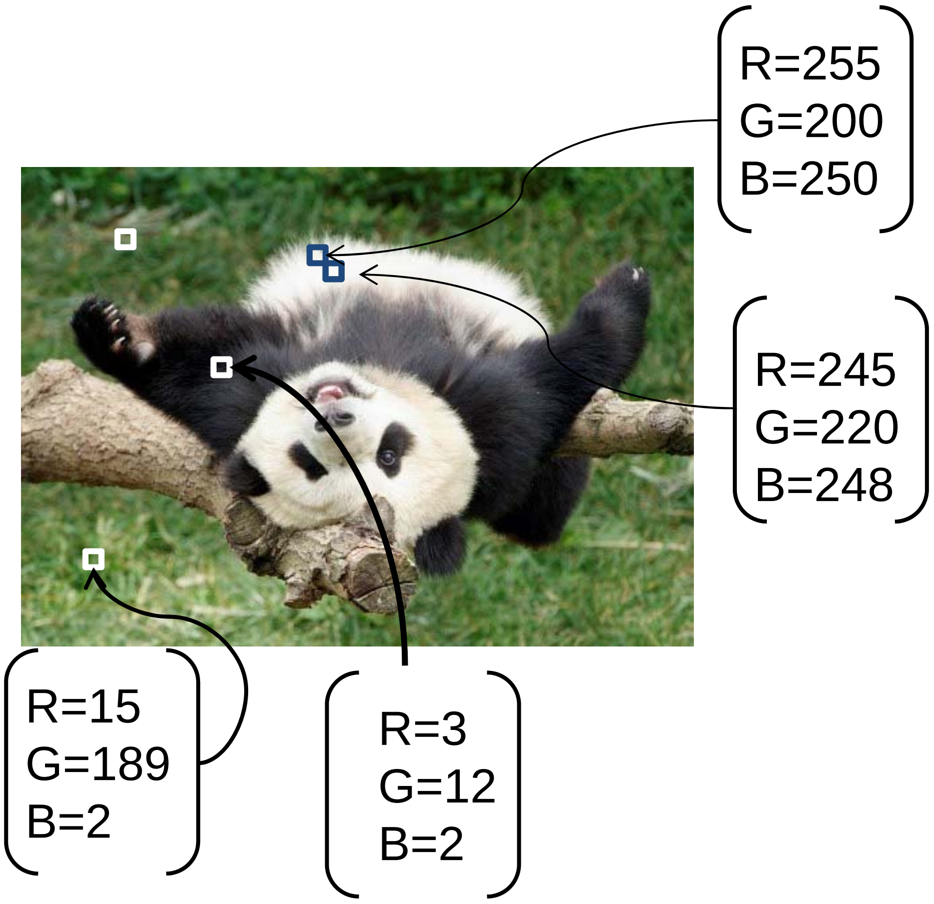
quantization of the feature space
segmentation label map

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **color** similarity





R=255
G=200
B=250

R=245
G=220
B=248

R=15
G=189
B=2

R=3
G=12
B=2

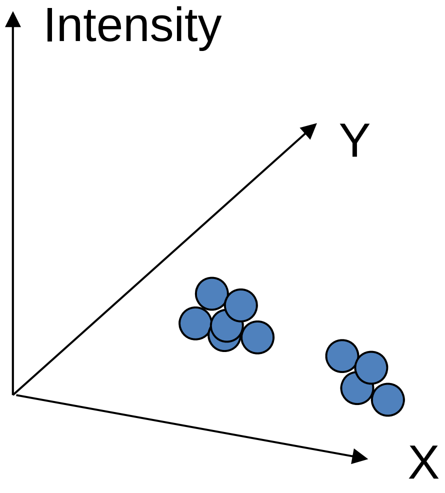
Feature space: color value (3-d)

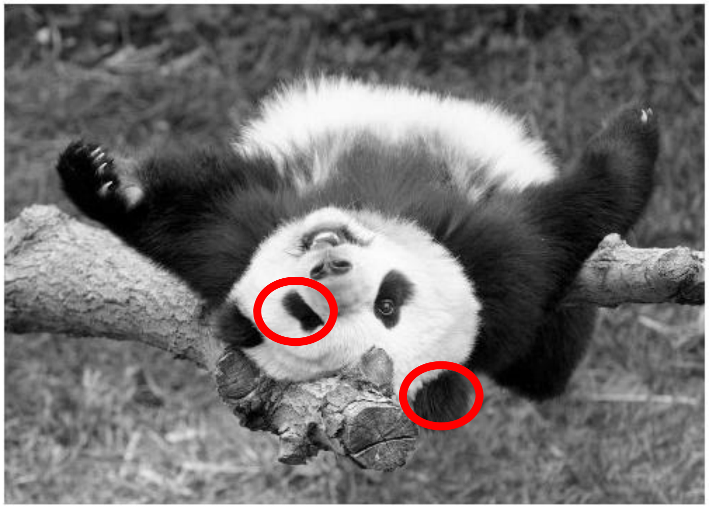
Kristen Grauman

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity+position** similarity





Both regions are black, but if we also include **position (x,y),** then we could group the two into distinct segments; way to encode both similarity & proximity.

Kristen Grauman

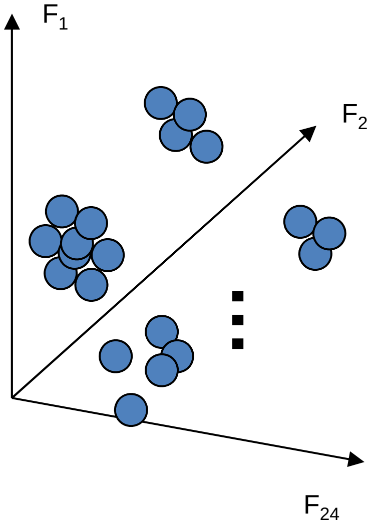
Segmentation as clustering

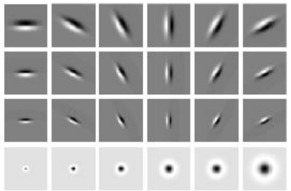
- Color, brightness, position alone are not enough to distinguish all regions…

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **texture** similarity

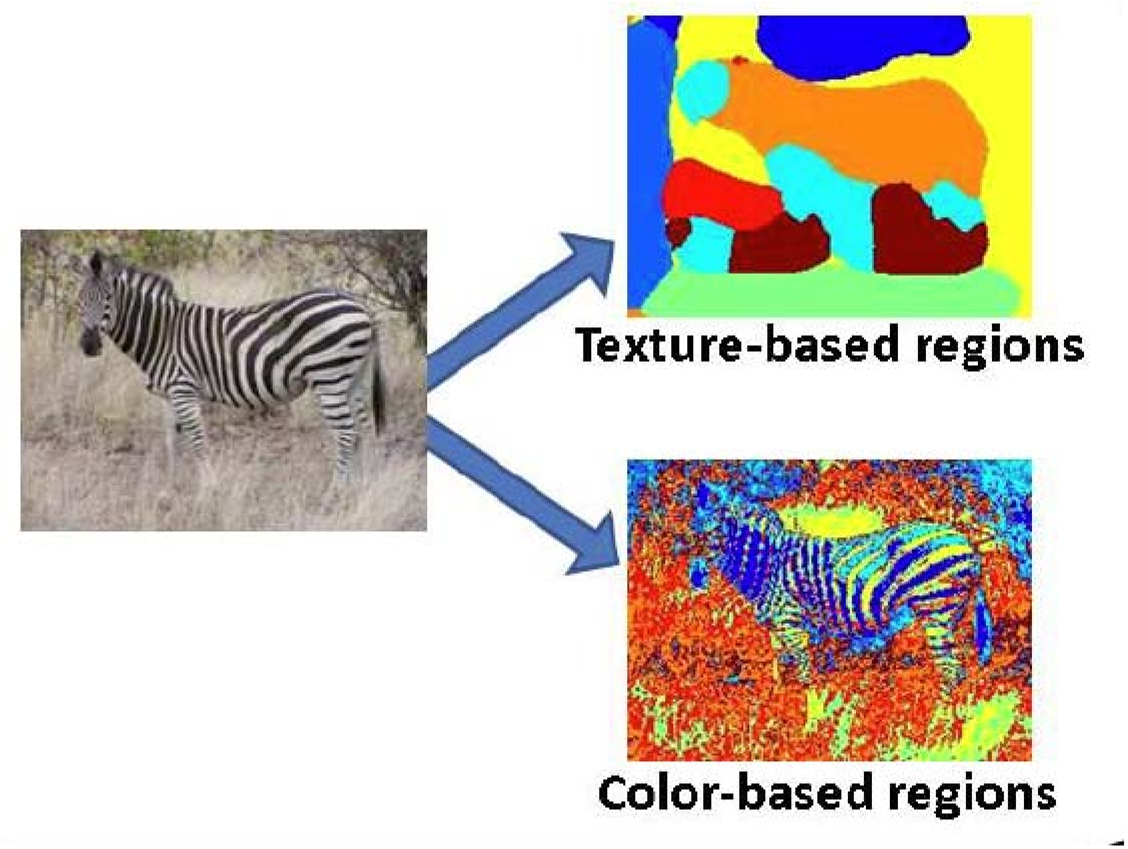




Filter bank of 24 filters

Feature space: filter bank responses (e.g., 24-d)

Image segmentation example

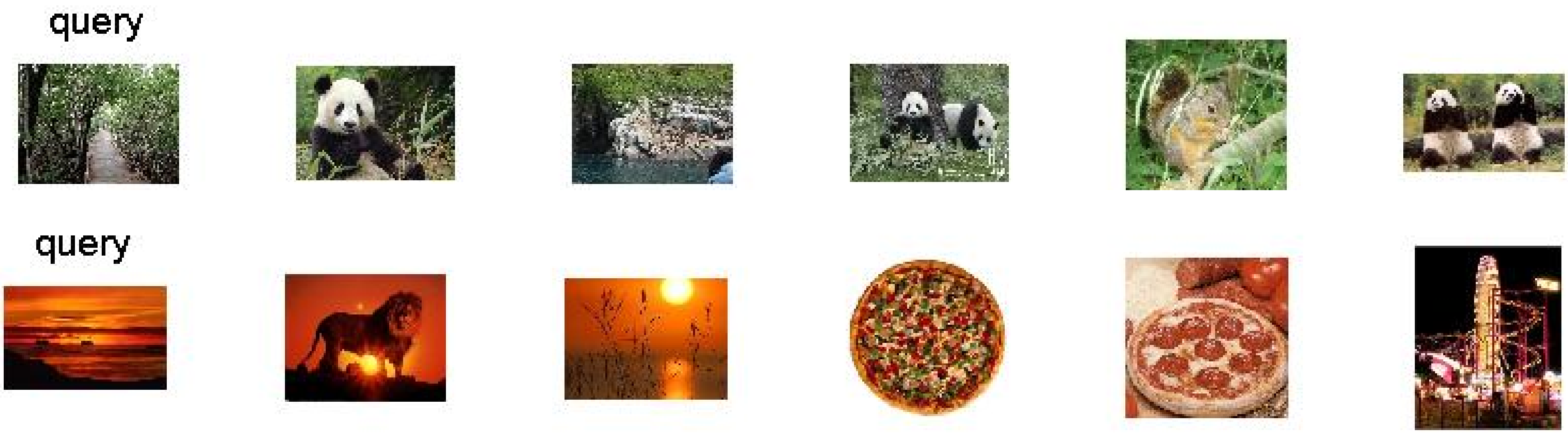


Texture-based regions

Color-based regions

Kristen Grauman

Pixel properties vs. neighborhood properties



These look very similar in terms of their color distributions (histograms).

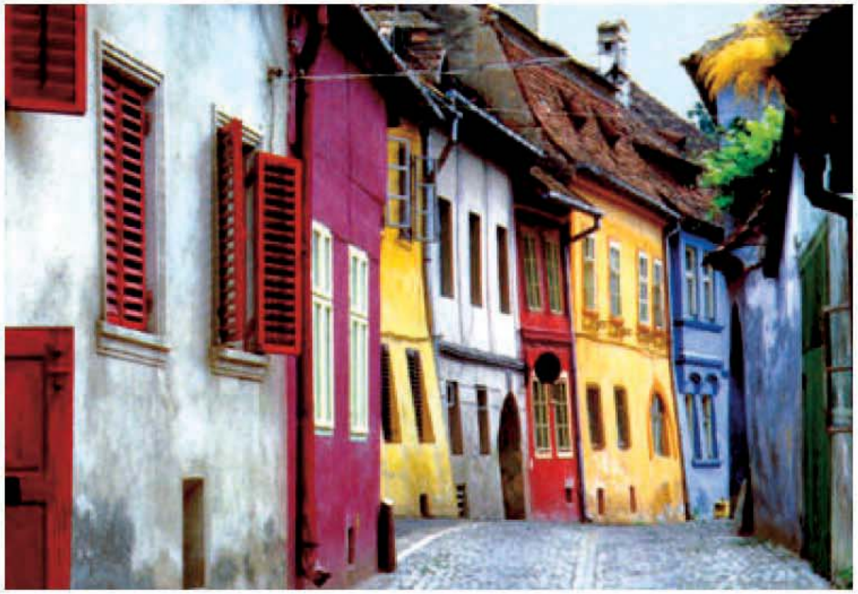
How would their *texture* distributions compare?

Kristen Grauman

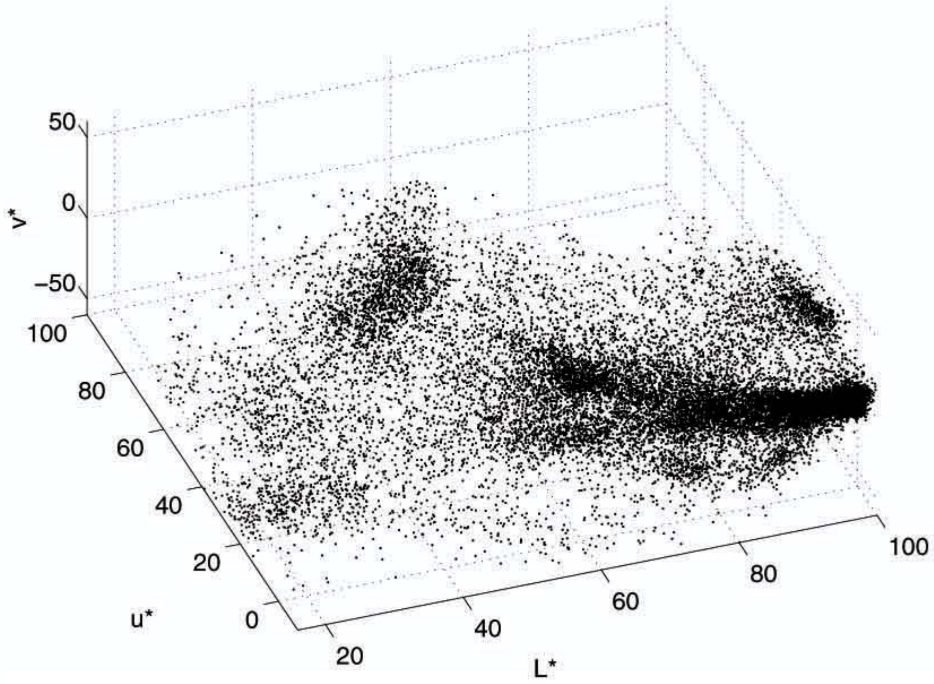
Mean shift algorithm

- The mean shift algorithm [Comaniciu and Meer] seeks *modes* or local maxima of density in the feature space

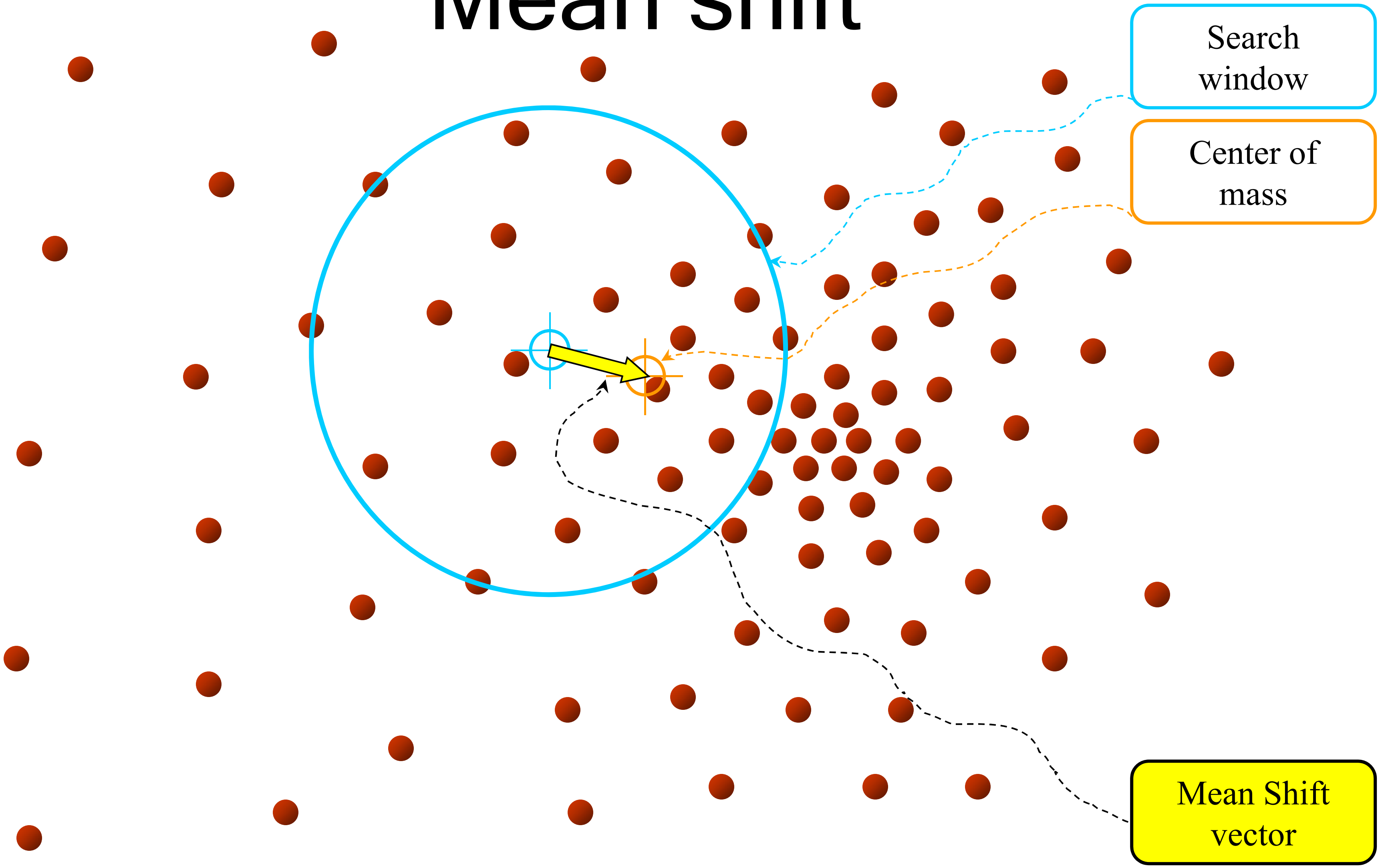
image



Feature space (L*u*v* color values)

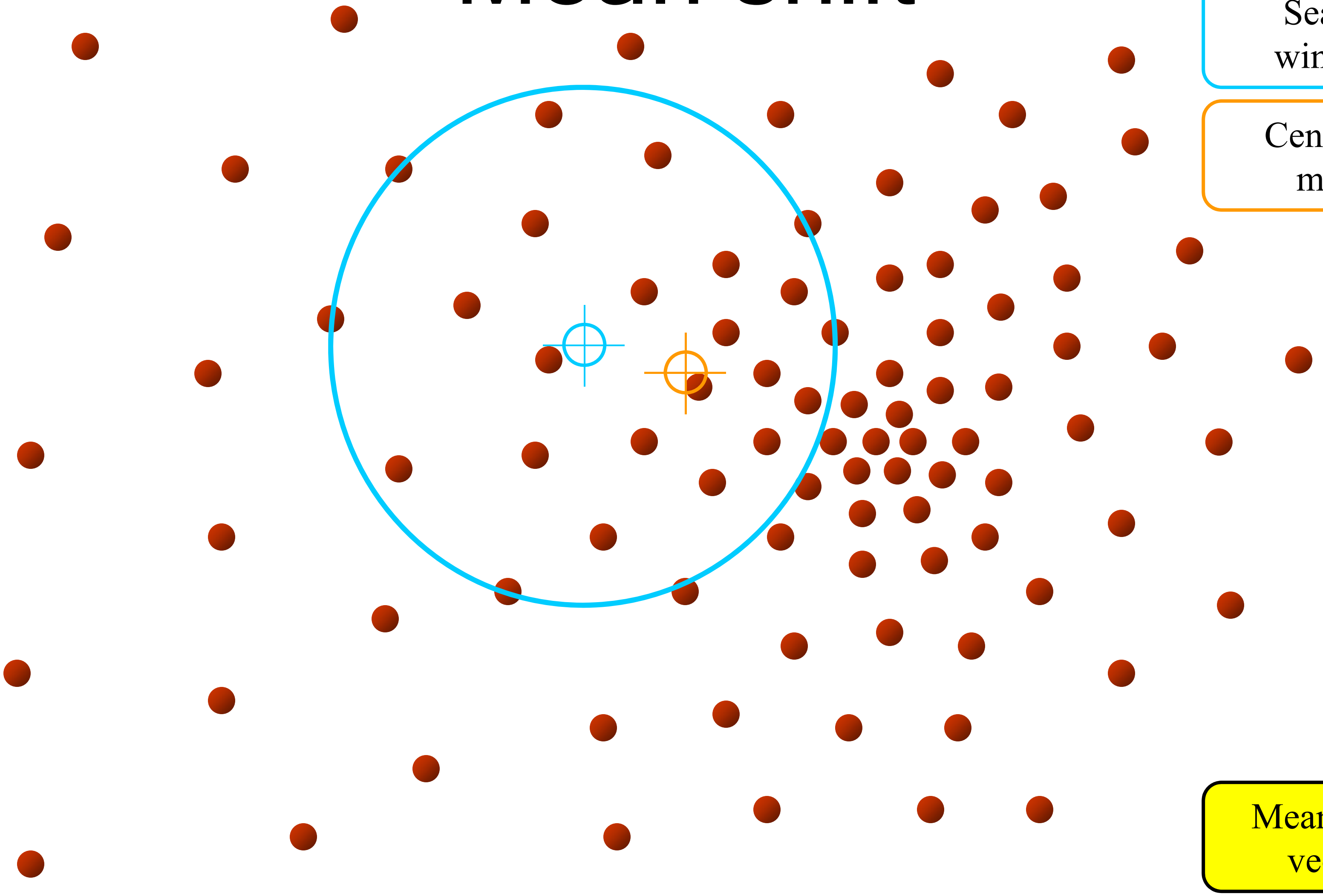


Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift



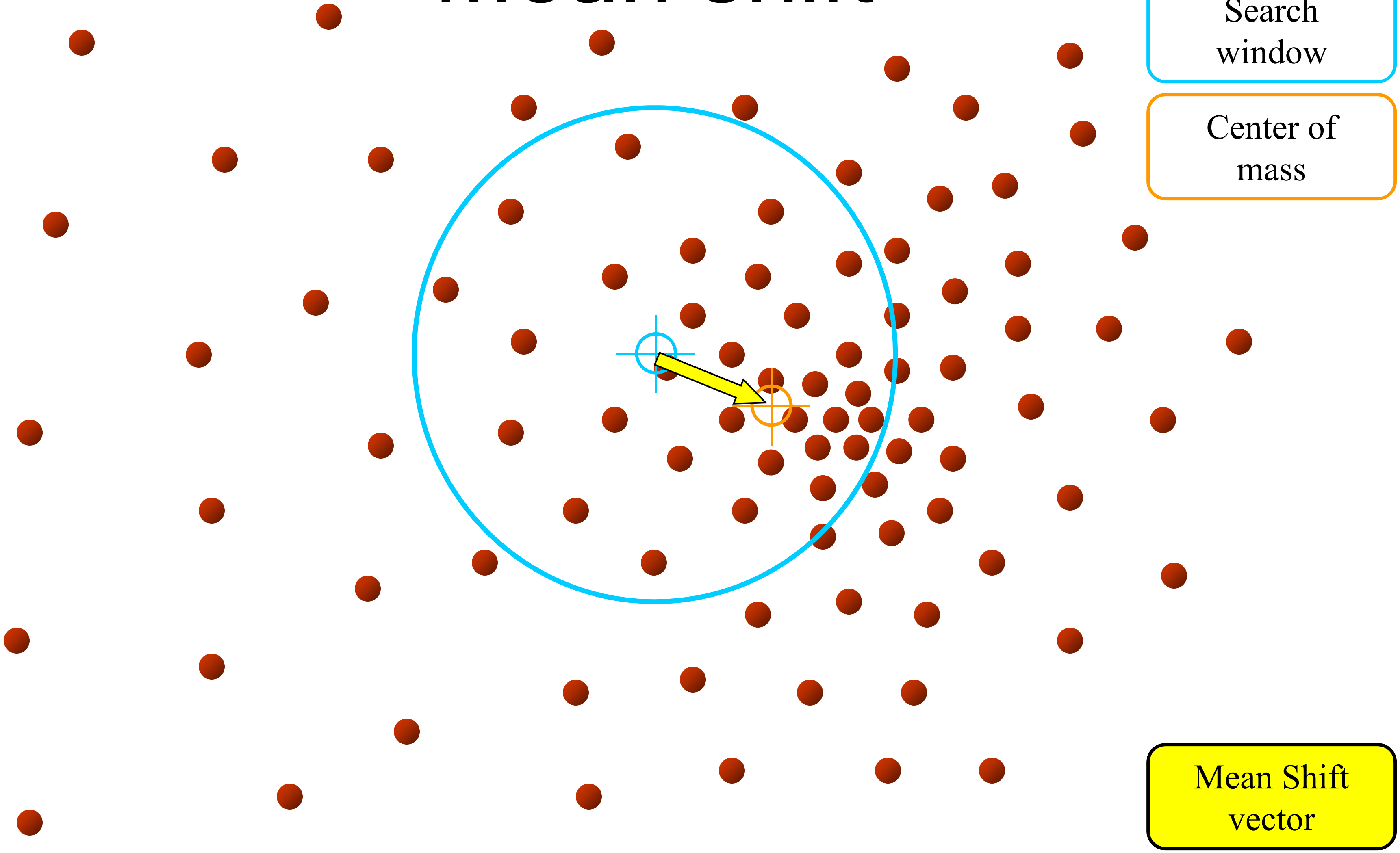
Search window

Center of mass

Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel

Mean shift



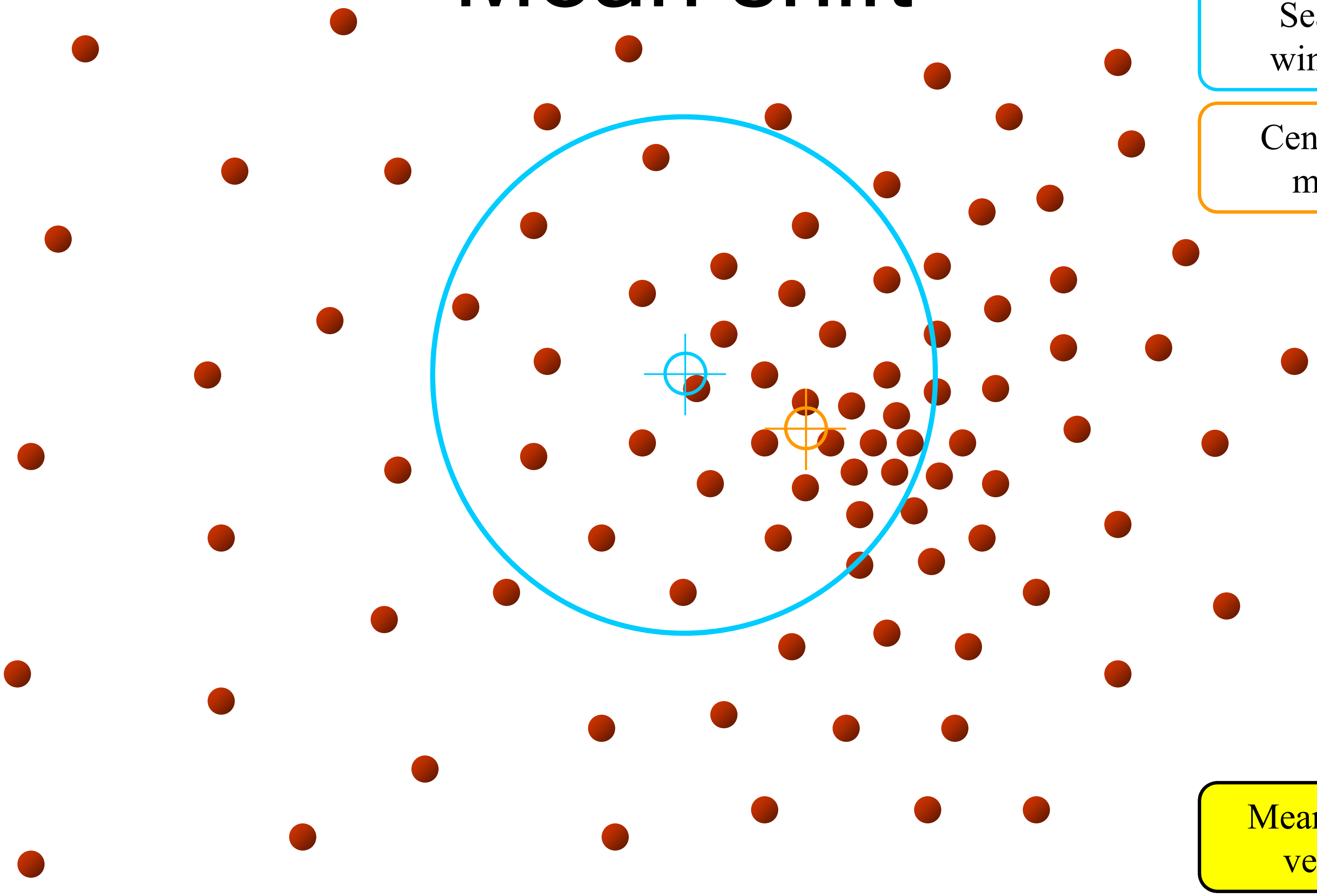
Search window

Center of mass

Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel

Mean shift



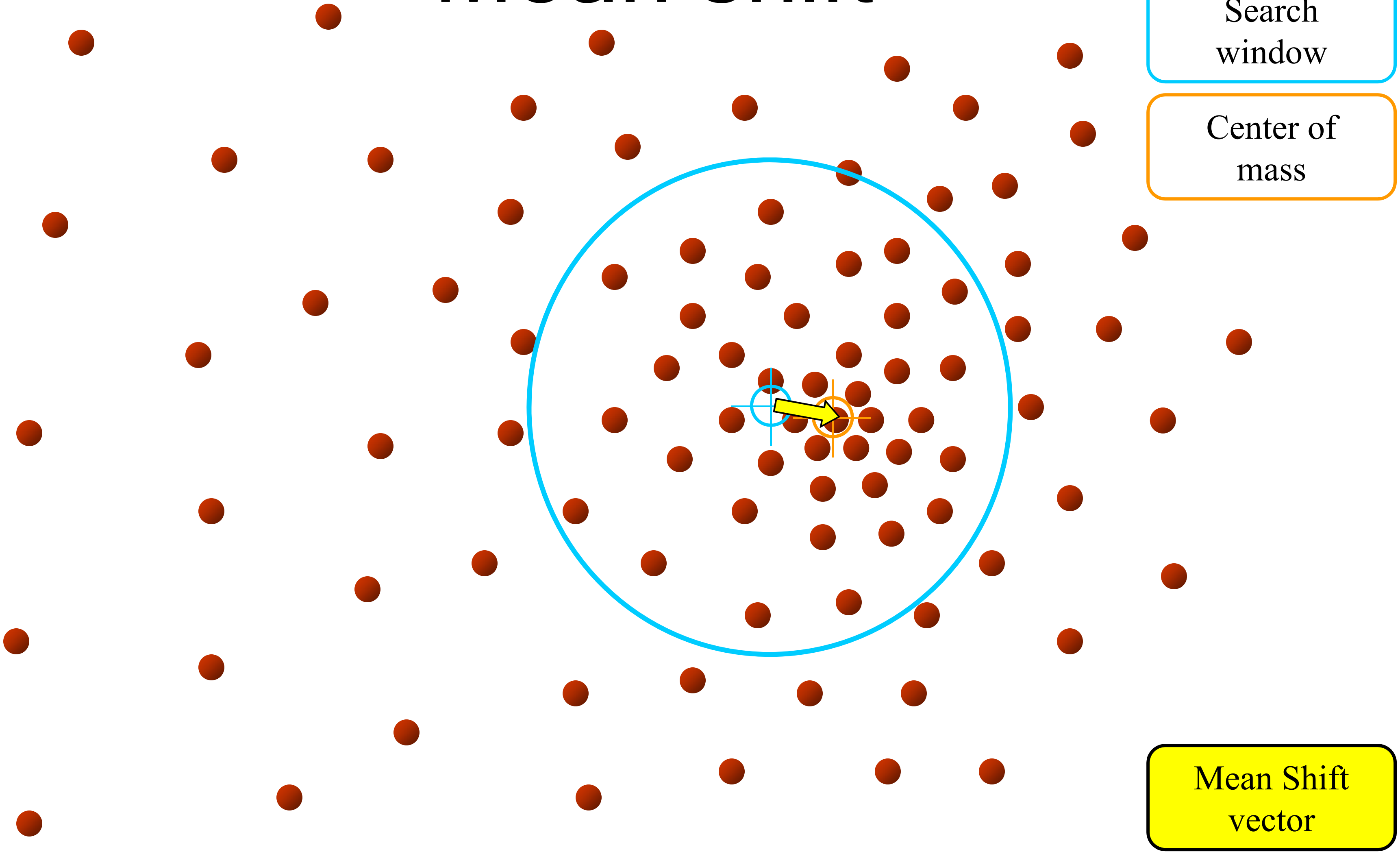
Search window

Center of mass

Mean Shift vector

Slide by Y. Ukrainitz & B. Sarel

Mean shift



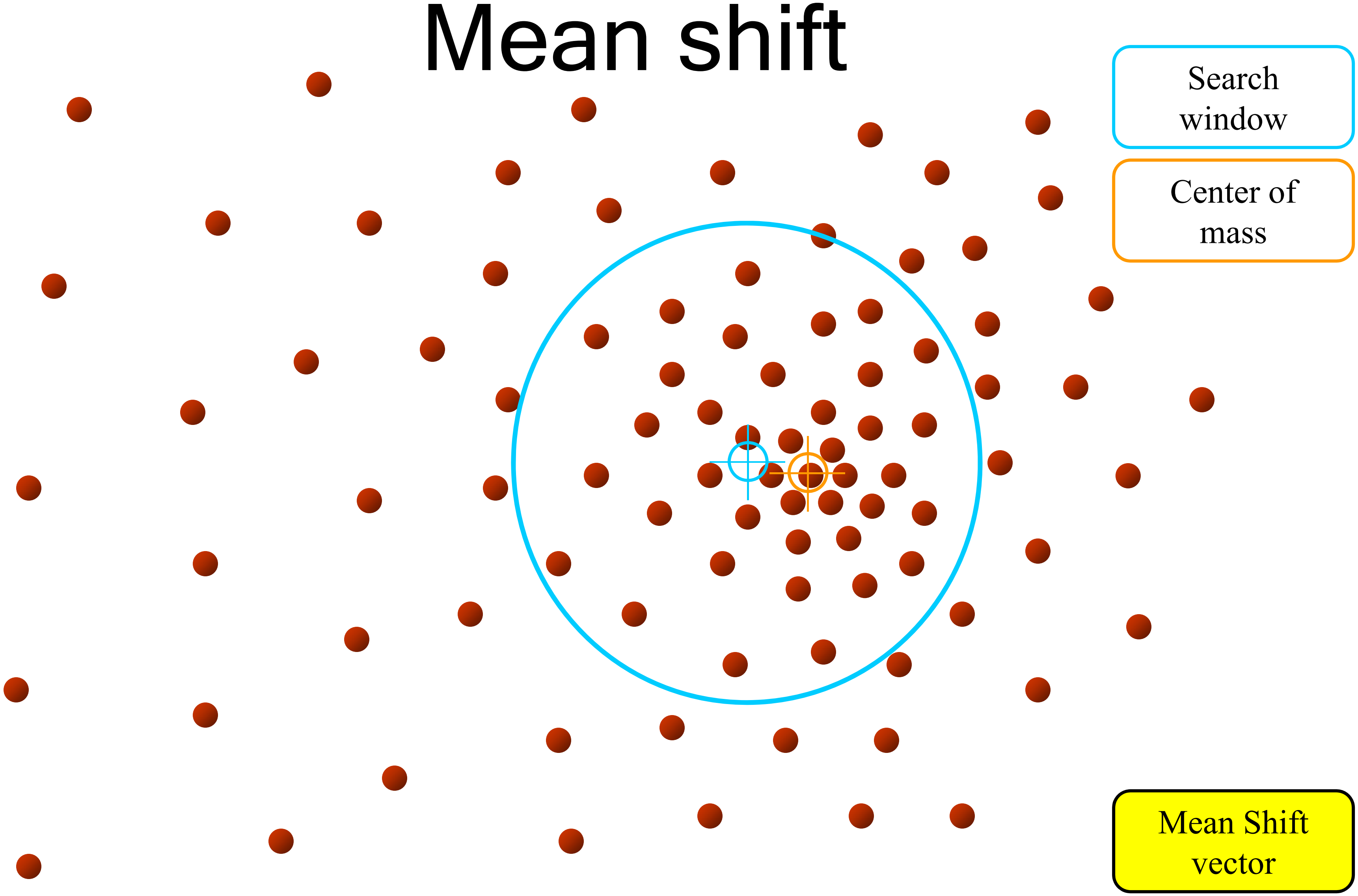
Search window

Center of mass

Mean Shift vector

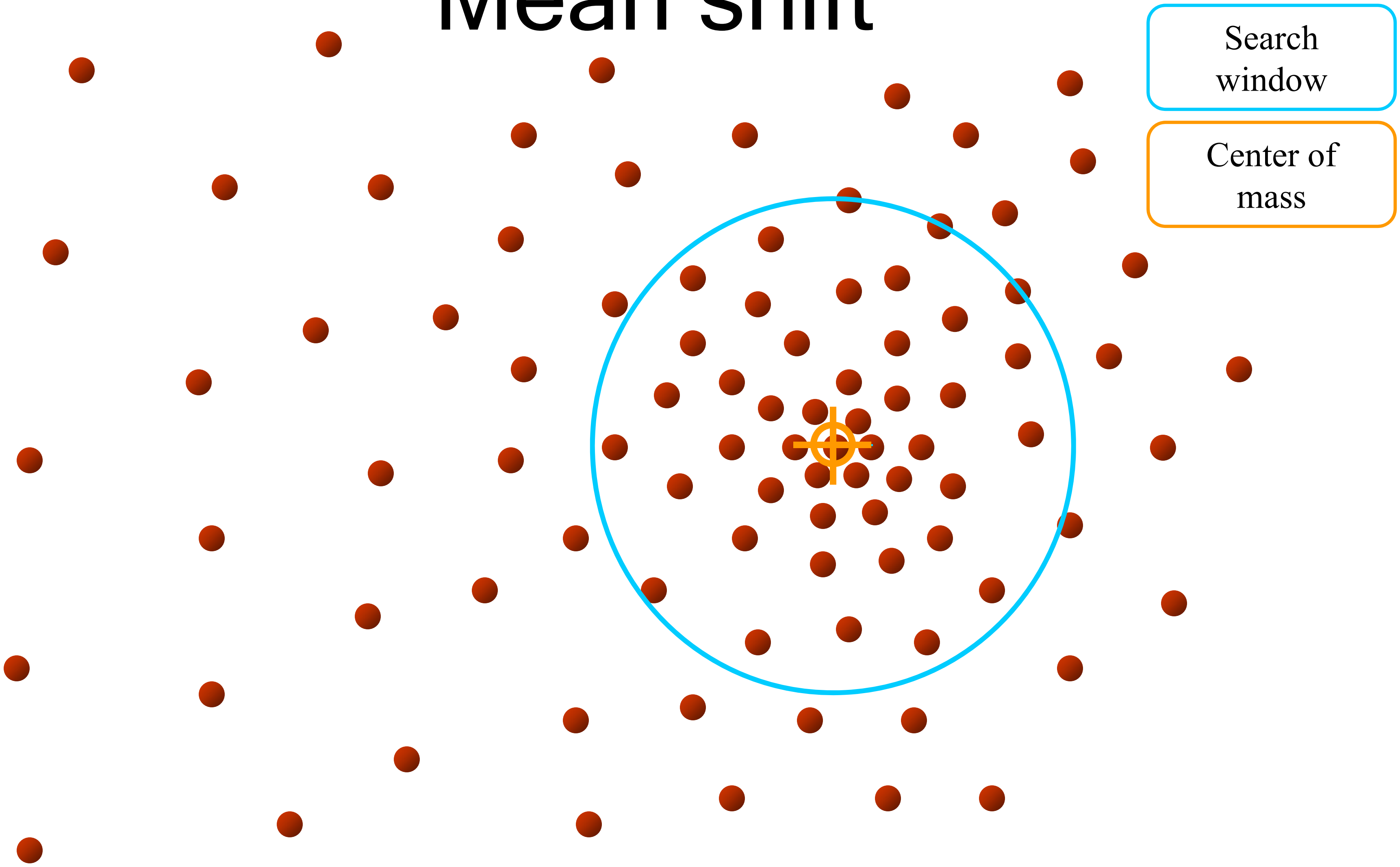
Slide by Y. Ukrainitz & B. Sarel

Mean shift



Slide by Y. Ukrainitz & B. Sarel

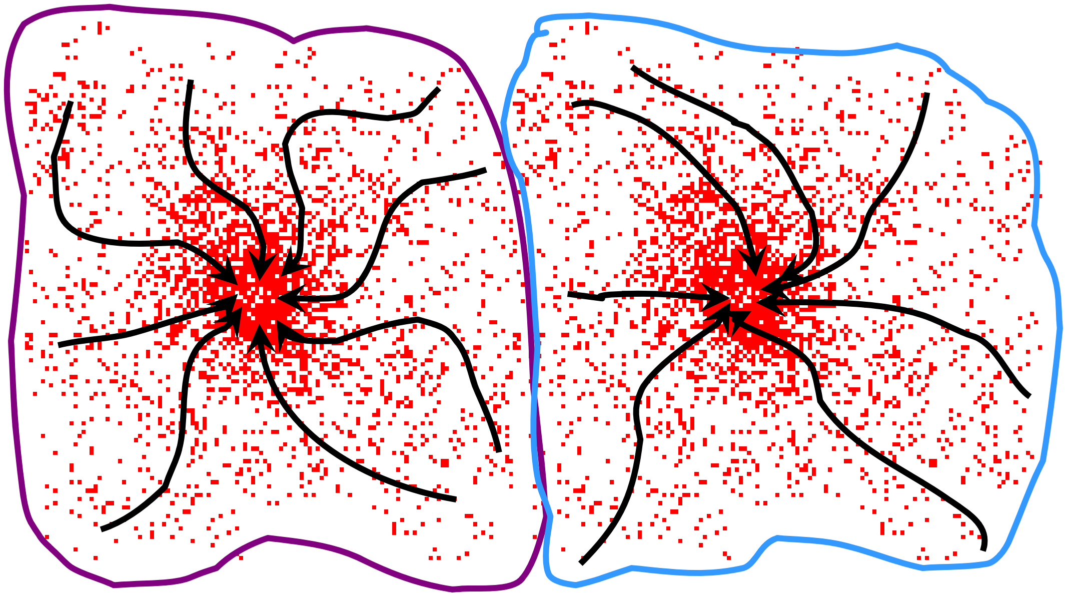
Mean shift



Slide by Y. Ukrainitz & B. Sarel

Mean shift clustering

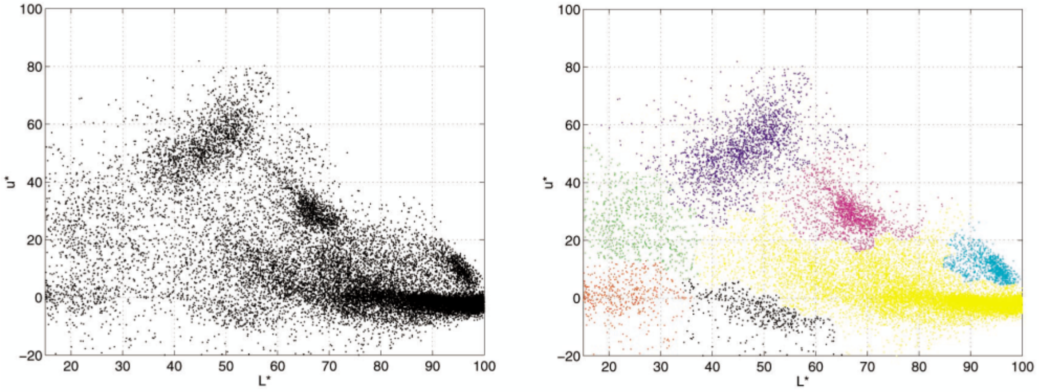
- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode



Slide by Y. Ukrainitz & B. Sarel

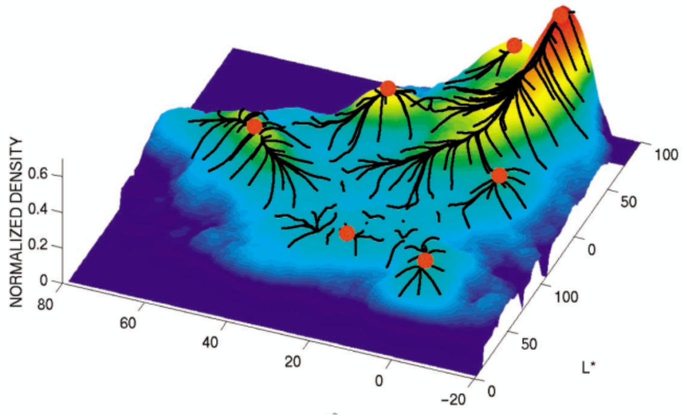
Mean shift clustering/segmentation

- Find features (color, gradients, texture, etc)
- Initialize windows at individual feature points
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode

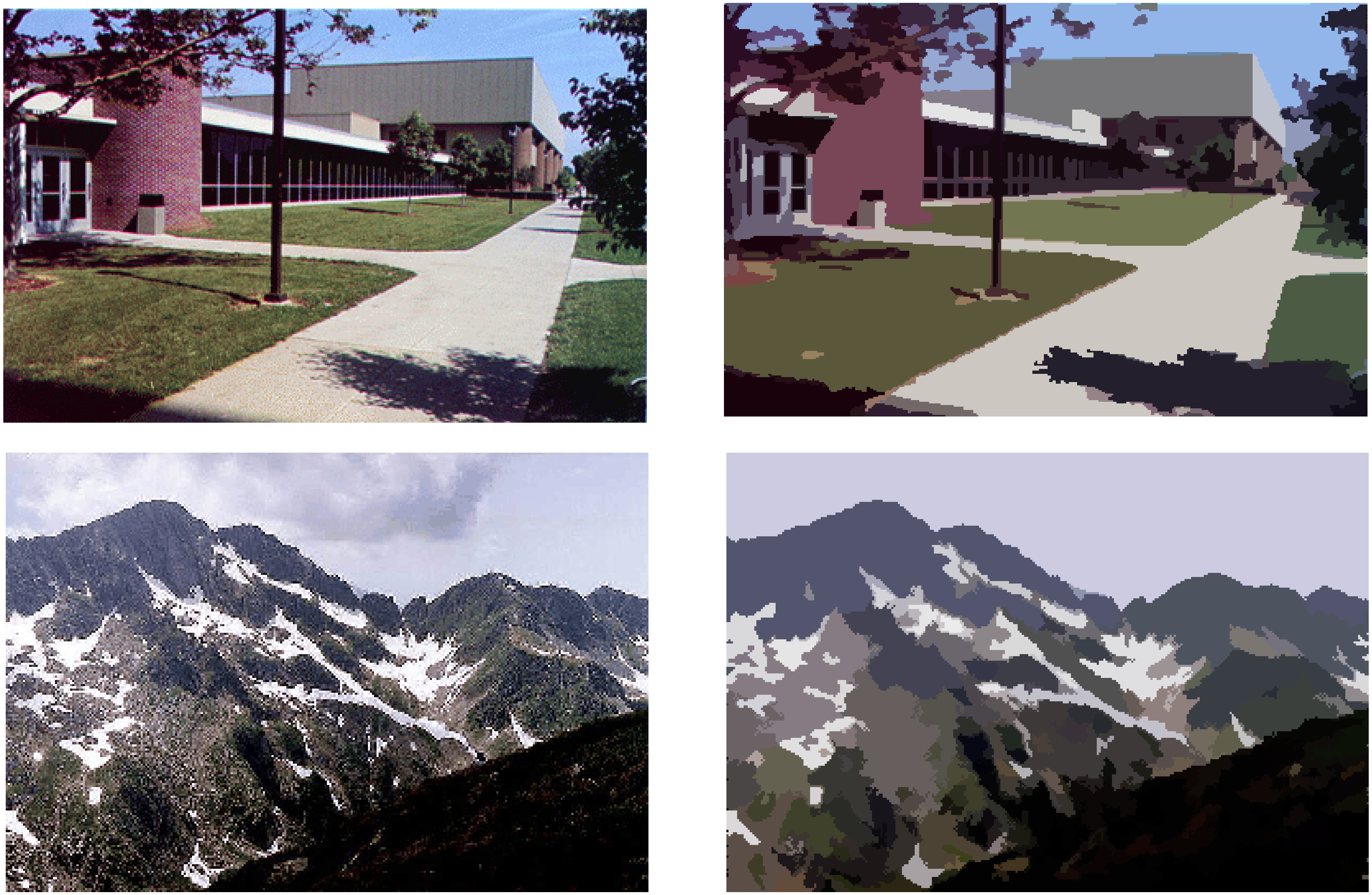


(a)

(b)

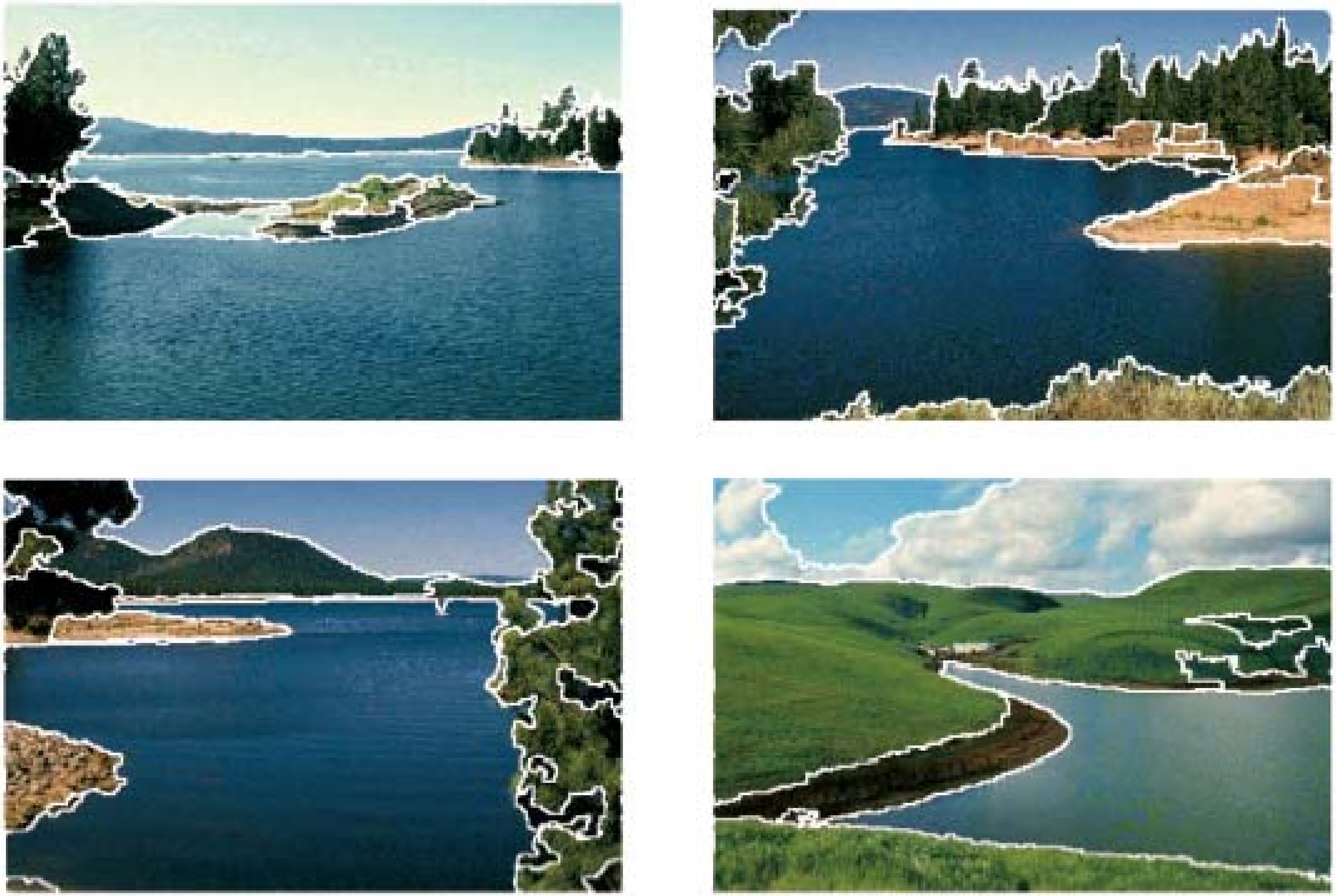


Mean shift segmentation results



http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html

Mean shift segmentation results



Mean shift

- Pros:
  - Does not assume shape on clusters
  - One parameter choice (bandwidth/window size)
  - Generic technique
  - Finds multiple modes
- Cons:
  - Selection of bandwidth
  - Does not scale well with dimension of feature space

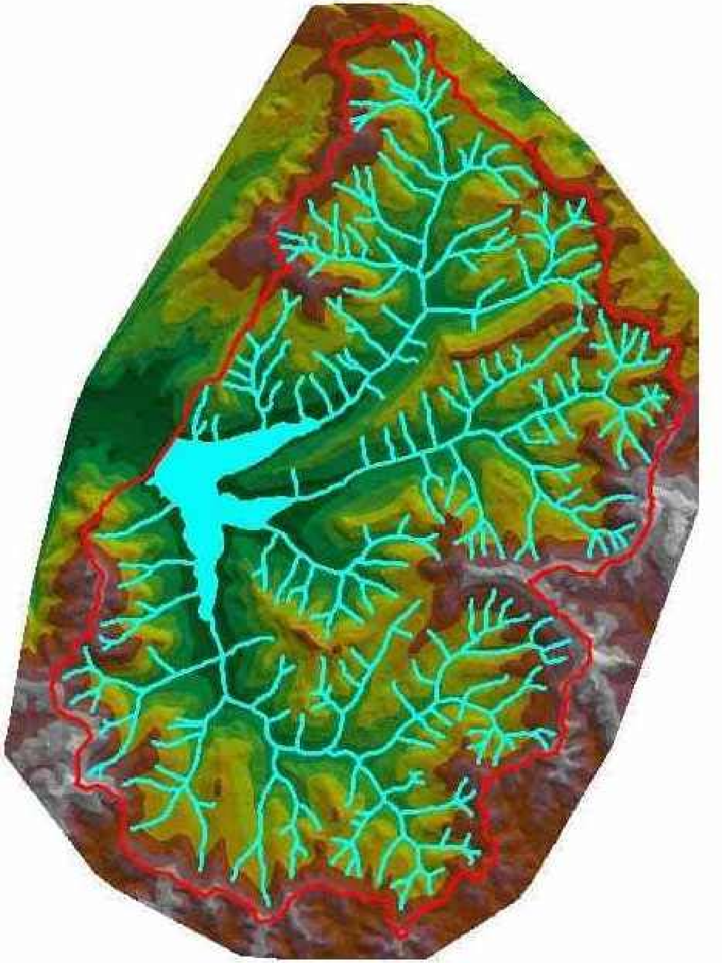
Kristen Grauman

Superpixel algorithms

- Goal is to divide the image into a large number of regions, such that each regions lie within object boundaries

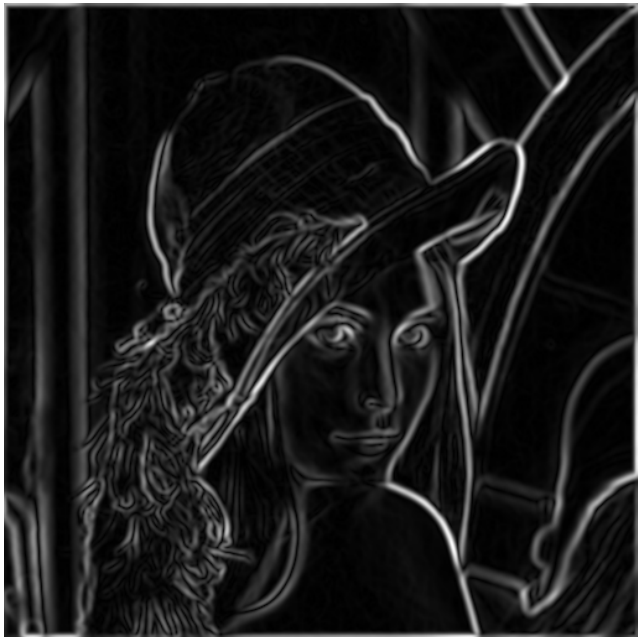
- Examples
  - Watershed
  - Felzenszwalb and Huttenlocher graph-based
  - Turbopixels
  - SLIC

Watershed algorithm

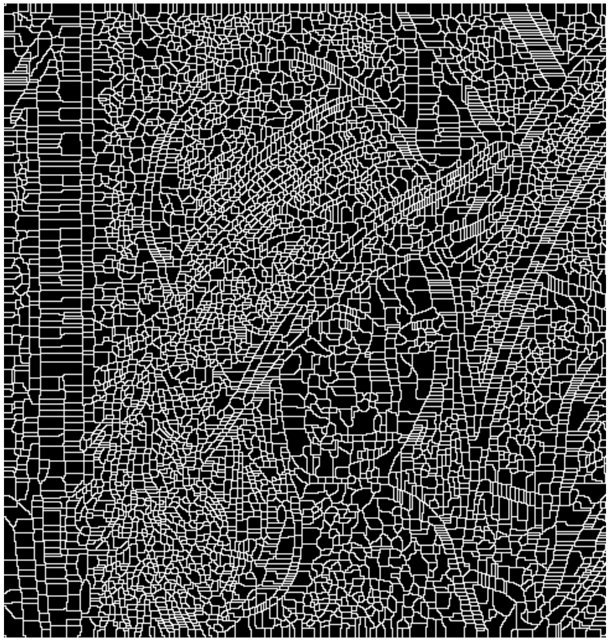


Watershed segmentation

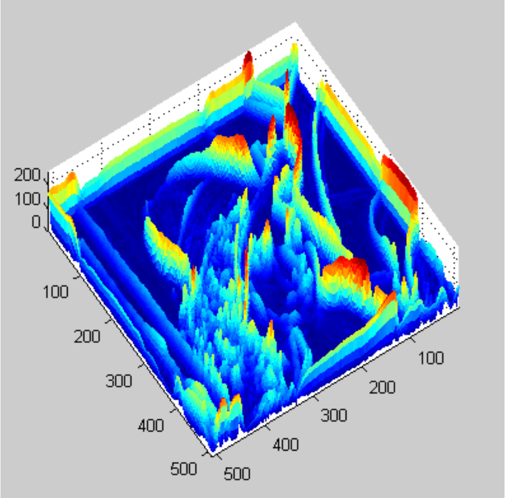
Image



Gradient



Watershed boundaries



Meyer’s watershed segmentation

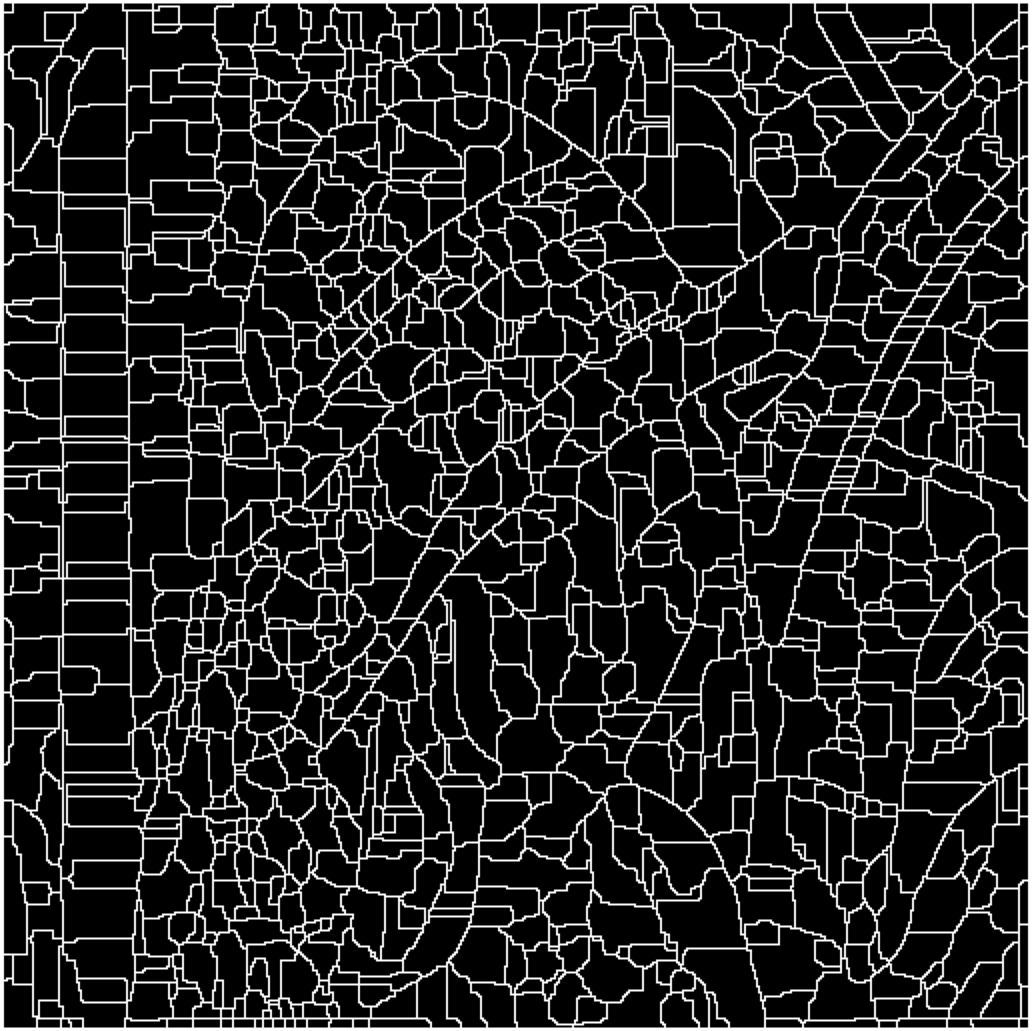
1. Choose local minima as region seeds
2. Add neighbors to priority queue, sorted by value
3. Take top priority pixel from queue
    1. If all labeled neighbors have same label, assign that label to pixel
    2. Add all non-marked neighbors to queue
4. Repeat step 3 until finished (all remaining pixels in queue are on the boundary)

Matlab: seg = watershed(bnd_im)

Meyer 1991

Simple trick

- Use Gaussian or median filter to reduce number of regions



Watershed pros and cons

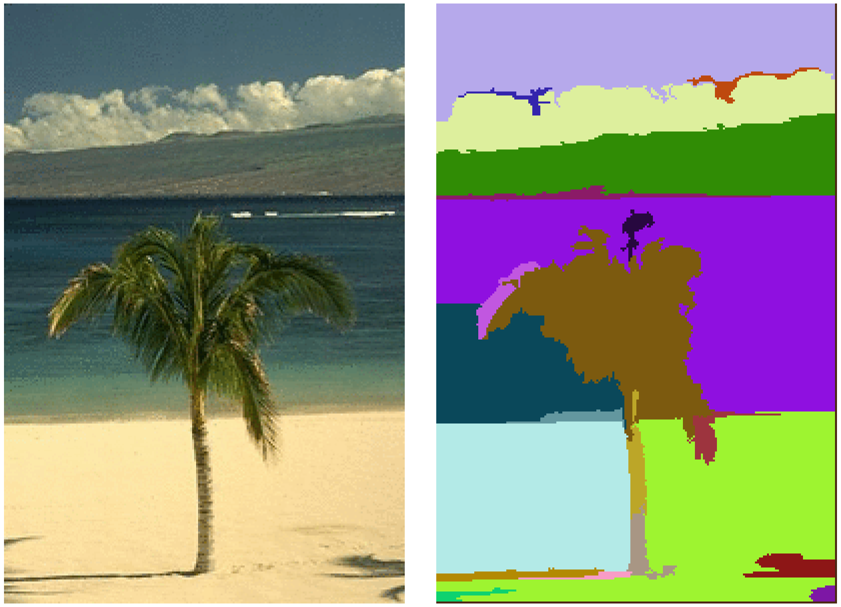
- Pros
  - Fast (< 1 sec for 512x512 image)
  - Preserves boundaries

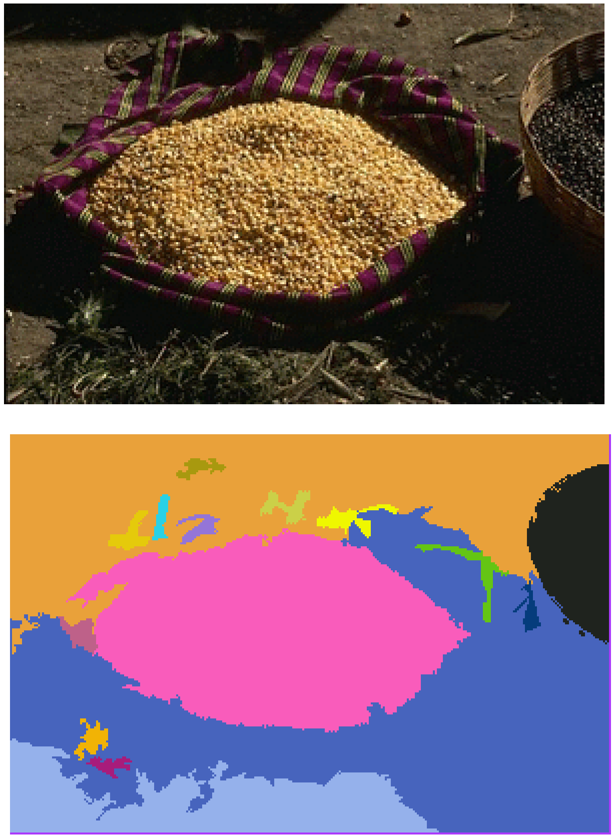
- Cons
  - Only as good as the soft boundaries (which may be slow to compute)
  - Not easy to get variety of regions for multiple segmentations

- Usage
  - Good algorithm for superpixels, hierarchical segmentation

Felzenszwalb and Huttenlocher: Graph-Based Segmentation

http://www.cs.brown.edu/~pff/segment/



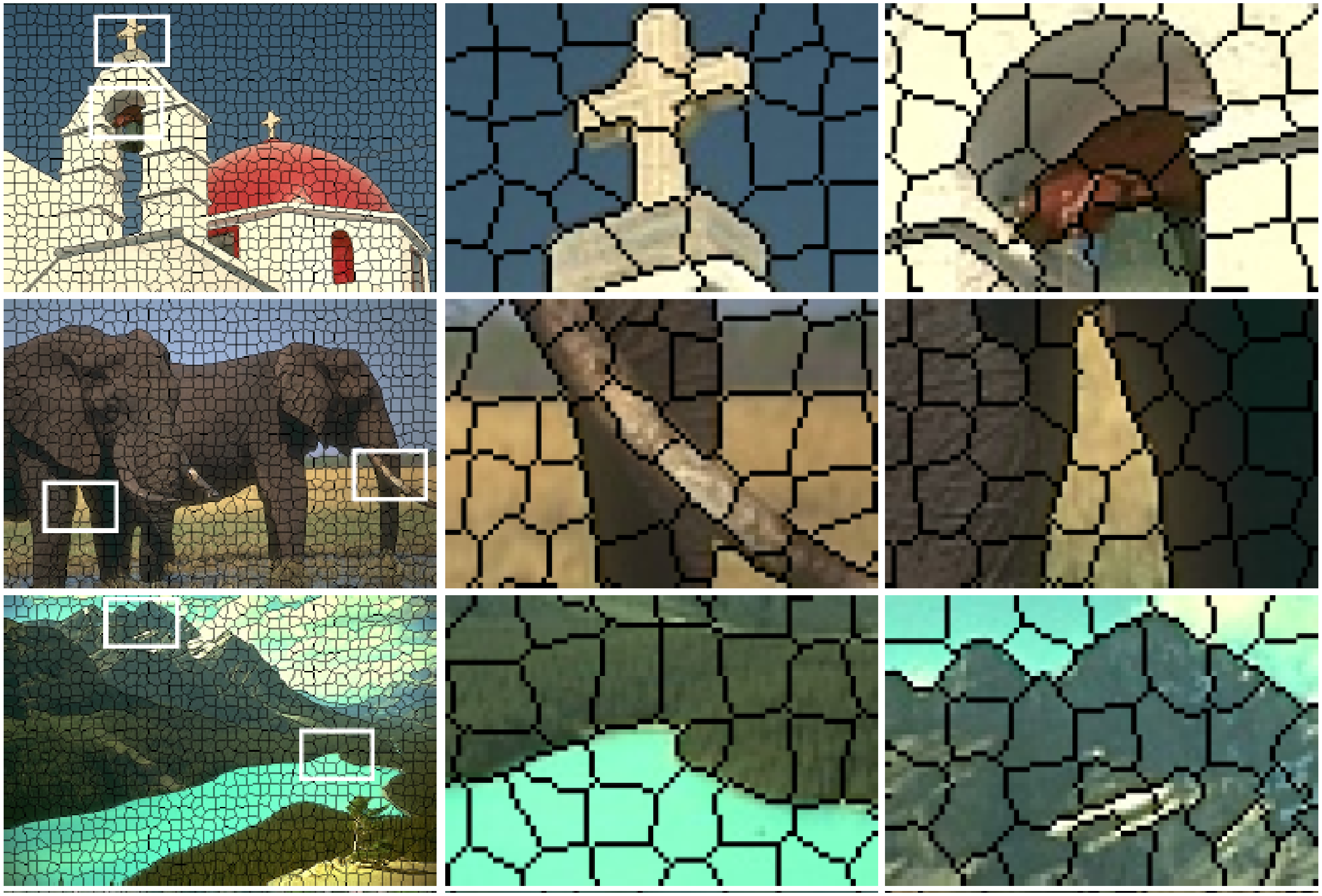


+ Good for thin regions
+ Fast
+ Easy to control coarseness of segmentations
+ Can include both large and small regions
- Often creates regions with strange shapes
- Sometimes makes very large errors

Turbo Pixels: Levinstein et al. 2009

http://www.cs.toronto.edu/~kyros/pubs/09.pami.turbopixels.pdf

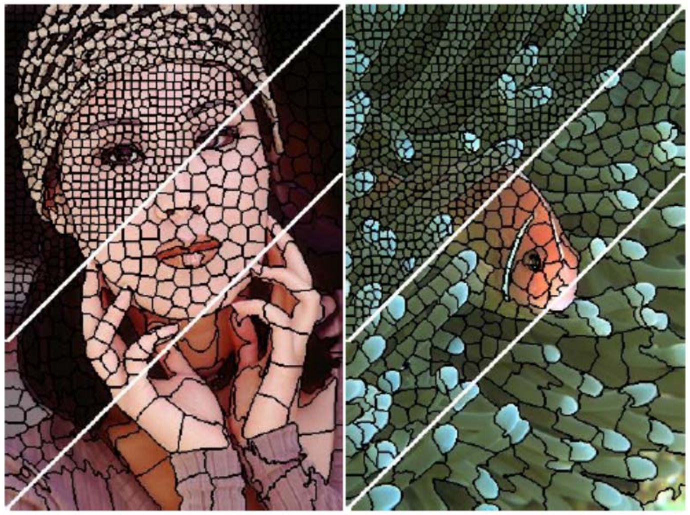
Tries to preserve boundaries like watershed but to produce more regular regions



SLIC (Achanta et al. PAMI 2012)

http://infoscience.epfl.ch/record/177415/files/Superpixel_PAMI2011-2.pdf

1. Initialize cluster centers on pixel grid in steps S
   - Features: Lab color, x-y position
2. Move centers to position in 3x3 window with smallest gradient
3. Compare each pixel to cluster center within 2S pixel distance and assign to nearest
4. Recompute cluster centers as mean color/position of pixels belonging to each cluster
5. Stop when residual error is small



+ Fast 0.36s for 320x240
+ Regular superpixels
+ Superpixels fit boundaries
- May miss thin objects
- Large number of superpixels

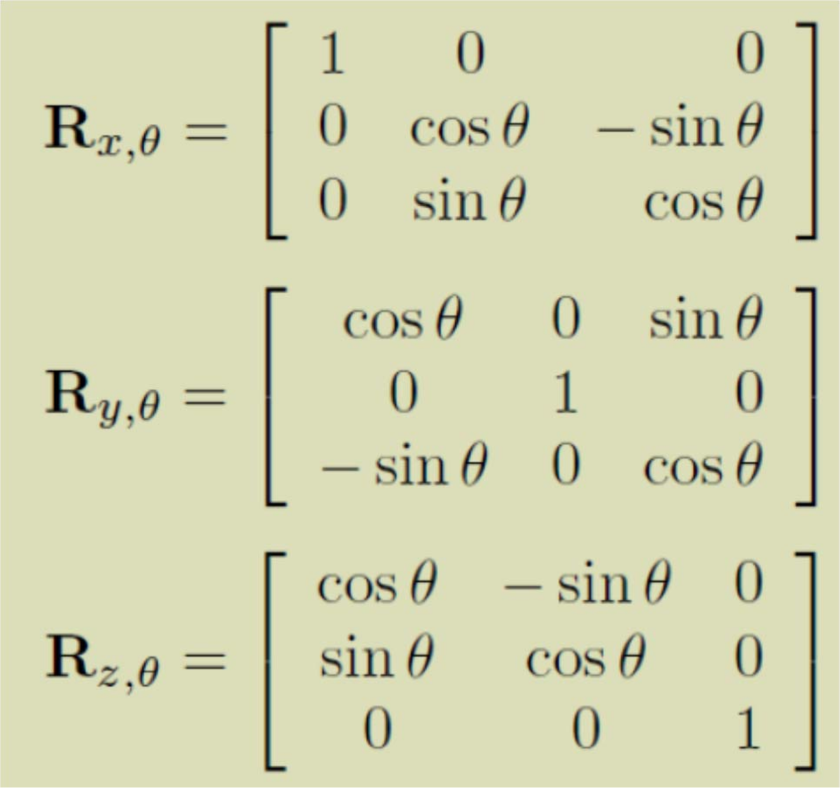
Introduction to Geometry

Based on slides by M. Pollefeys (ETH)
and D. Cappelleri (Purdue)

Rotations

- Rotation matrices around the 3 axes

=> What is the inverse of a rotation matrix?

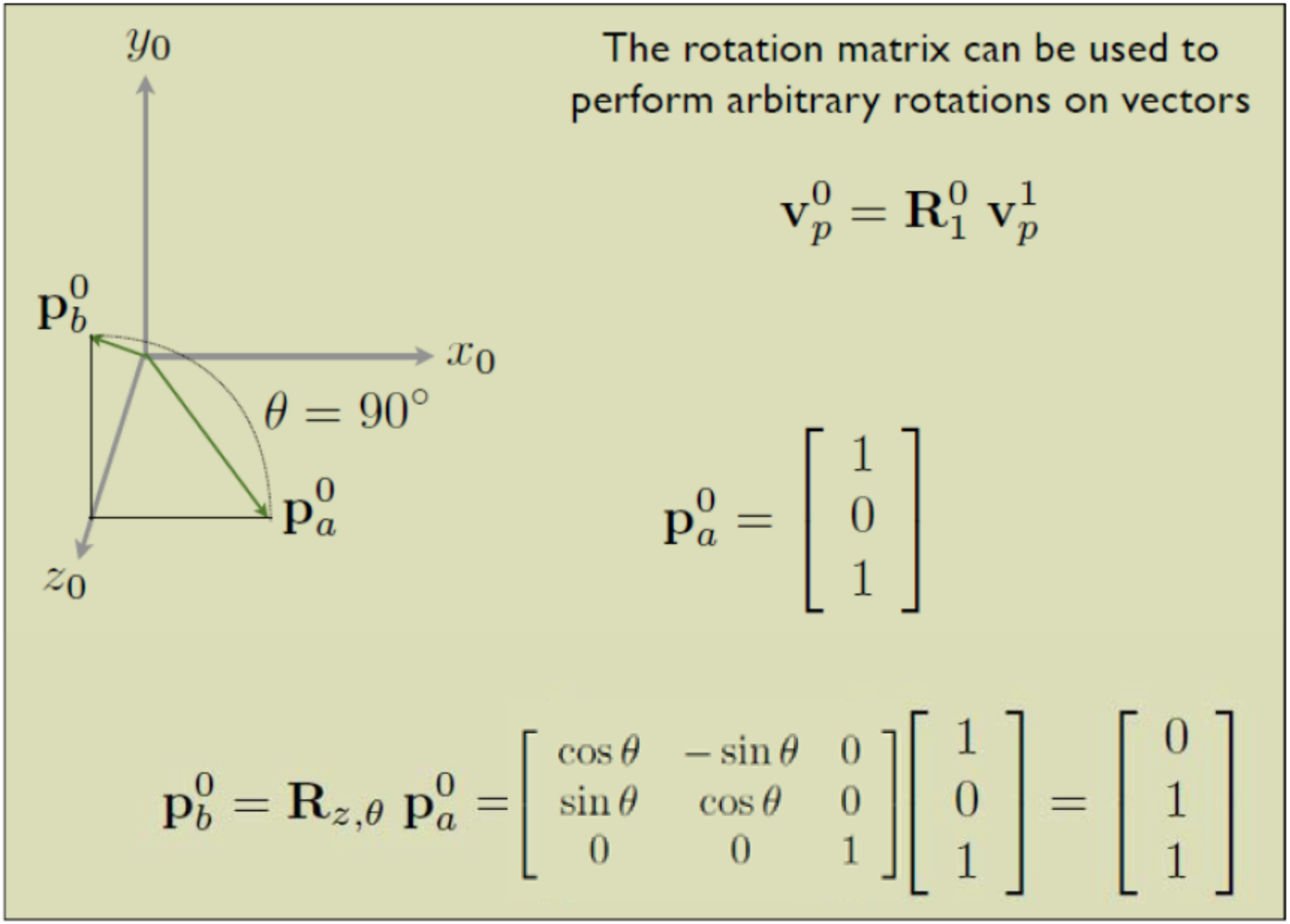


$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$\mathbf{R}_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation Example



The rotation matrix can be used to perform arbitrary rotations on vectors

$$\mathbf{v}_p^0 = \mathbf{R}_1^0 \ \mathbf{v}_p^1$$

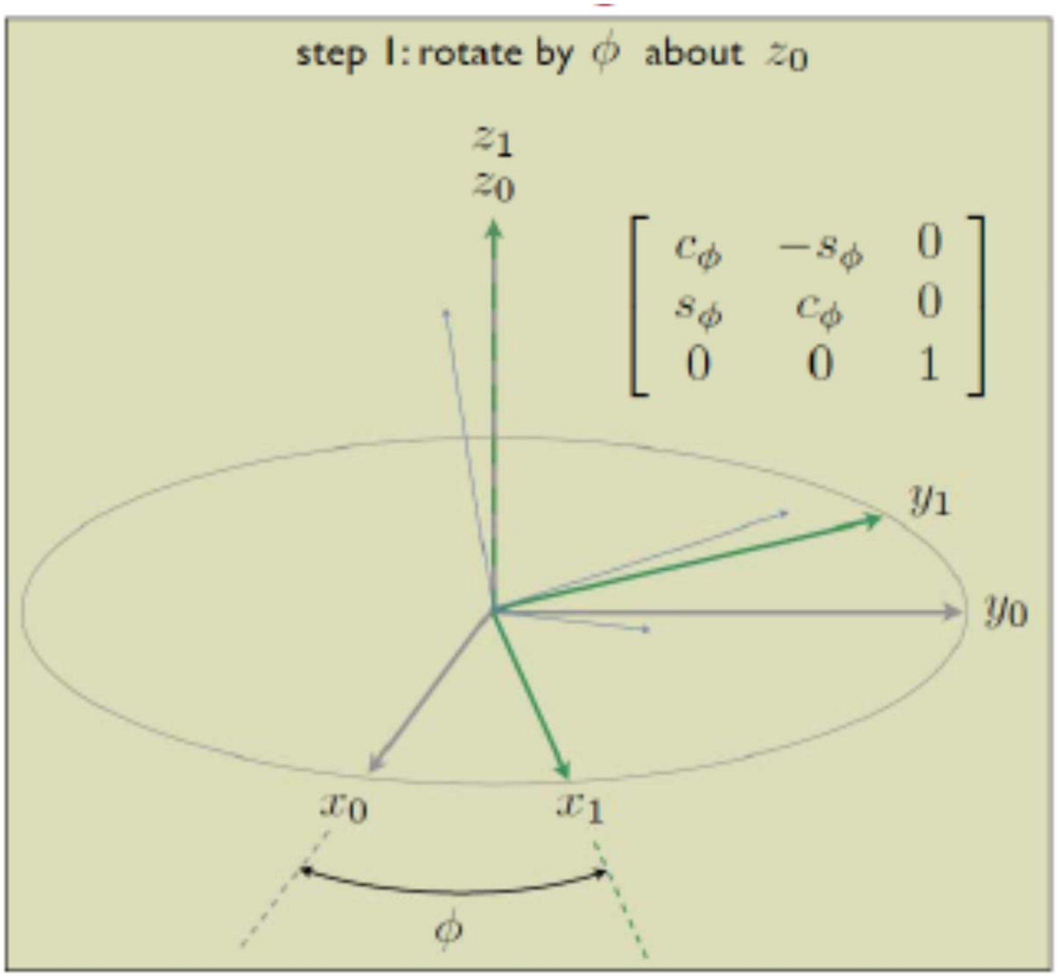
$$\mathbf{p}_a^0 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\mathbf{p}_b^0 = \mathbf{R}_{z,\theta} \ \mathbf{p}_a^0 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

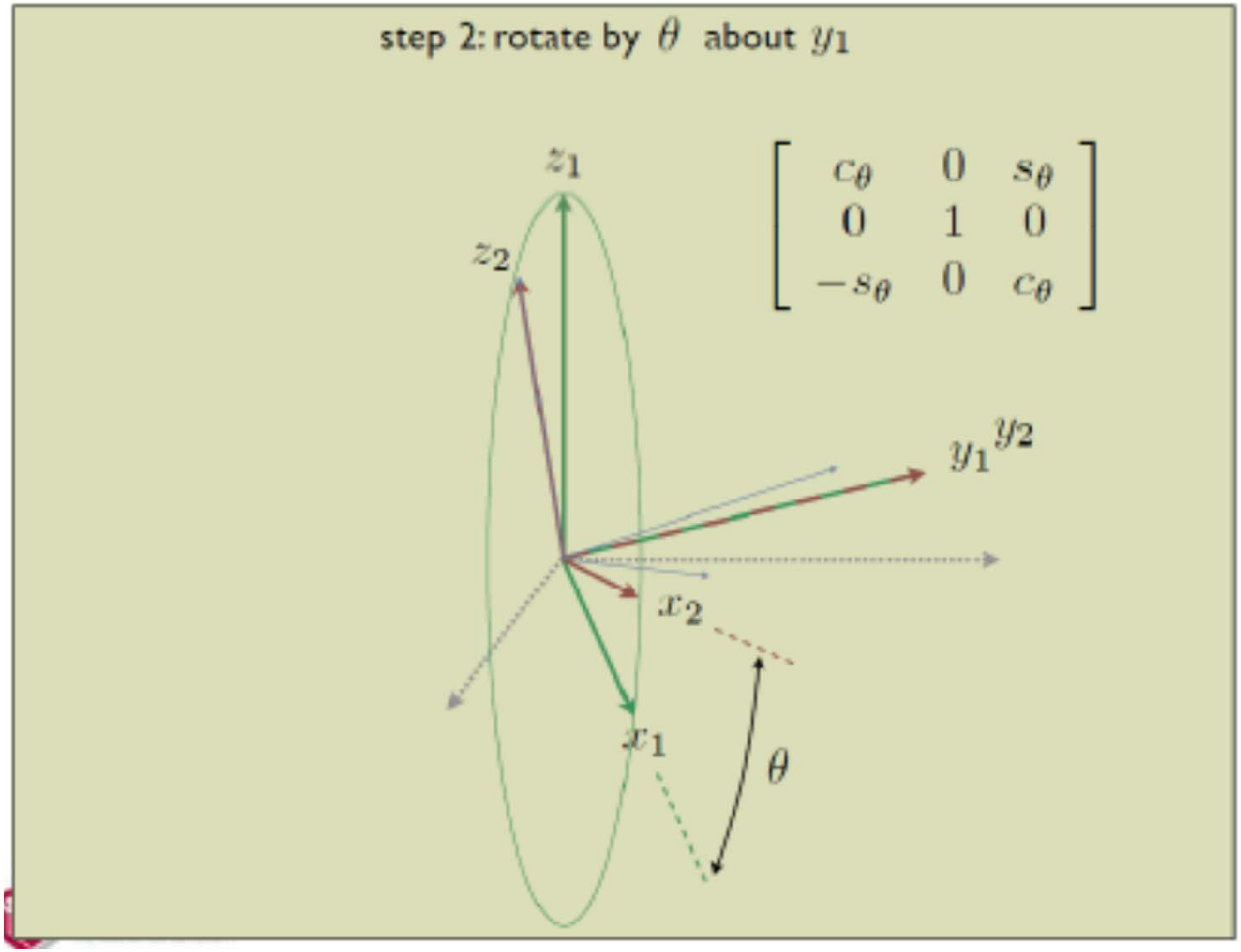
Parameterization of Rotations

- In 3D, the 9-element rotation matrix has 3 DOF
- Several methods exist for representing a 3D rotation
  - Euler angles
  - Pitch, Roll, Yaw angles
  - Axis/Angle representation
  - Quaternions

Euler Angles


step 1: rotate by ϕ about z_0
z_1
z_0
$\begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$
y_1
y_0
x_0
x_1
ϕ

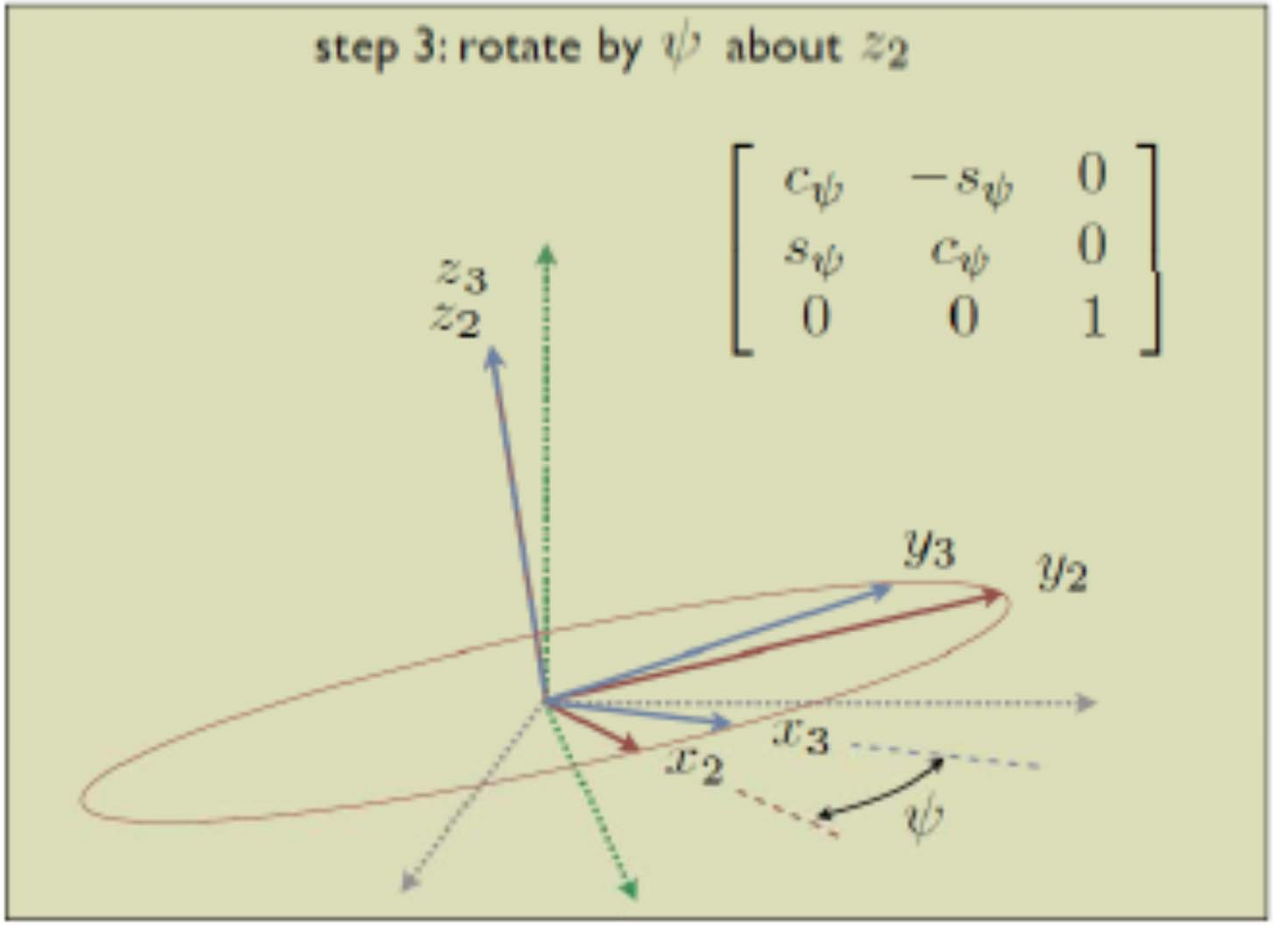
Euler Angles



step 2: rotate by θ about y_1

$$\begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}$$

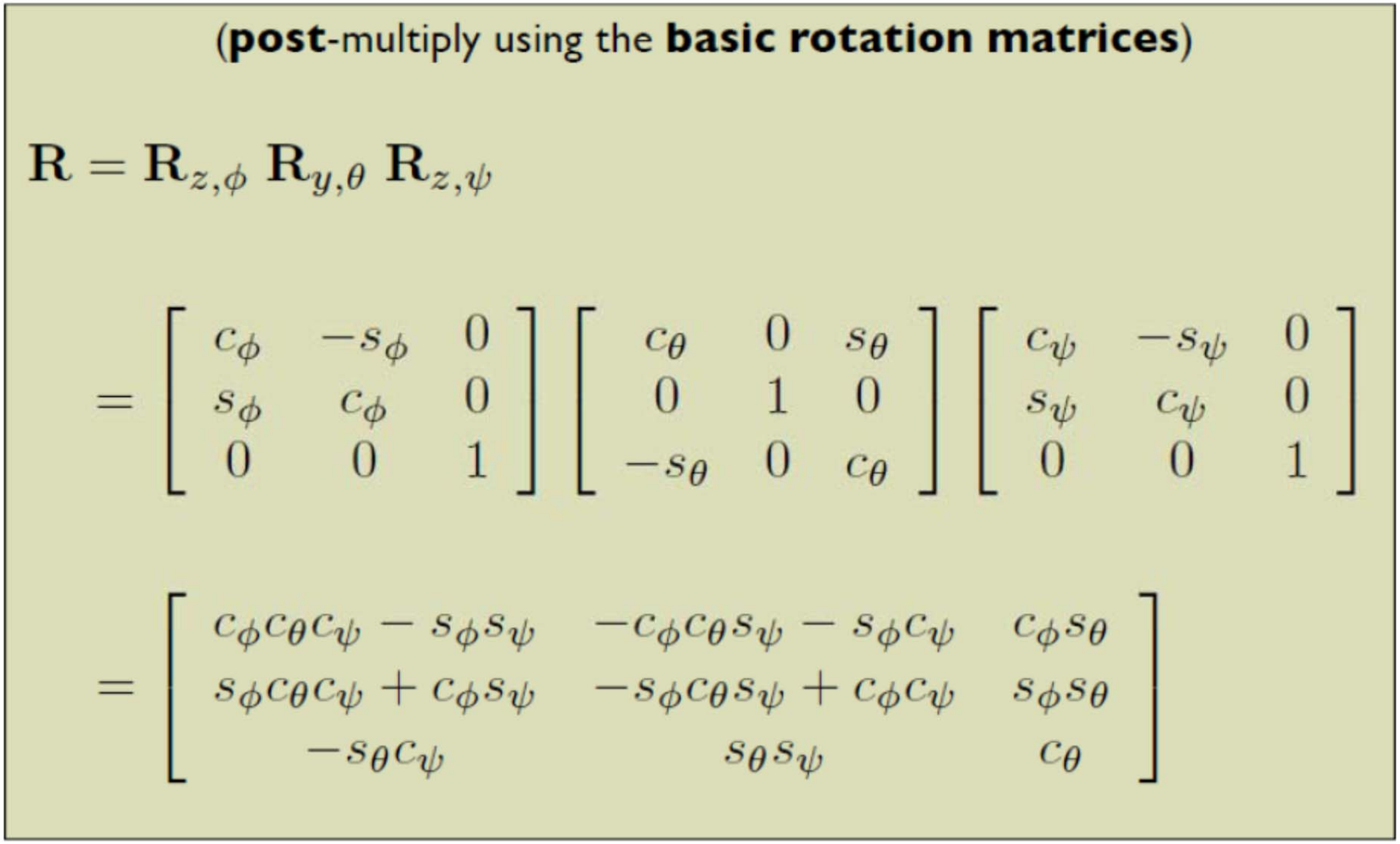
Euler Angles



step 3: rotate by ψ about z_2

$$\begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Euler Angles to Rotation Matrix



(**post**-multiply using the **basic rotation matrices**)

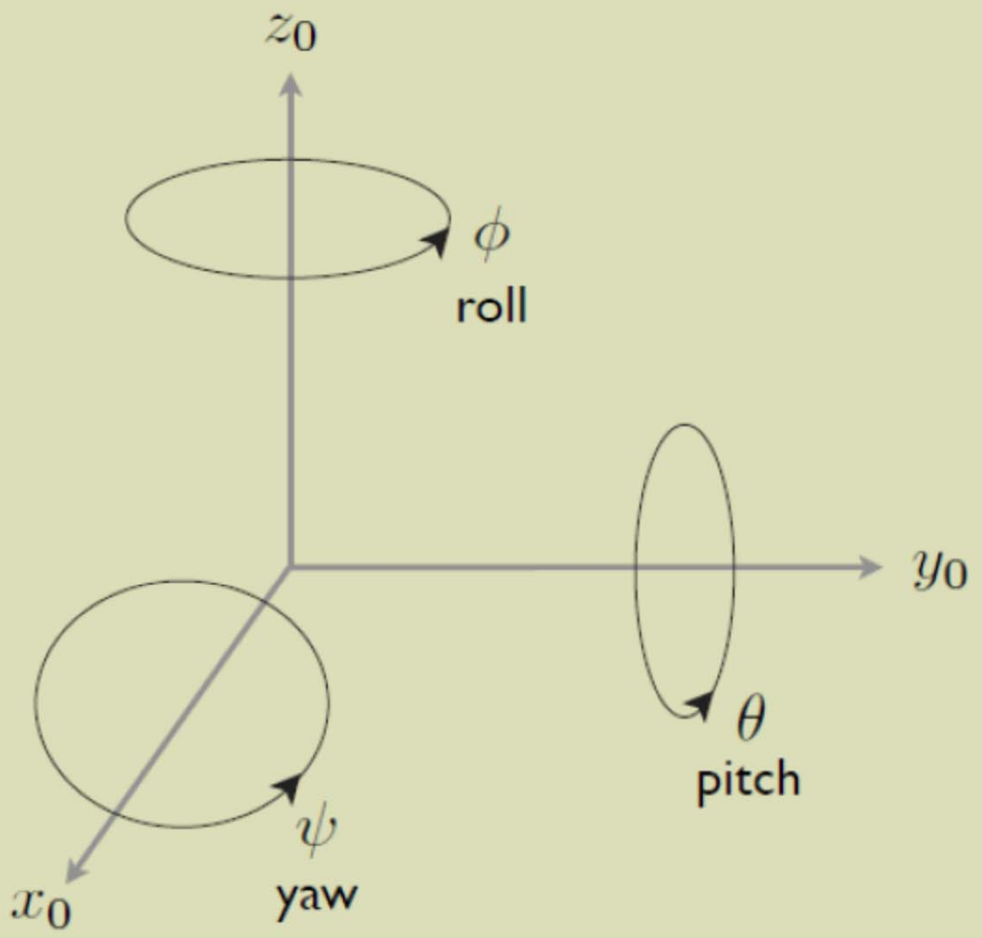
$$\mathbf{R} = \mathbf{R}_{z,\phi}\,\mathbf{R}_{y,\theta}\,\mathbf{R}_{z,\psi}$$

$$= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

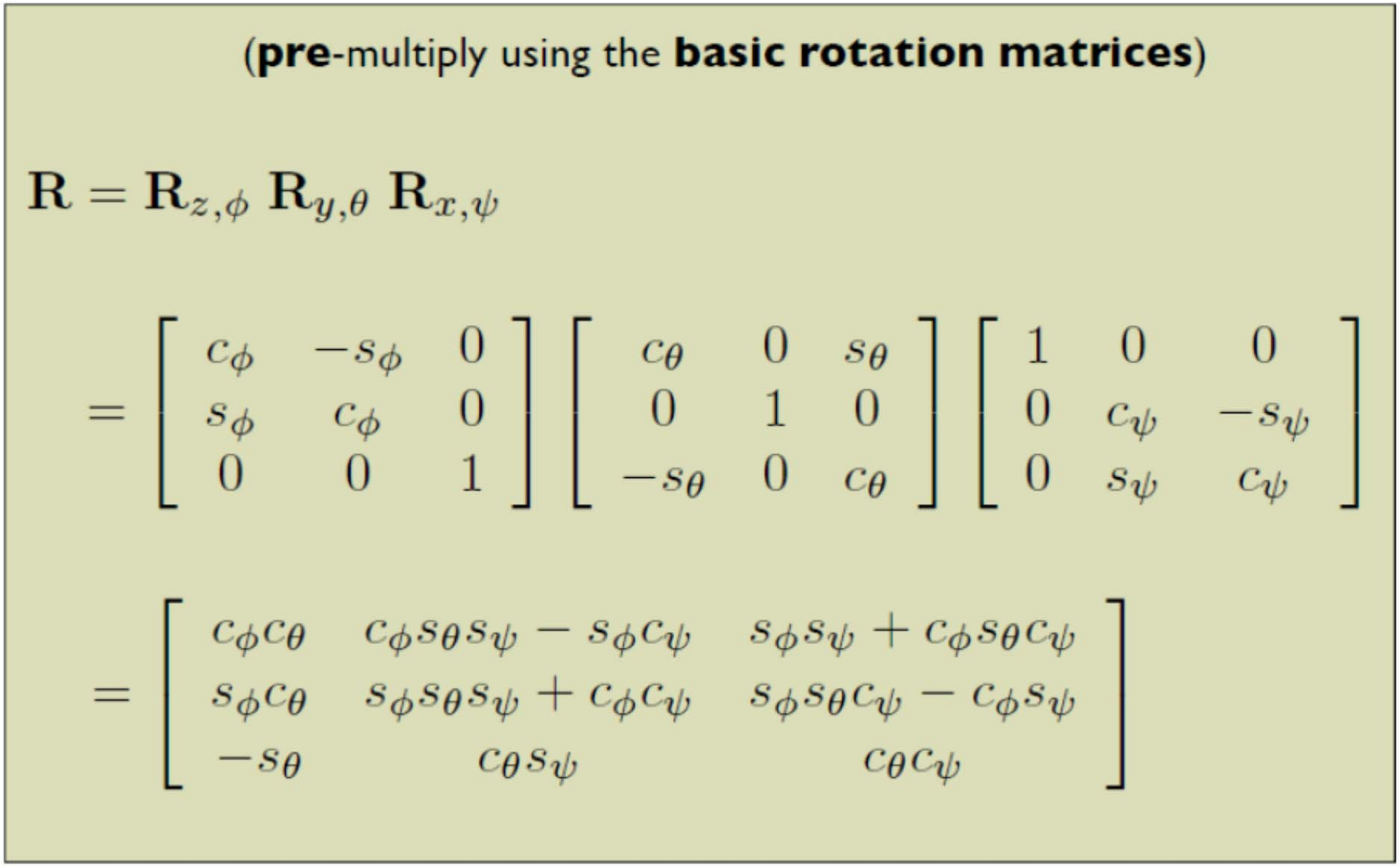
$$= \begin{bmatrix} c_\phi c_\theta c_\psi - s_\phi s_\psi & -c_\phi c_\theta s_\psi - s_\phi c_\psi & c_\phi s_\theta \\ s_\phi c_\theta c_\psi + c_\phi s_\psi & -s_\phi c_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta \\ -s_\theta c_\psi & s_\theta s_\psi & c_\theta \end{bmatrix}$$

Roll, Pitch, Yaw Angles

defined as a set of three angles about a **fixed** reference



Roll, Pitch, Yaw Angles to Rotation Matrix



(**pre**-multiply using the **basic rotation matrices**)

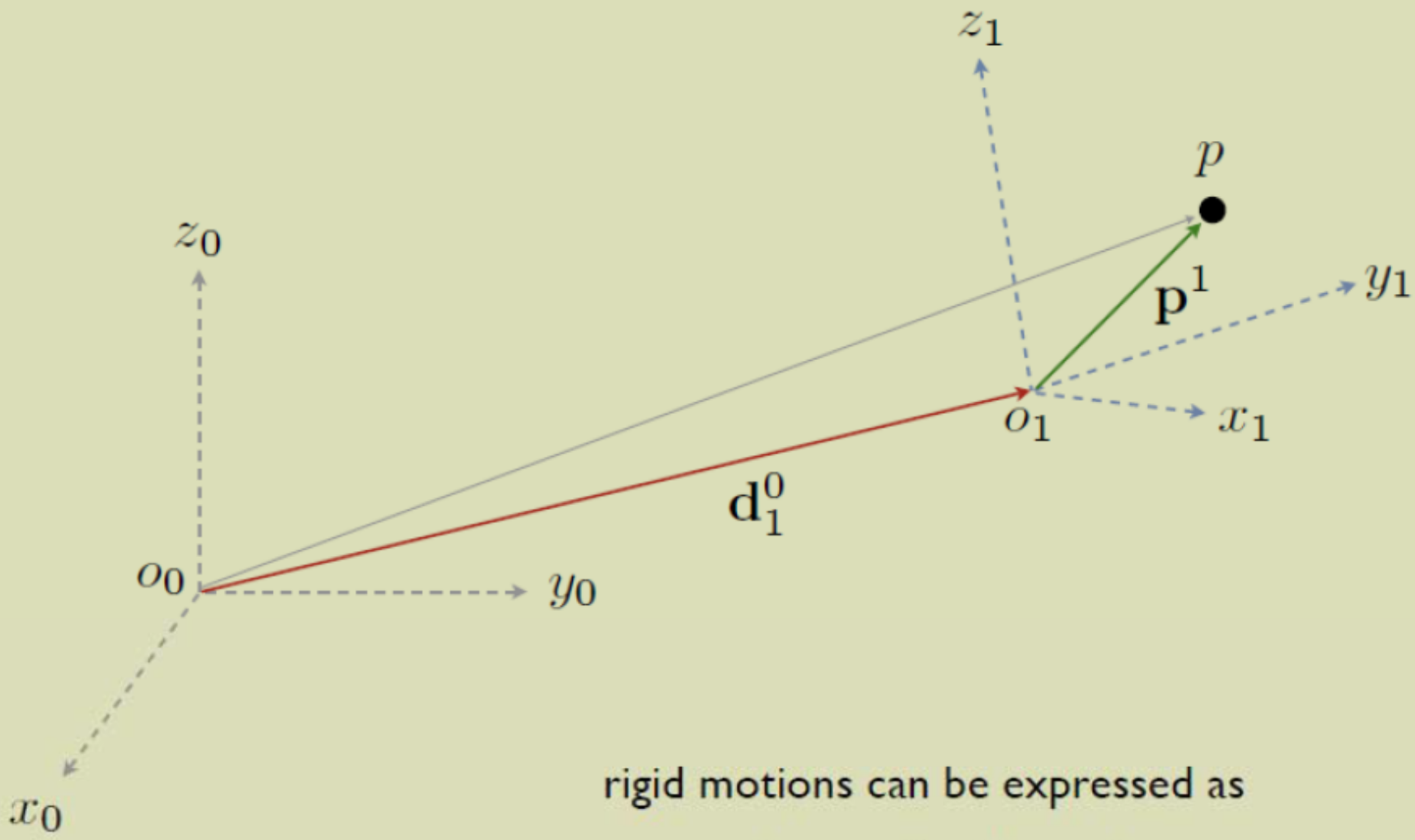
$$\mathbf{R} = \mathbf{R}_{z,\phi}\,\mathbf{R}_{y,\theta}\,\mathbf{R}_{x,\psi}$$

$$= \begin{bmatrix} c_\phi & -s_\phi & 0 \\ s_\phi & c_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\psi & -s_\psi \\ 0 & s_\psi & c_\psi \end{bmatrix}$$

$$= \begin{bmatrix} c_\phi c_\theta & c_\phi s_\theta s_\psi - s_\phi c_\psi & s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}$$

Rigid Motion

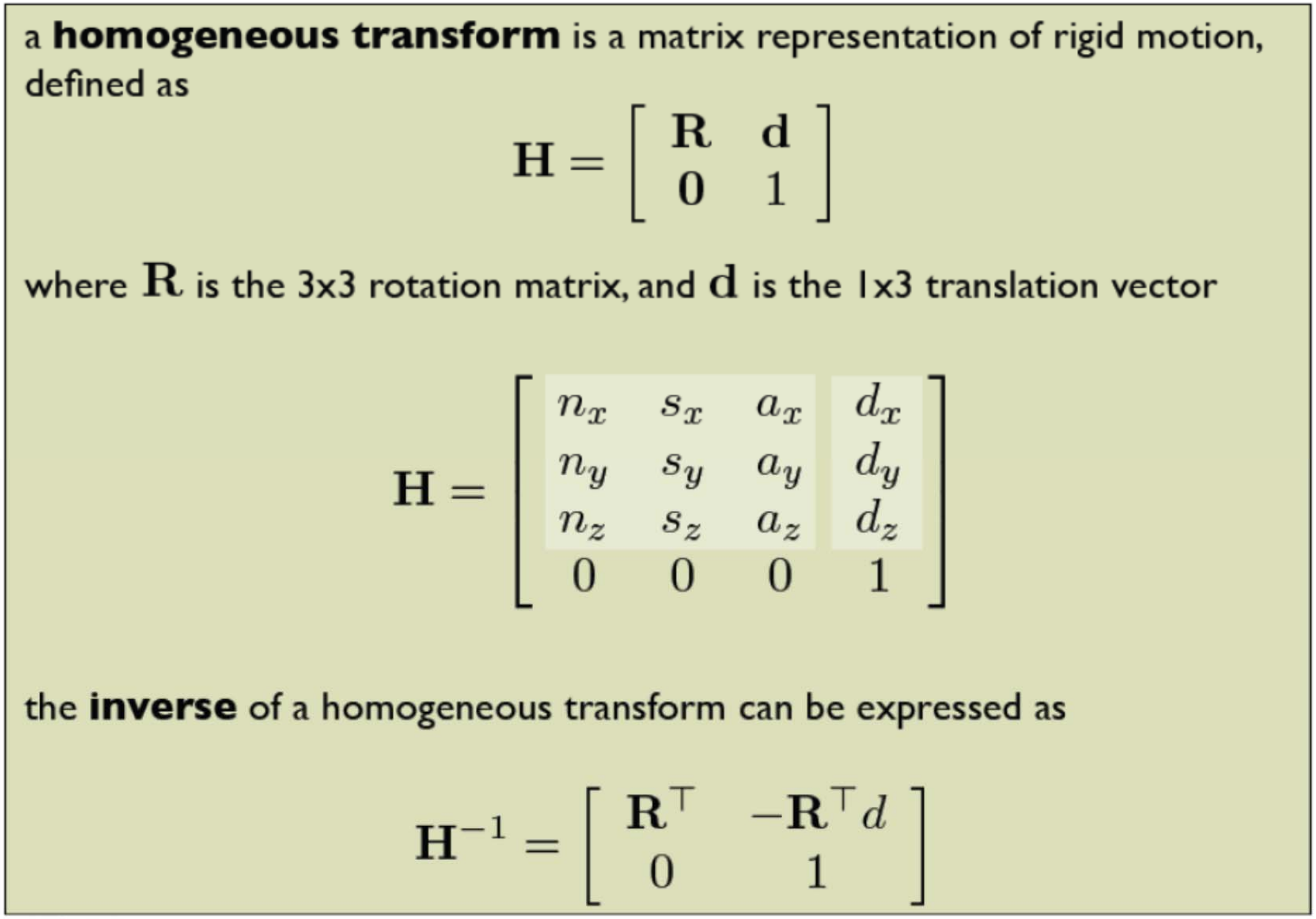
a **rigid motion** couples pure translation with pure rotation



rigid motions can be expressed as

$$\mathbf{p}^0 = \mathbf{R}_1^0\,\mathbf{p}^1 + \mathbf{d}_1^0$$

Homogeneous Transformation



a **homogeneous transform** is a matrix representation of rigid motion, defined as

$$\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix}$$

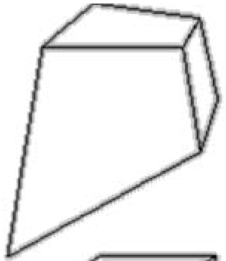
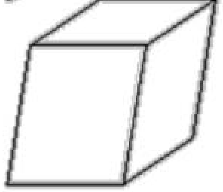
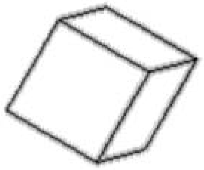
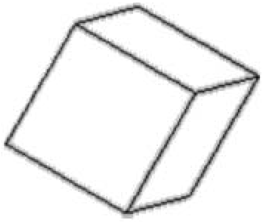
where $\mathbf{R}$ is the 3x3 rotation matrix, and $\mathbf{d}$ is the 1x3 translation vector

$$\mathbf{H} = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the **inverse** of a homogeneous transform can be expressed as

$$\mathbf{H}^{-1} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top d \\ 0 & 1 \end{bmatrix}$$

Hierarchy of 3D Transformations

Projective 15dof	$\begin{bmatrix} \mathrm{A} & \mathrm{t} \\ \mathrm{v}^{\mathrm{T}} & v \end{bmatrix}$		Intersection and tangency
Affine 12dof	$\begin{bmatrix} \mathrm{A} & \mathrm{t} \\ 0^{\mathrm{T}} & 1 \end{bmatrix}$		Parallellism of planes, Volume ratios, centroids, **The plane at infinity π_∞**
Similarity 7dof	$\begin{bmatrix} s\mathrm{R} & \mathrm{t} \\ 0^{\mathrm{T}} & 1 \end{bmatrix}$		Angles, ratios of length **The absolute conic Ω_∞**
Euclidean 6dof	$\begin{bmatrix} \mathrm{R} & \mathrm{t} \\ 0^{\mathrm{T}} & 1 \end{bmatrix}$		Volume

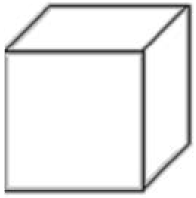
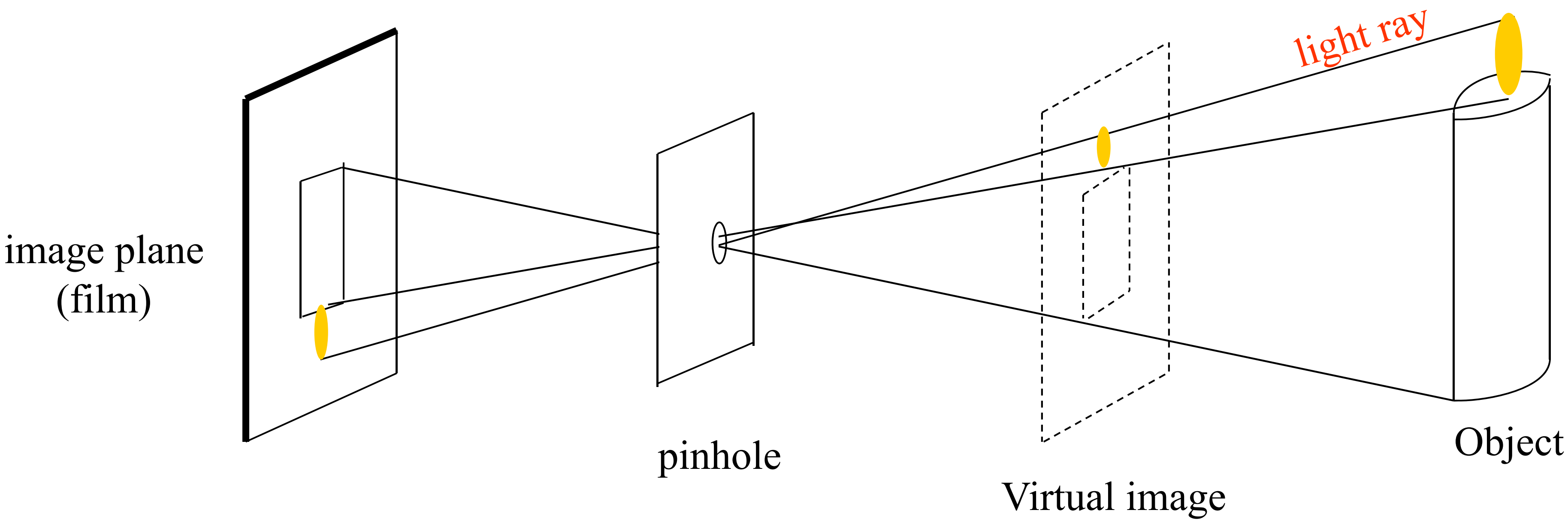


Image Formation

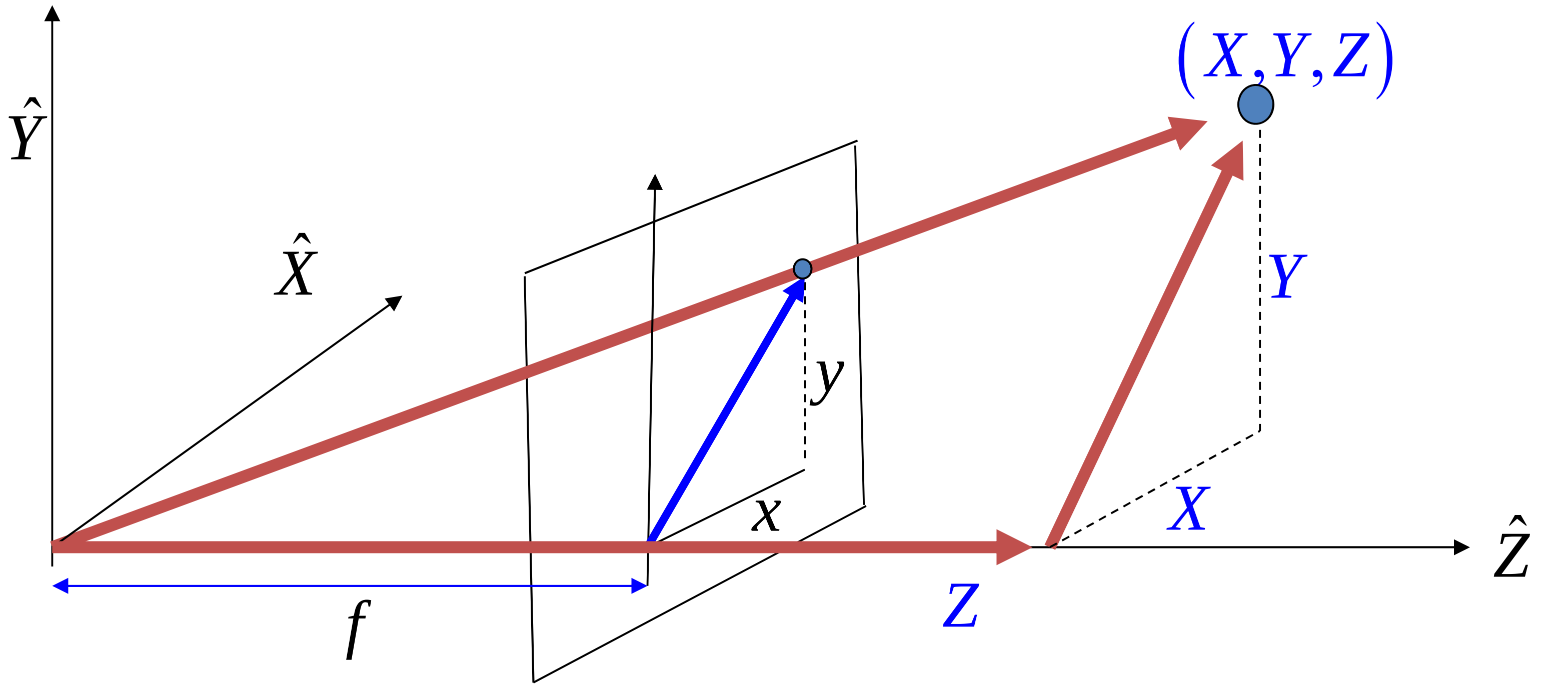
Based on slides by John Oliensis

Image Formation

Pinhole camera



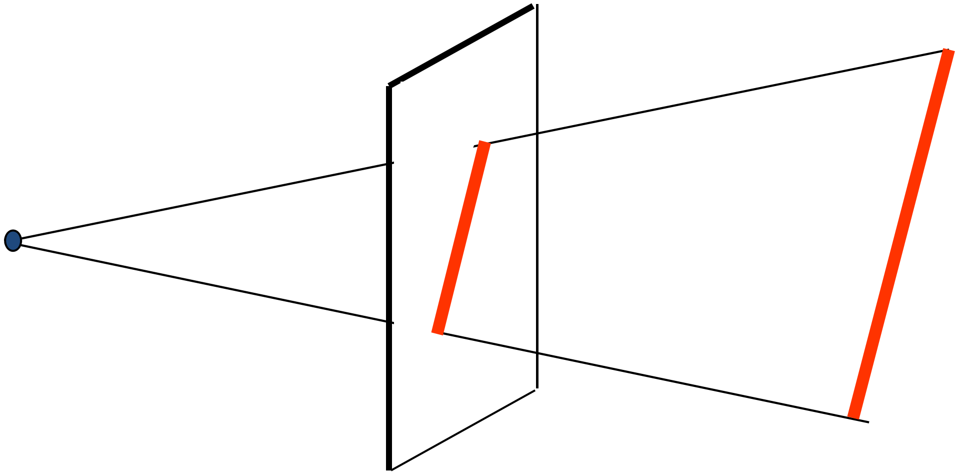
Projection Equation: 3D



Similar triangles: $\frac{x}{X} = \frac{y}{Y} = \frac{f}{Z}$ ⟶ $\boxed{(x,y) = \frac{f}{Z}(X,Y)}$

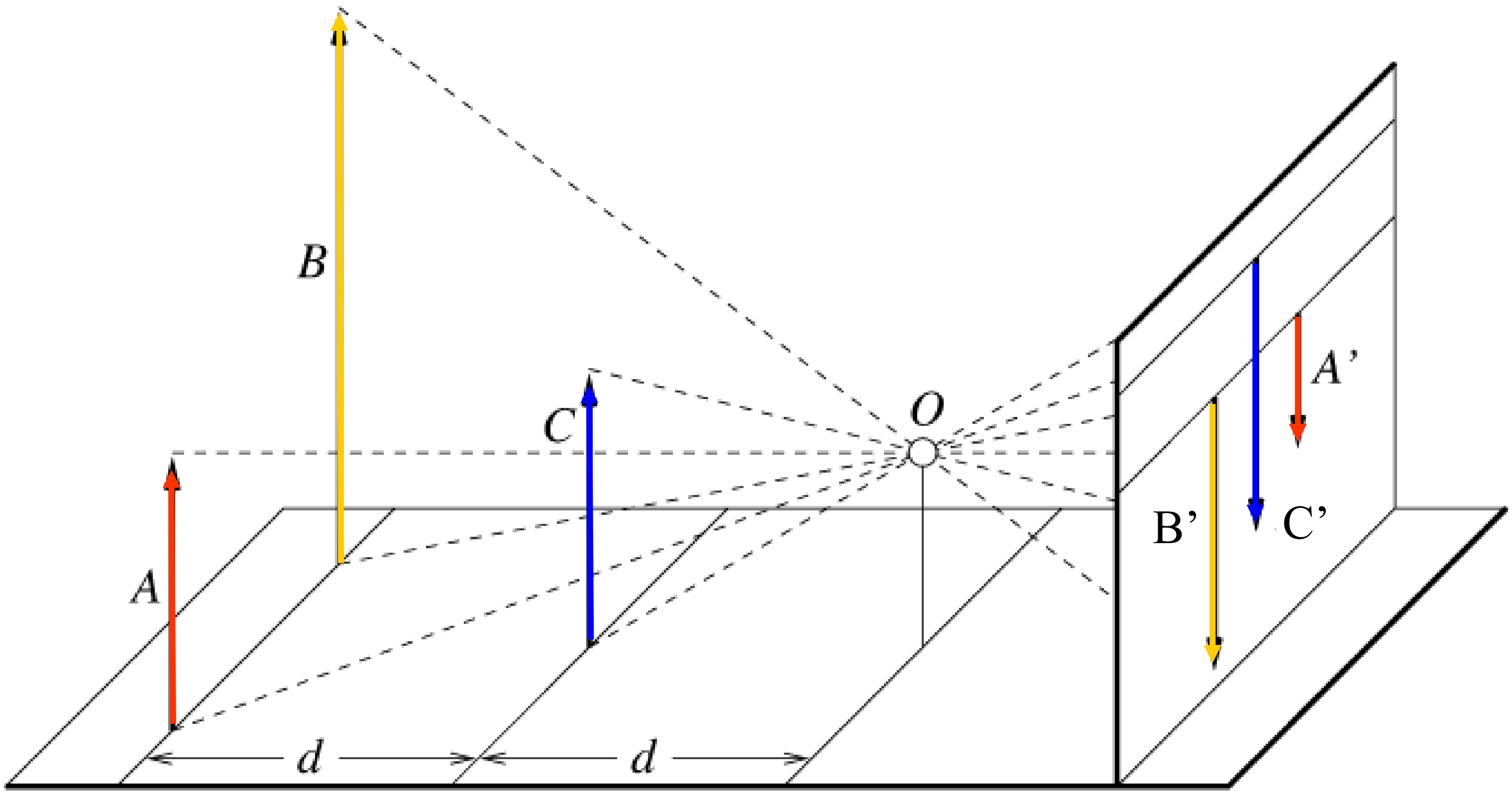
Perspective Projection: Properties

- 3D points ➔ image points
- 3D straight lines ➔ image straight lines



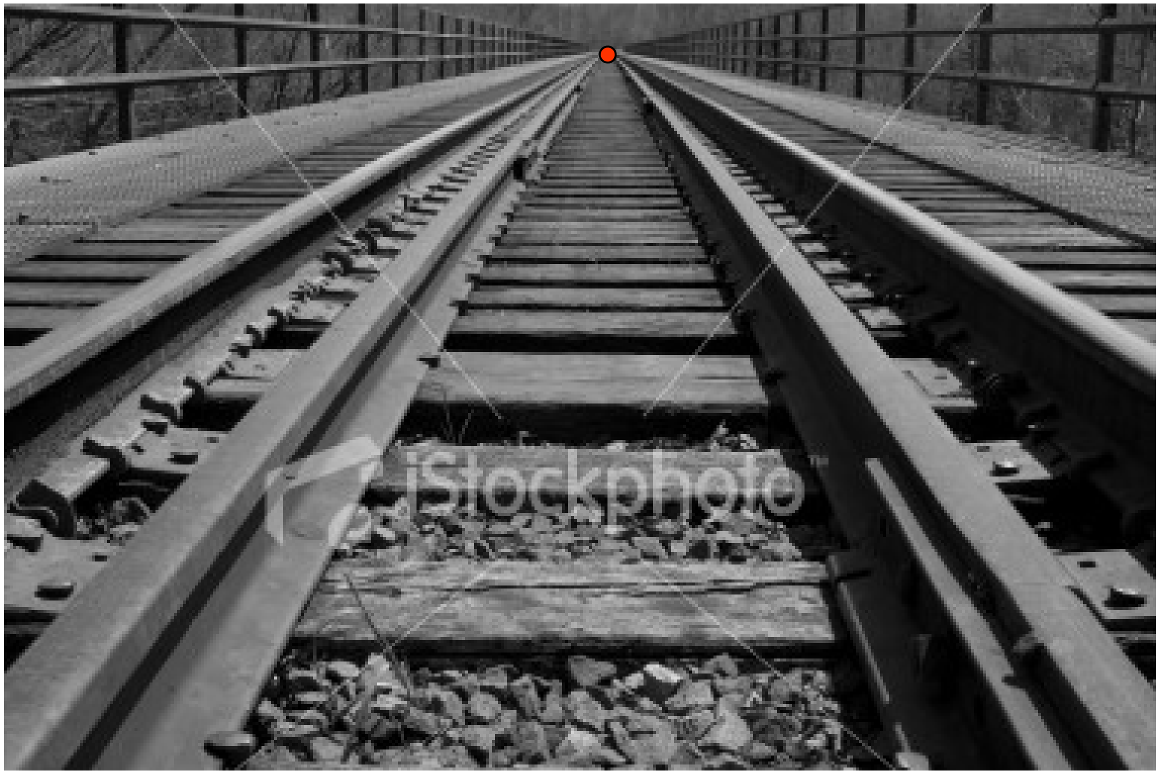
- 3D Polygons ➔ image polygons

Properties: Distant objects appear smaller

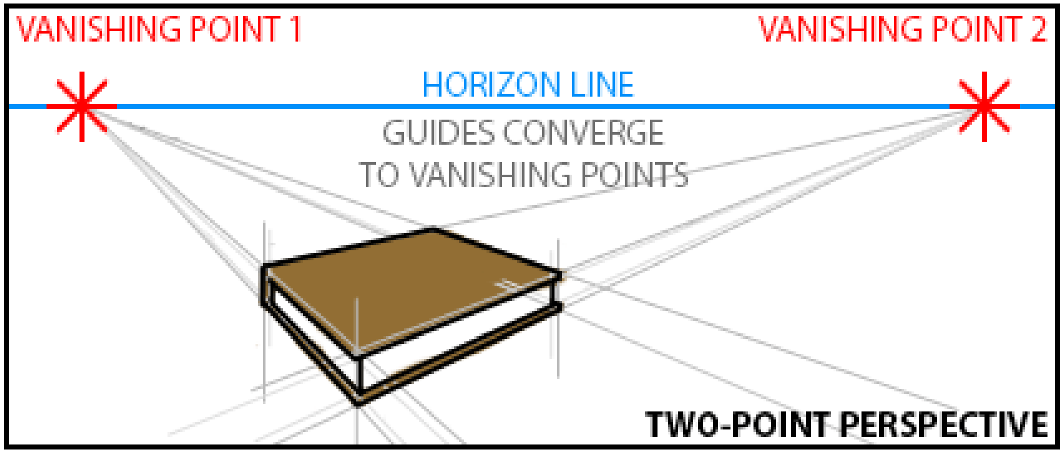


Properties: Vanishing Points

- Image of an infinitely distant 3D point

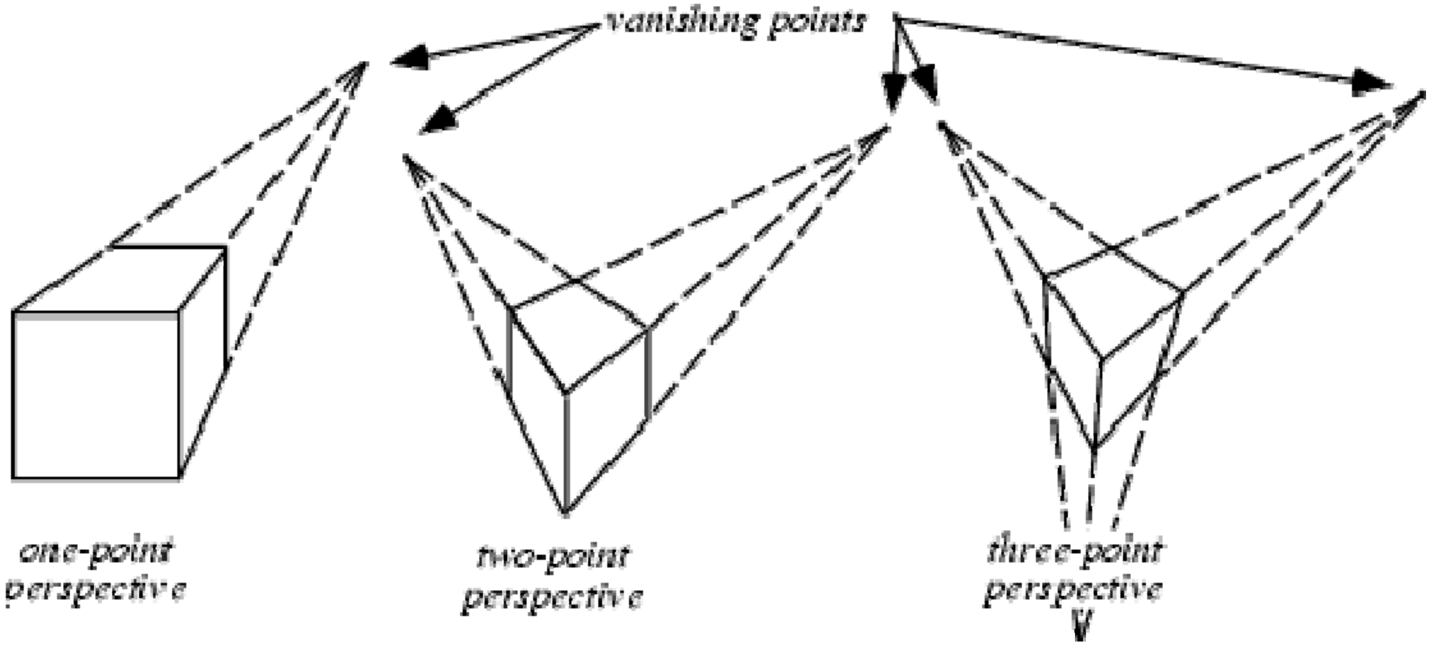


Vanishing Points + Horizon



- Horizon: all vanishing points for World Lines in (or parallel to) plane

Properties: Vanishing Points



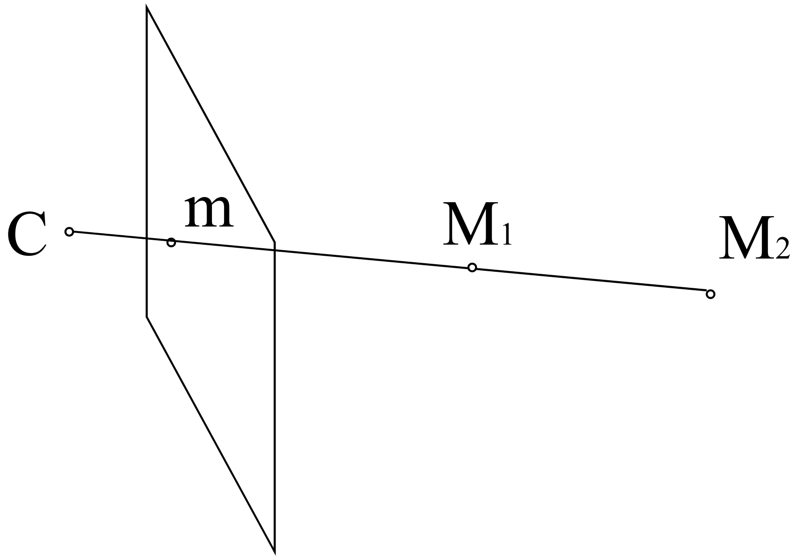
Single View Geometry

Richard Hartley and Andrew Zisserman

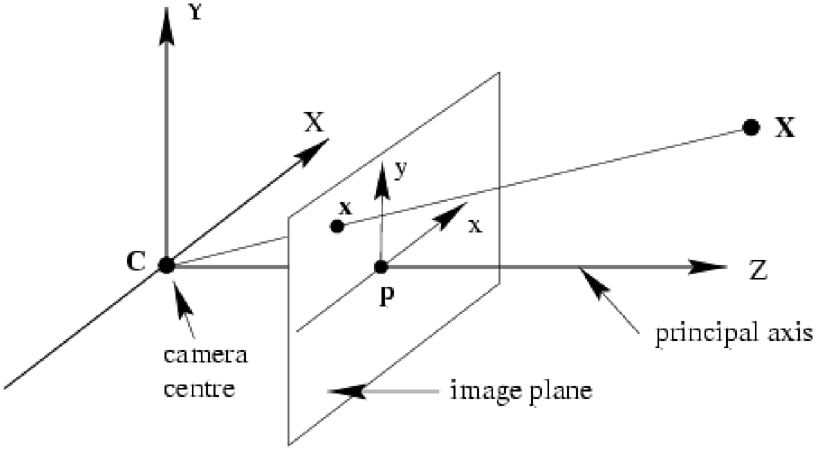
Marc Pollefeys

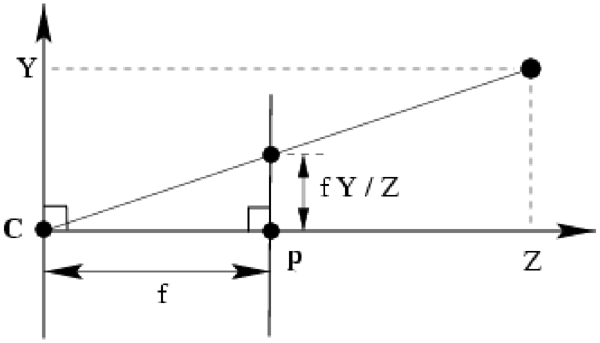
Homogeneous Coordinates

- 3-D points represented as 4-D vectors (X Y Z 1)^T
- Equality defined up to scale
  - (X Y Z 1)^T ~ (WX WY WZ W)^T
- Useful for perspective projection → makes equations linear



Pinhole camera model



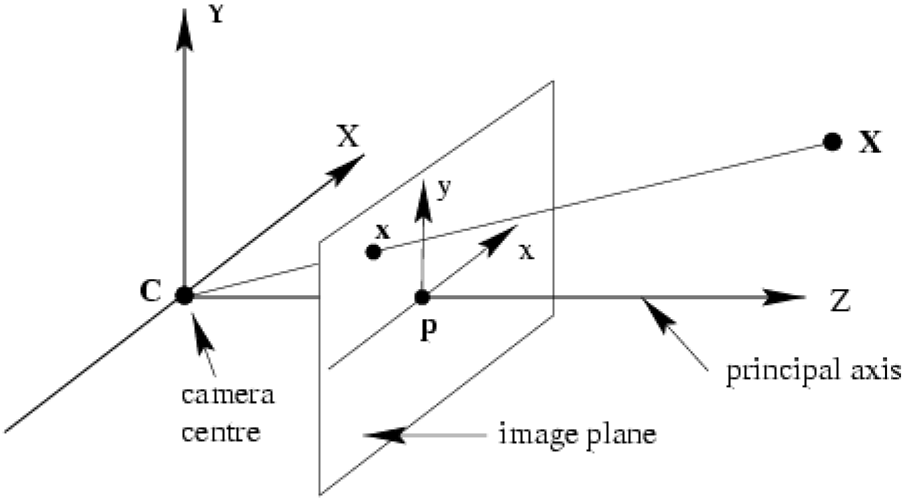


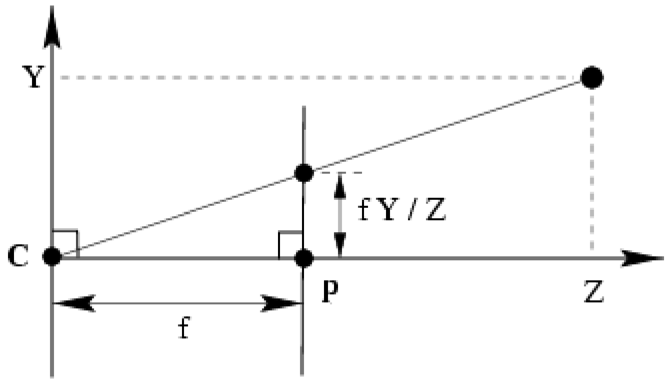
$$(X, Y, Z)^T \mapsto (fX / Z,\ fY / Z)^T$$

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

linear projection in homogeneous coordinates!

The Pinhole Camera



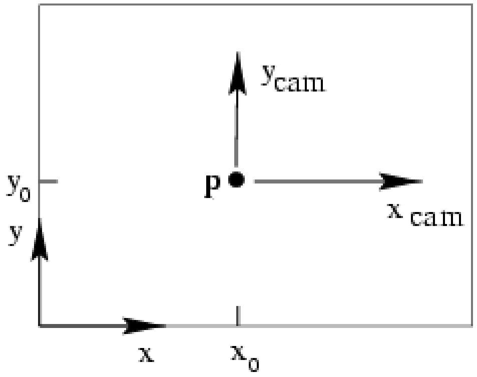


$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Principal Point Offset

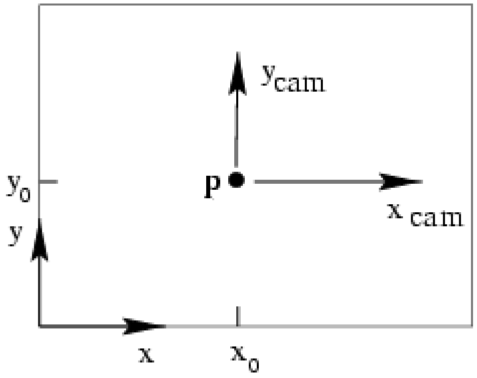


$$(X,Y,Z)^T \mapsto (fX/Z + p_x, fY/Z + p_y)^T$$

$(p_x, p_y)^T$ principal point

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Principal Point Offset



$$\mathrm{x} = \mathrm{K}\left[\mathrm{I} \mid 0\right]\mathrm{X}_{\mathrm{cam}}$$

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_x \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

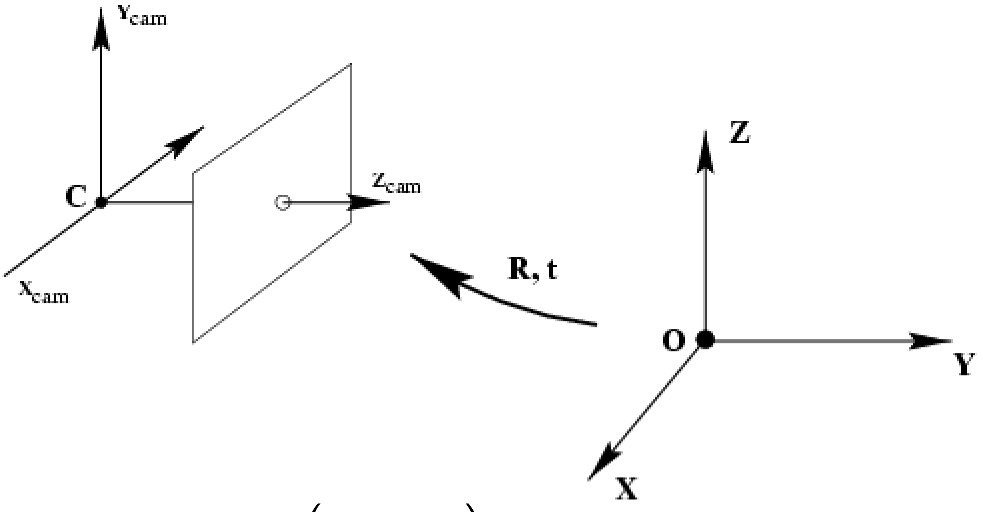
$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$ calibration matrix

Hands On: Image Formation

- For a 640 by 480 image with focal length equal to 640 pixels, find 3D points that are marginally visible at the four borders of the image

- Increase and decrease the focal length. What happens?

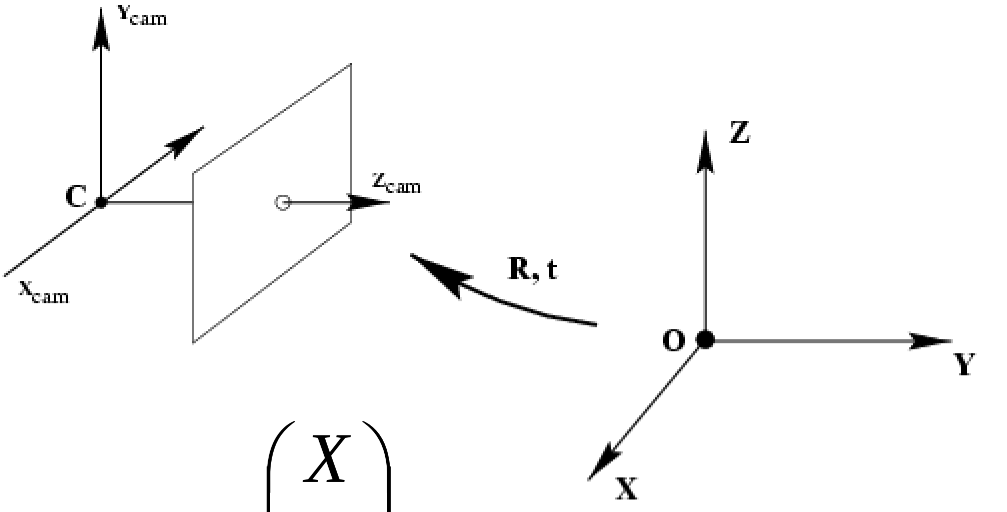
Camera Rotation and Translation



$$\tilde{X}_{cam} = R\left(\tilde{X} - \tilde{C}\right)$$

$$X_{cam} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} X$$

Camera Rotation and Translation



$$X_{cam} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} X$$

$$x = K[I \mid 0] X_{cam}$$

$$x = KR[I \mid -\tilde{C}] X$$

$$x = PX$$

$$P = K[R \mid t]$$

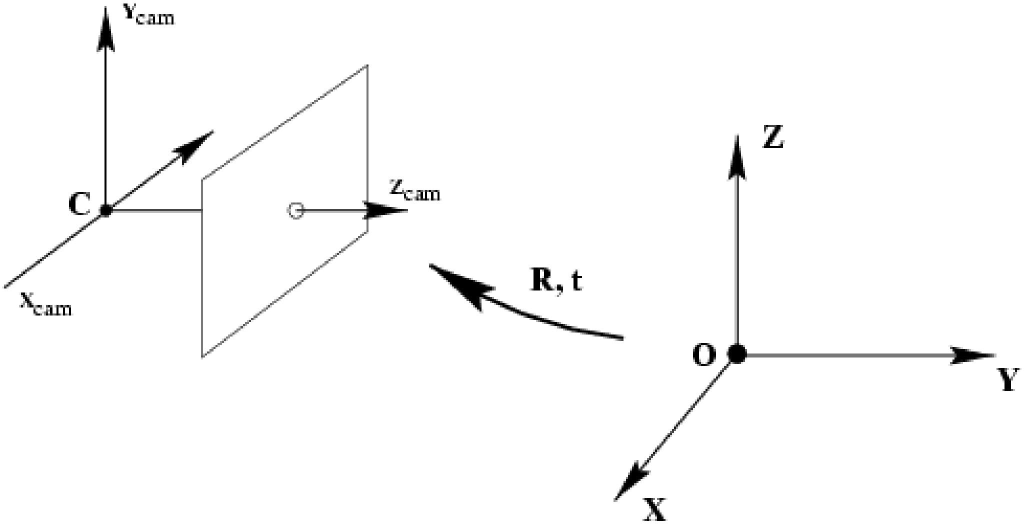
$$t = -R\tilde{C}$$

Intrinsic Parameters

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ & f_y & c_y \\ & & 1 \end{bmatrix} \quad \text{or} \quad \mathbf{K} = \begin{bmatrix} af & f\cos(s) & u_o \\ & f & v_o \\ & & 1 \end{bmatrix}$$

- Camera deviates from pinhole
  - *s*: skew
  - $f_x \neq f_y$: different magnification in *x* and *y*
  - $(c_x\ c_y)$: optical axis does not pierce image plane exactly at the center
- Usually:
  - rectangular pixels: $s = 0$
  - square pixels: $f_x = f_y$
  - principal point known: $(c_x, c_y) = \left(\frac{w}{2}, \frac{h}{2}\right)$

Extrinsic Parameters



Scene motion $$M = \begin{bmatrix} R_{(3x3)} & t_{(3x1)} \\ 0_{(1x3)} & 1 \end{bmatrix}$$

Camera motion $$M' = \begin{bmatrix} R^T_{(3x3)} & -(R^T t)_{3x1} \\ 0_{(1x3)} & 1 \end{bmatrix}$$

Projection matrix

- Includes coordinate transformation and camera intrinsic parameters

$$\lambda\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Everything we need to know about a pinhole camera
- Unambiguous
- Can be decomposed into parameters

Projection matrix

- Mapping from 2-D to 3-D is a function of internal and external parameters

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left[R^{\top} \mid -R^{\top} t \right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\lambda x = K \left[R^{\top} \mid -R^{\top} t \right] X$$

$$\lambda x = PX$$

Two-View Geometry

Slides by R. Hartley, A. Zisserman and M. Pollefeys

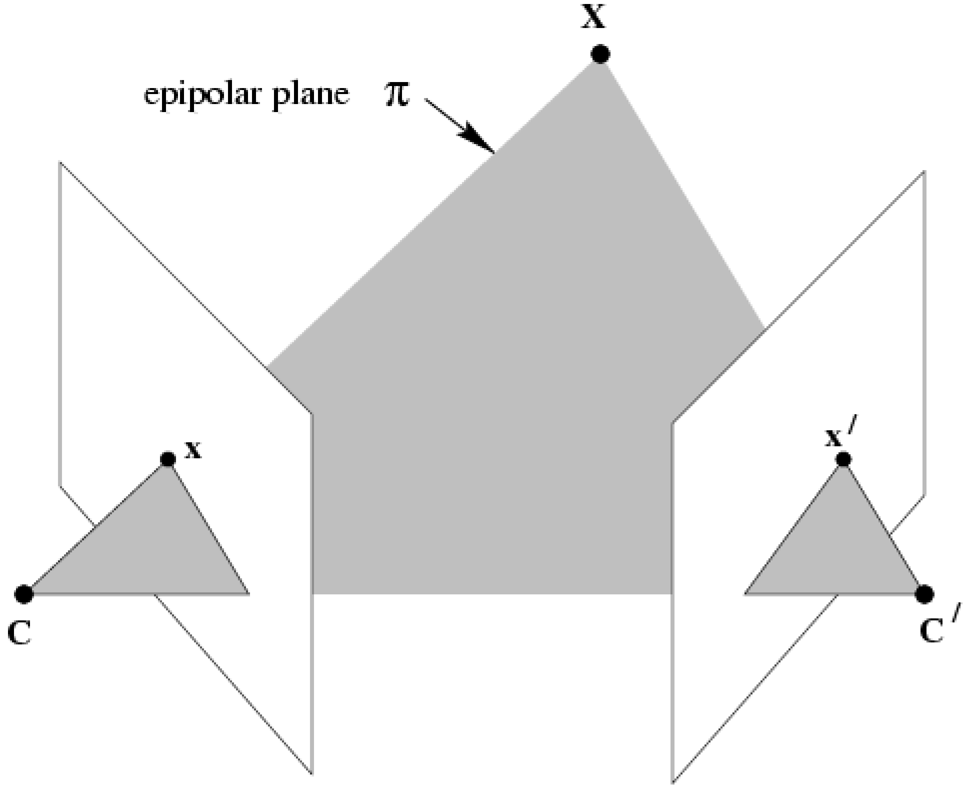
Three questions:

(i) Correspondence geometry: Given an image point x in the first image, how does this constrain the position of the corresponding point x' in the second image?

(ii) Camera geometry (motion): Given a set of corresponding image points $\{x_i \leftrightarrow x'_i\}$, i=1,…,n, what are the cameras P and P' for the two views?

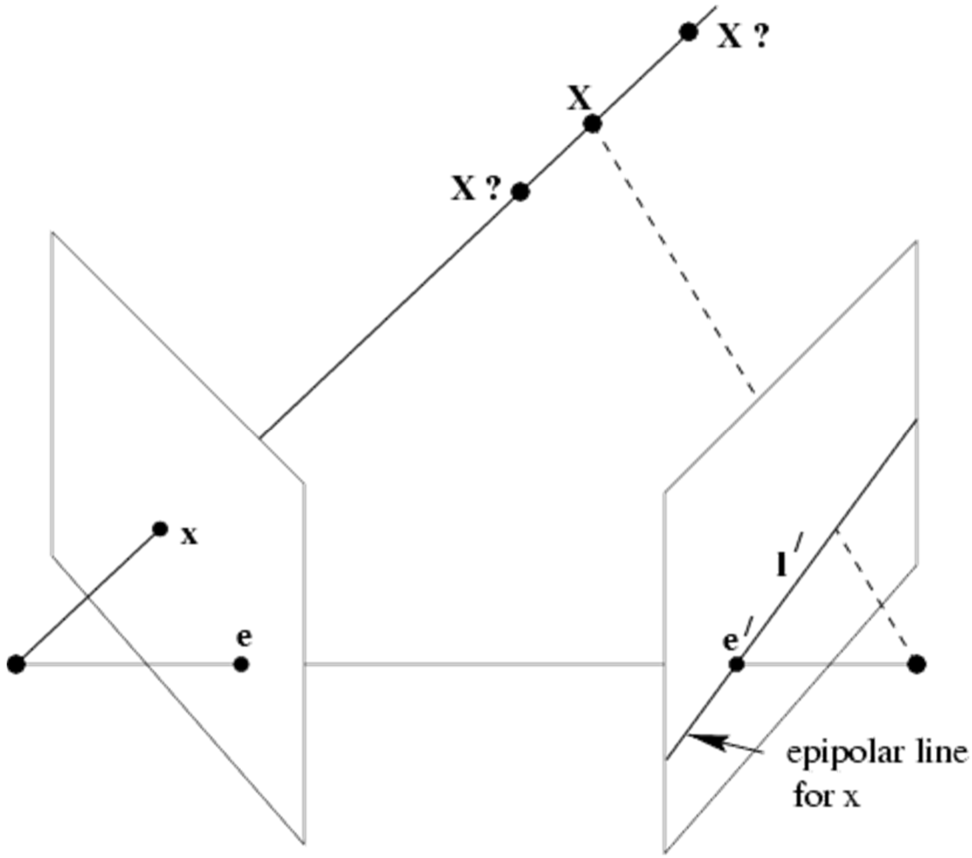
(iii) Scene geometry (structure): Given corresponding image points $x_i \leftrightarrow x'_i$ and cameras P, P', what is the position of (their pre-image) X in space?

The Epipolar Geometry



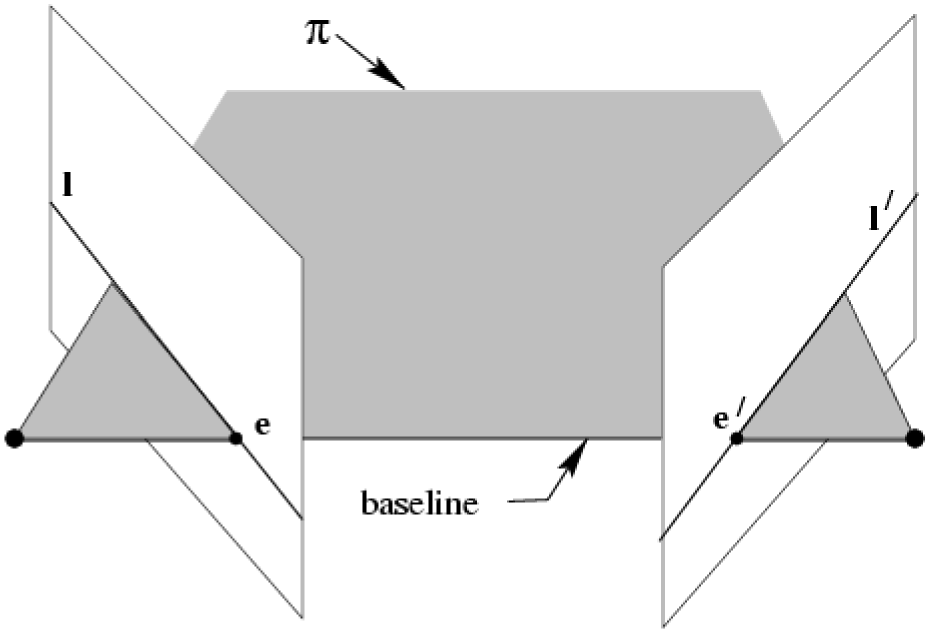
C, C’, x, x’ and X are coplanar

The Epipolar Geometry



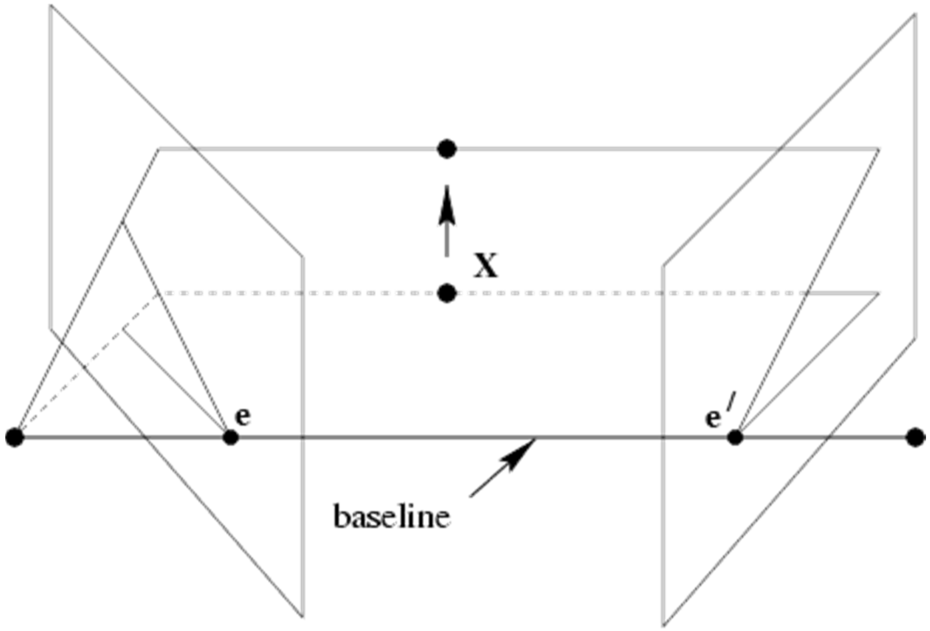
What if only C,C’,x are known?

The Epipolar Geometry



All points on π project on l and l’

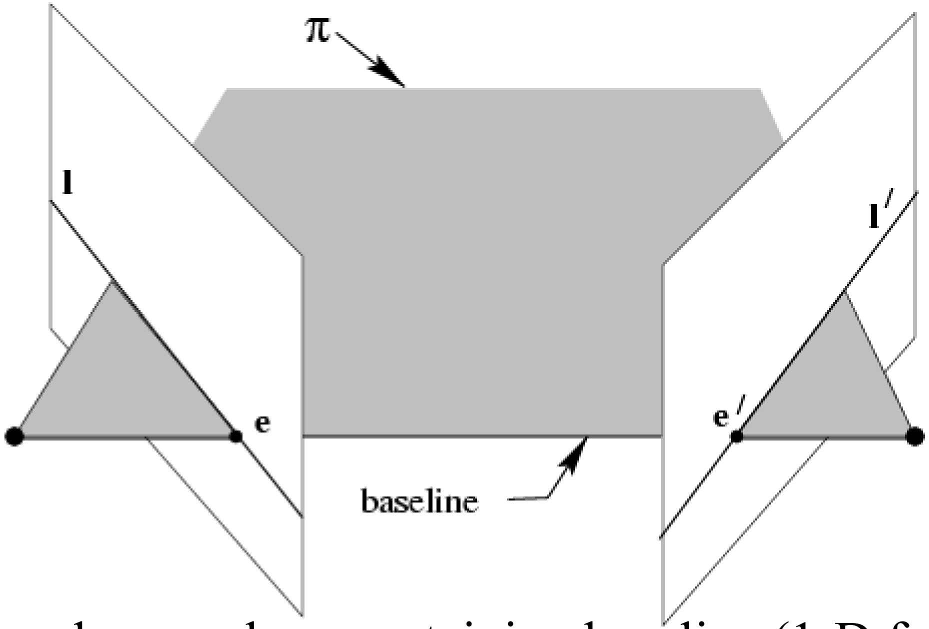
The Epipolar Geometry



Family of planes π and lines l and l'
Intersection in e and e'

The Epipolar Geometry

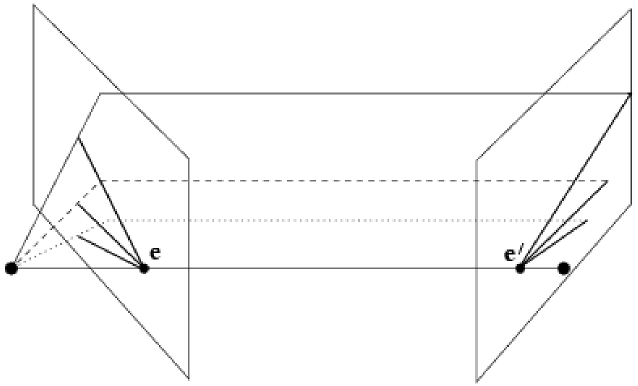
epipoles e, e’
= intersection of baseline with image plane
= projection of projection center in other image
= vanishing point of camera motion direction

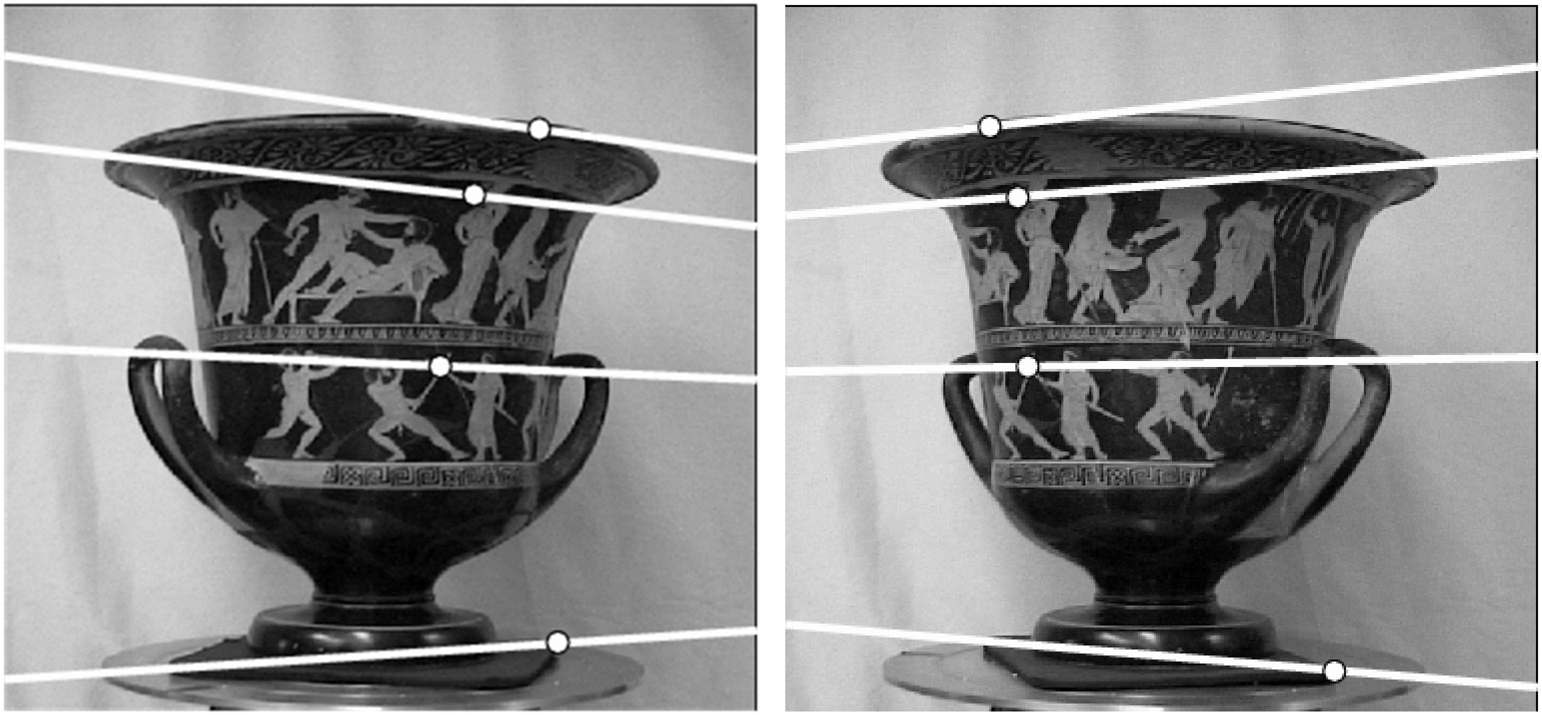


an epipolar plane = plane containing baseline (1-D family)

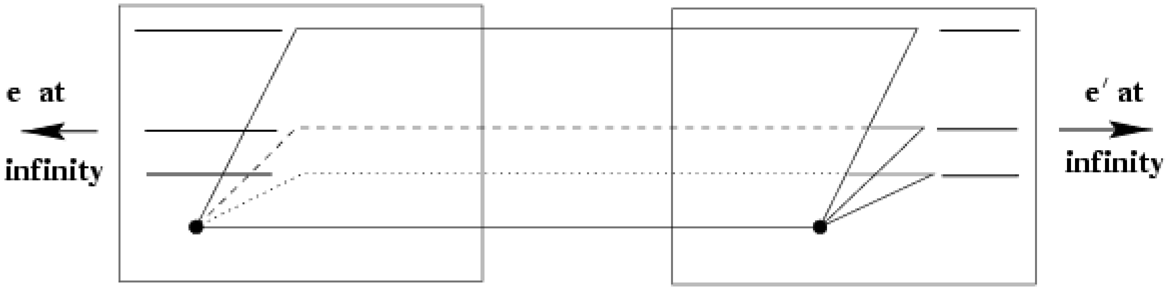
an epipolar line = intersection of epipolar plane with image
(always come in corresponding pairs)

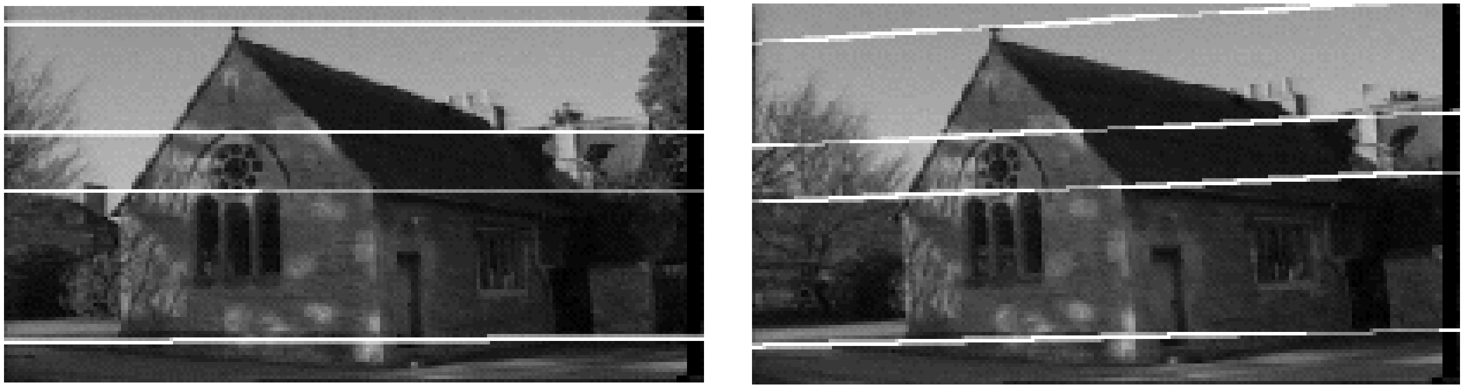
Example: Converging Cameras





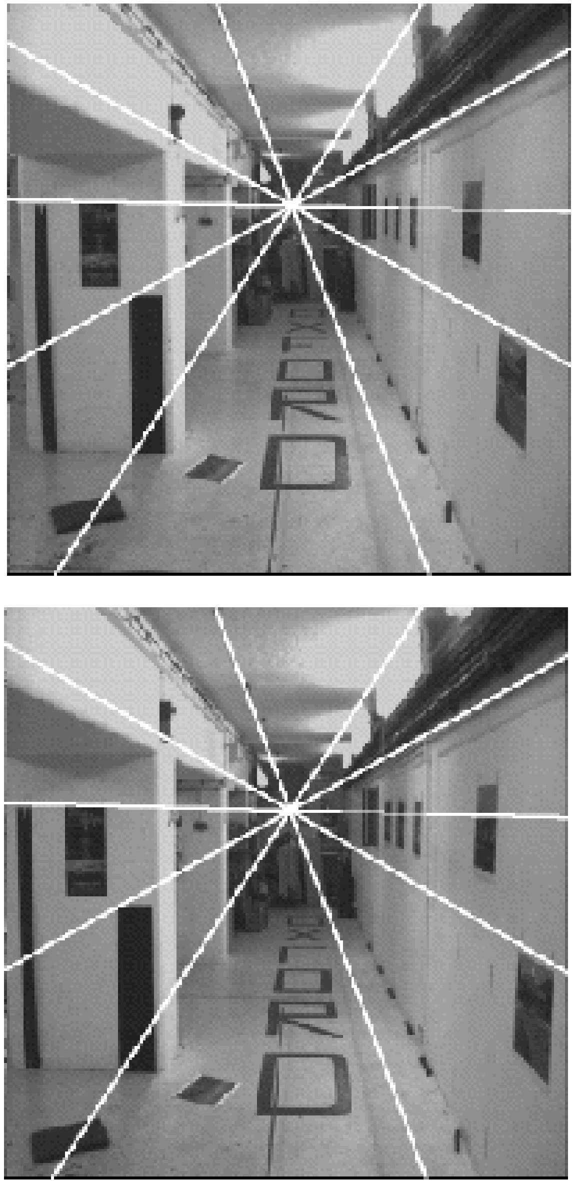
Example: Motion Parallel to Image Plane

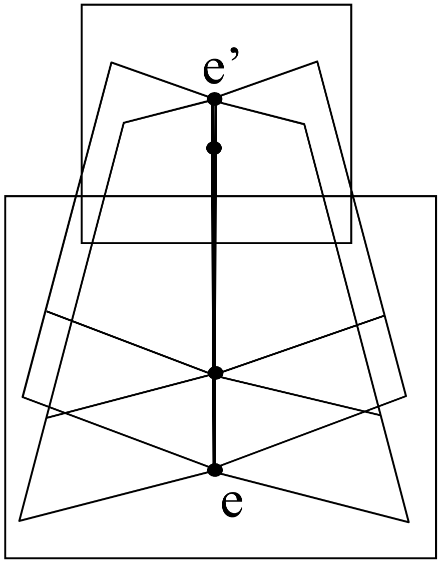




(simple for stereo → rectification)

Example: Forward Motion





The Fundamental Matrix F

algebraic representation of epipolar geometry

$$x \mapsto l'$$

we will see that mapping is a (singular) correlation
(i.e. projective mapping from points to lines)
represented by the fundamental matrix F

The Fundamental Matrix F

correspondence condition

The fundamental matrix satisfies the condition that for any pair of corresponding points x↔x’ in the two images

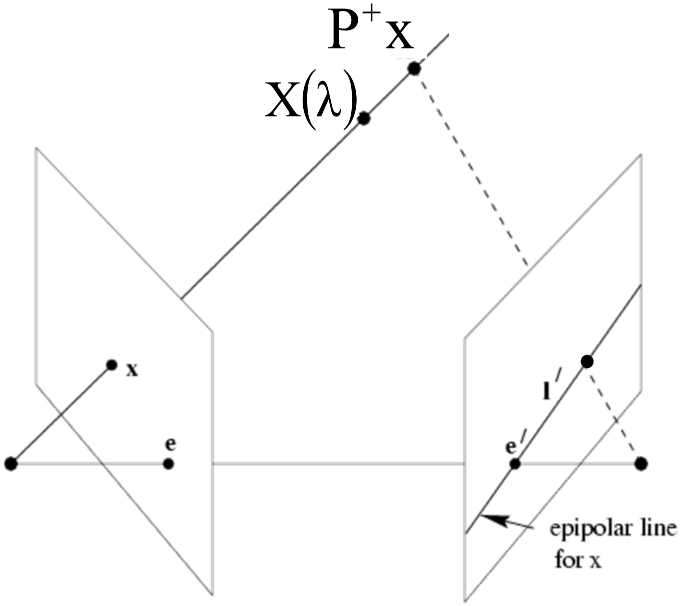
$$\mathrm{x}'^{\mathrm{T}}\,\mathrm{F}\mathrm{x} = 0 \qquad \left(\mathrm{x}'^{\mathrm{T}}\,\mathrm{l}' = 0\right)$$

The Fundamental Matrix F

$$X(\lambda) = P^{+}x + \lambda C \qquad \left(PP^{+} = I\right)$$

$$l = P'C \times P'P^{+}x$$

$$F = [e']_{\times} P'P^{+}$$



(note: doesn’t work for C=C’ ⇒ F=0)

The Fundamental Matrix F

F is the unique 3x3 rank 2 matrix that satisfies $x'^TFx=0$ for all $x \leftrightarrow x'$

- **(i) Transpose:** if F is fundamental matrix for (P,P'), then F^T is fundamental matrix for (P',P)
- **(ii) Epipolar lines:** $l'=Fx$ & $l=F^Tx'$
- **(iii) Epipoles:** on all epipolar lines, thus $e'^TFx=0, \forall x \Rightarrow e'^TF=0$, similarly $Fe=0$
- **(iv) F** has 7 d.o.f. , i.e. 3x3-1(homogeneous)-1(rank2)
- **(v) F** is a correlation, projective mapping from a point x to a line $l'=Fx$ (not a proper correlation, i.e. not invertible)

Two View Geometry Computation: Linear Algorithm

For every match (m,m´): $\mathrm{x'}^{\mathrm{T}}\,\mathrm{Fx} = 0$

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

separate known from unknown

$$[x'x, x'y, x', y'x, y'y, y', x, y, 1][f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}]^{\mathrm{T}} = 0$$

(data)

(unknowns)

(linear)

$$\begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathrm{f} = 0$$

$$\mathrm{Af} = 0$$

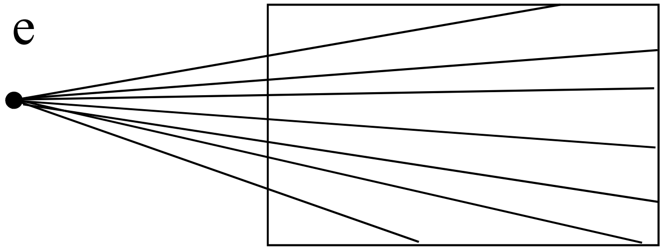
Benefits from having F

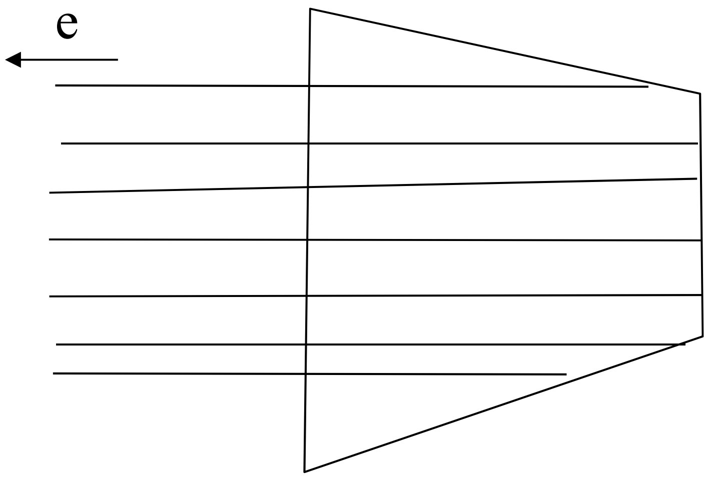
- Given a pixel in one image, the corresponding pixel has to lie on epipolar line
- Search space reduced from 2-D to 1-D

Image Pair Rectification

simplify stereo matching
by warping the images

Apply projective transformation so that epipolar lines
correspond to horizontal scanlines





$$\begin{bmatrix}1\\0\\0\end{bmatrix} = He$$

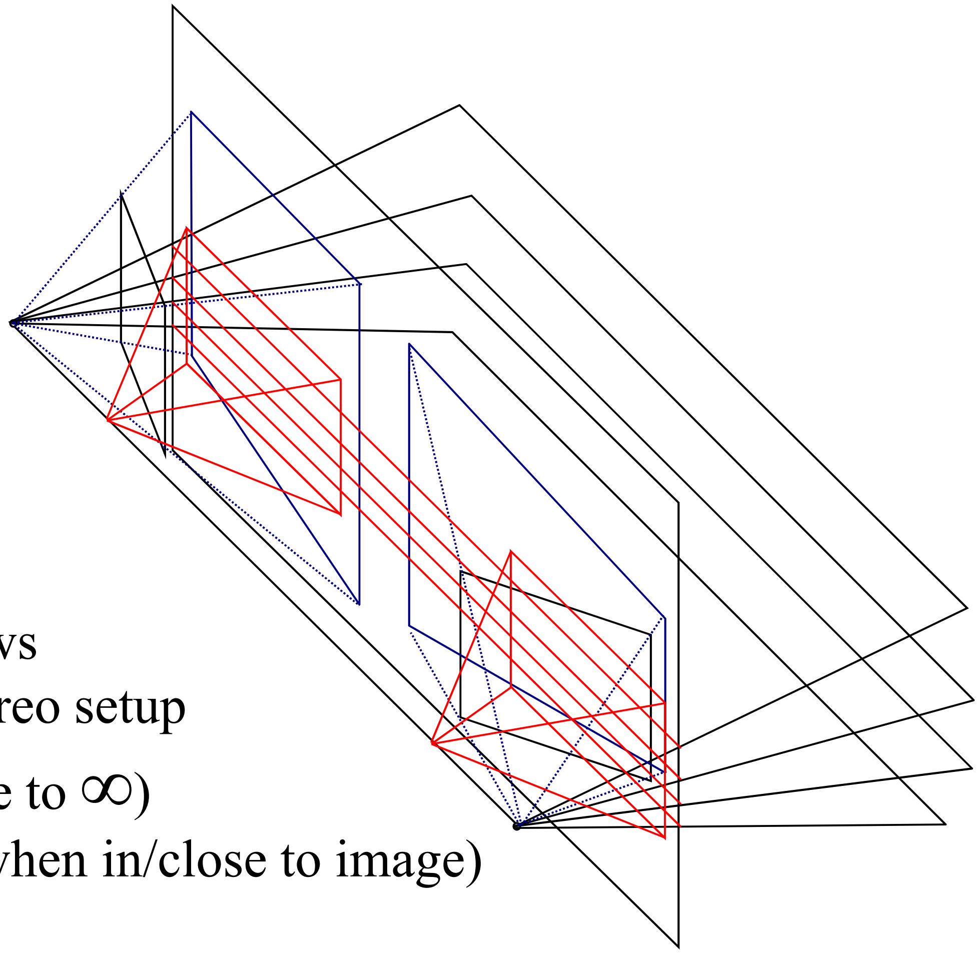
map epipole e to (1,0,0)

try to minimize image distortion

problem when epipole in (or close to) the image

Planar Rectification

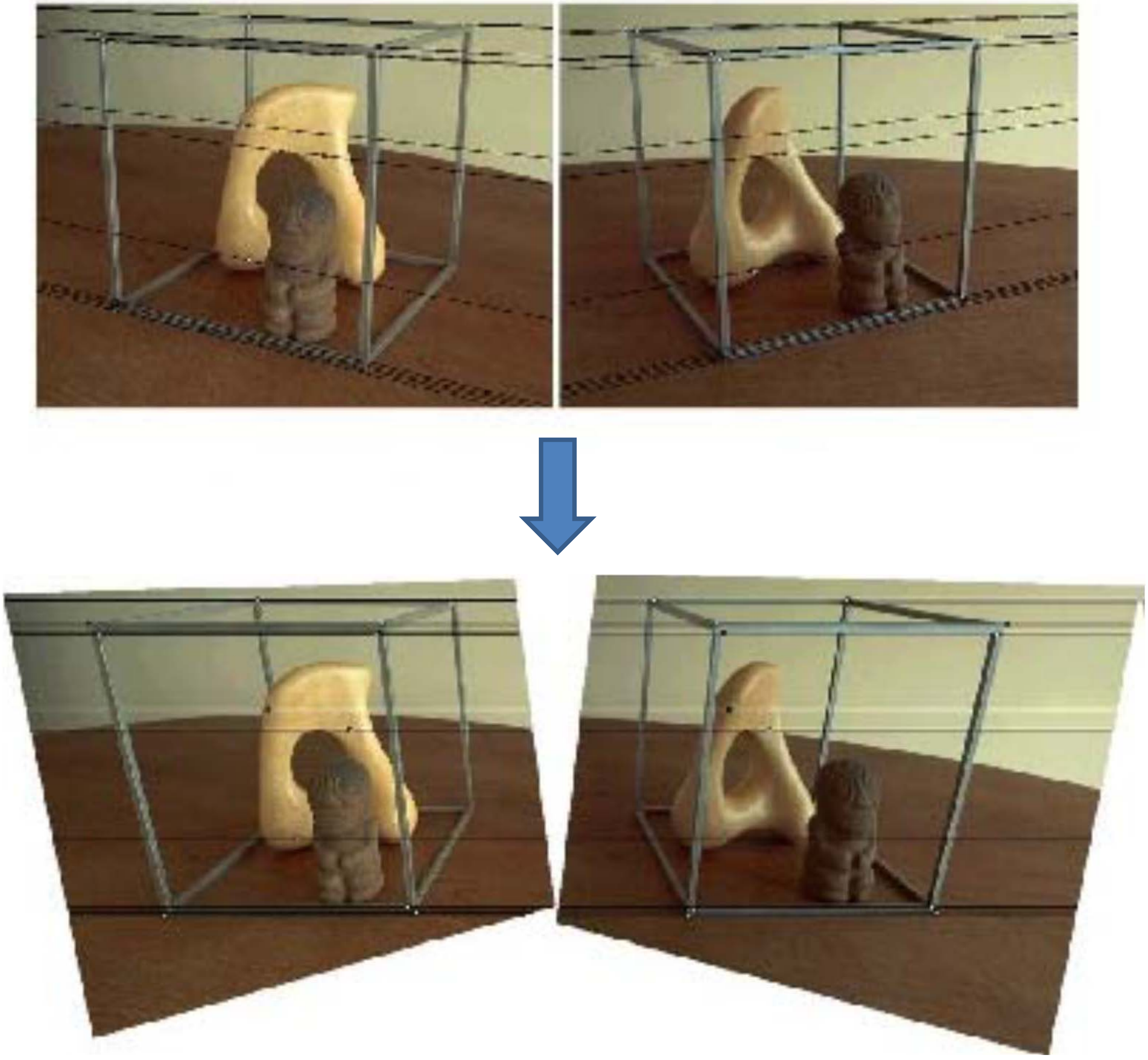
(standard approach)



Bring two views
to standard stereo setup

(moves epipole to ∞)
(not possible when in/close to image)

Rectification Example



Slide Credits

- This set of sides contains contributions kindly made available by the following authors
  - Derek Hoiem
  - Kristen Grauman
  - Marc Pollefeys
  - David Cappelleri
  - Richard Hartley
  - Andrew Moore
  - John Oliensis
  - Steve Seitz
  - Andrew Zisserman