Computer Vision I - Algorithms and Applications: *Image Formation Process – Part 2*

Carsten Rother

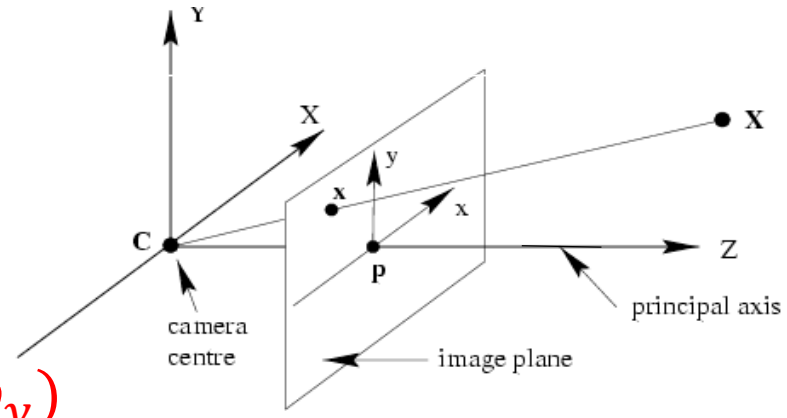
17/11/2013

Roadmap this lecture

- Geometric primitives and transformations (sec. 2.1.1-2.1.4)
- Geometric image formation (sec 2.1.5, 2.1.6)
  - Pinhole camera
  - Lens effects
- Photometric image formation process (sec 2.2)
- The Human eye
- Camera Types and Hardware (sec 2.3)
- Appearance based matching

Reminder: Pinhole Camera (Geometry)

- Camera matrix P has 11 DoF



- Intrinsic parameters
  - Principal point coordinates (p_x, p_y)
  - Focal length f
  - Pixel magnification factors m
  - Skew (non-rectangular pixels) s

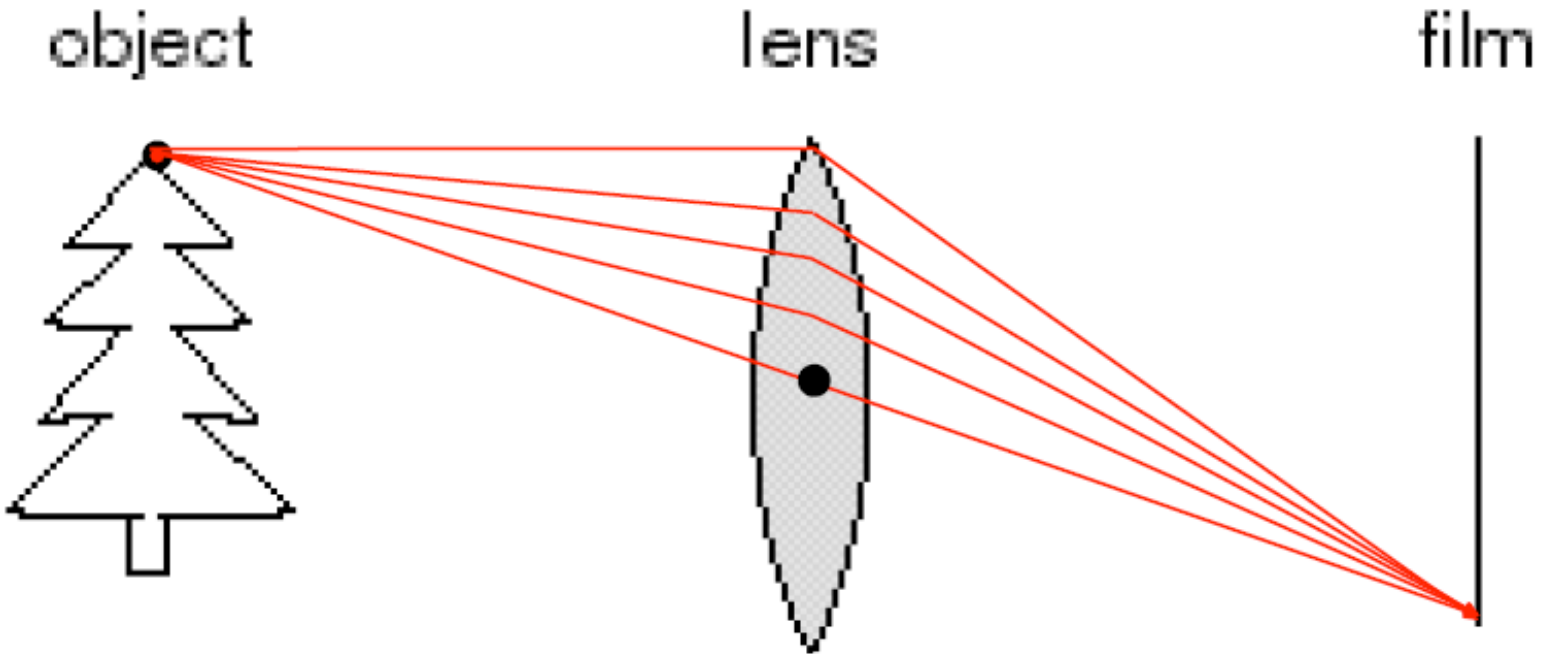
$$\boldsymbol{x} = \boldsymbol{P}\,\boldsymbol{X}$$

$$\boldsymbol{x} = \boldsymbol{K}\,\boldsymbol{R}\,(\boldsymbol{I}_{3\times 3} \mid -\tilde{\boldsymbol{C}})\,\boldsymbol{X}$$

$$\boldsymbol{K} = \begin{bmatrix} f & s & p_x \\ 0 & mf & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Extrinsic parameters
  - Rotation $\boldsymbol{R}$ (3DoF) and translation $\tilde{\mathbf{C}}$ (3DoF) relative to world coordinate system

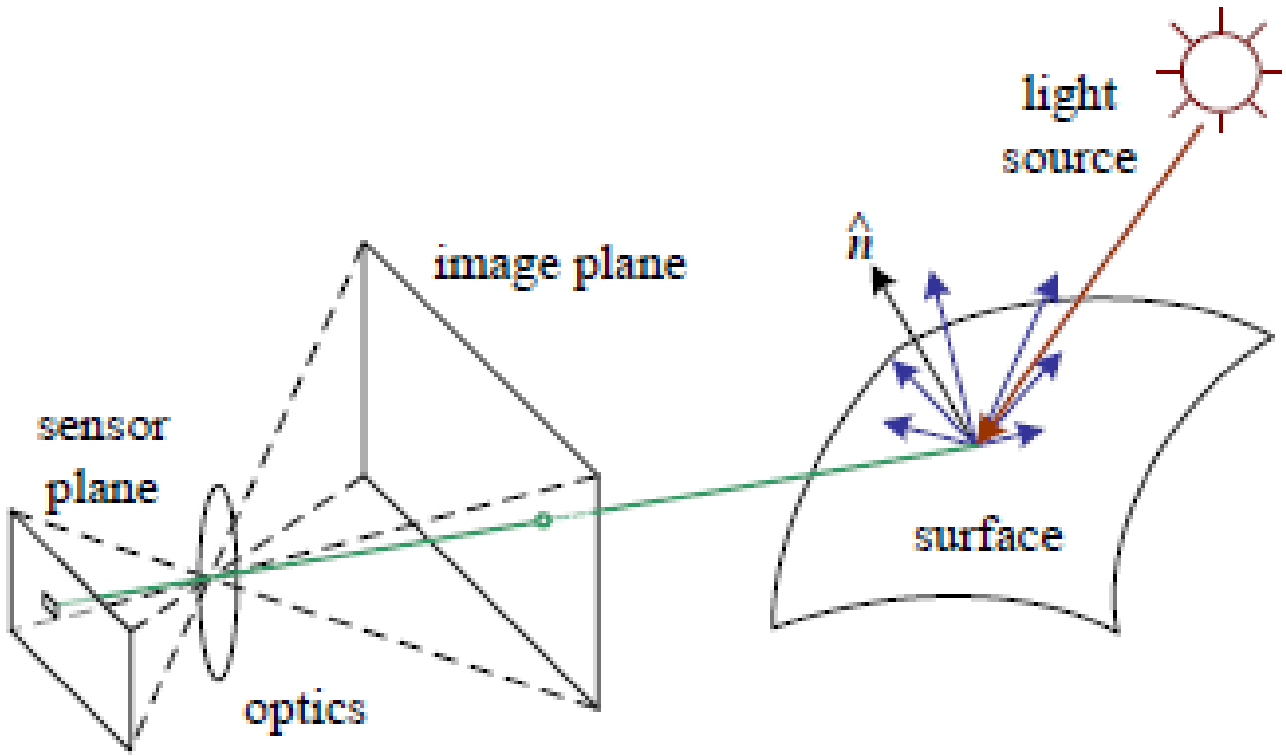
Reminder: Lens effects



Roadmap this lecture

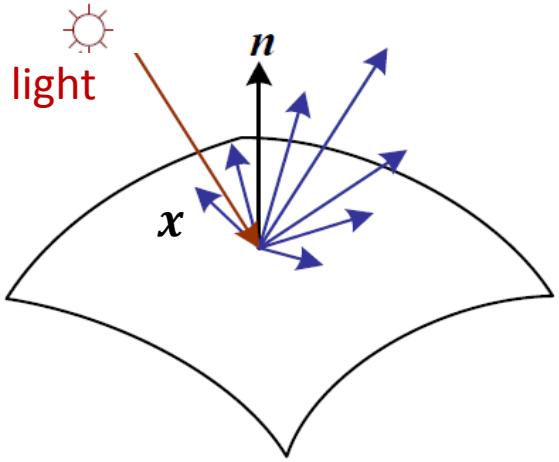
- Geometric primitives and transformations (sec. 2.1.1-2.1.4)
- Geometric image formation (sec 2.1.5, 2.1.6)
  - Pinhole camera
  - Lens effects
- Photometric image formation process (sec 2.2)
- The Human eye
- Camera Types and Hardware (sec 2.3)
- Appearance based matching

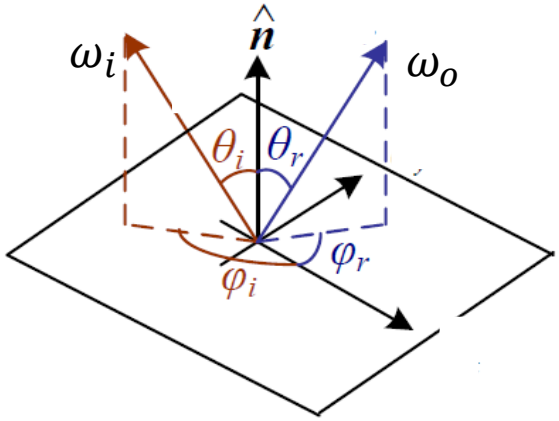
Image formation Process



Model of the Surface

BRDF (Bi-diretcional Reflectance Function)





Rendering equation:

Outgoing energy (shining surface)

Incoming light

$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t)\, L_i(\mathbf{x}, \omega_i, \lambda, t)\, (\omega_i \cdot \mathbf{n})\, \mathrm{d}\,\omega_i$$

Outgoing energy:

- t – time
- **x** – position
- λ – wavelength
- ω_i / ω_o - incoming / outgoing direction

BRDF function (4DoF only ω_i , ω_o considered)

Fore-shorting angle (always ≥ 0)

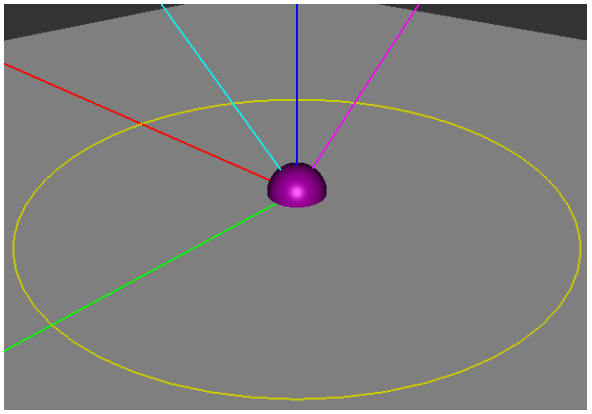
Example: diffuse illumination

Rendering equation

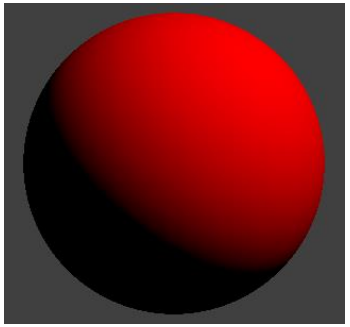
$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t)\, L_i(\mathbf{x}, \omega_i, \lambda, t)\, (\omega_i \cdot \mathbf{n})\, \mathrm{d}\,\omega_i$$

0 constant

DIFFUSE



f_r same for all ω_o



Single Light source:

$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t)\, L_i(\mathbf{x}, \omega_i, \lambda, t)\, (\omega_i \cdot \mathbf{n})\, \mathrm{d}\,\omega_i$$

0 constant constant (fixed ω_i)

Shading effects come from: $\max(0, w_i\, \boldsymbol{n})$

Examples: BRDFs

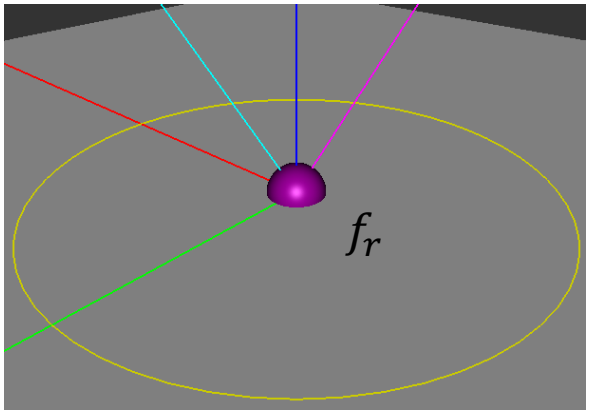
Rendering equation (single light source)

$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t)\, L_i(\mathbf{x}, \omega_i, \lambda, t)\, (\omega_i \cdot \mathbf{n})\, d\omega_i$$

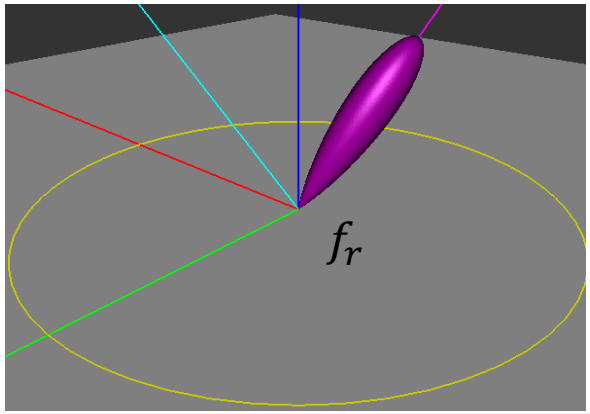
0

constant (fixed ω_i)

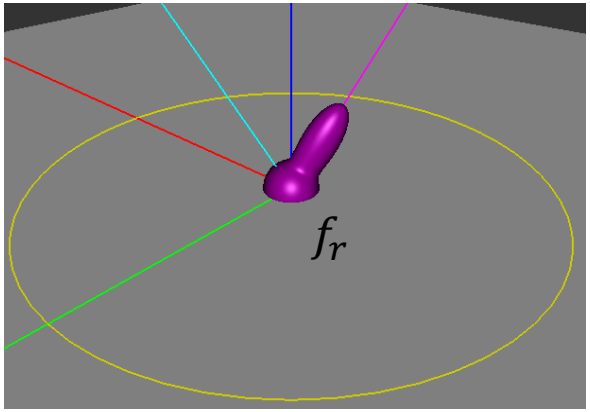
DIFFUSE

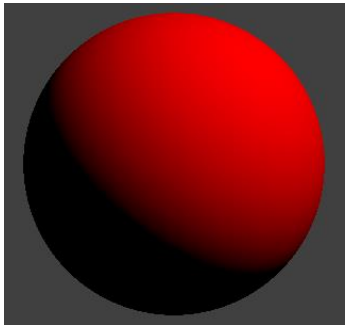


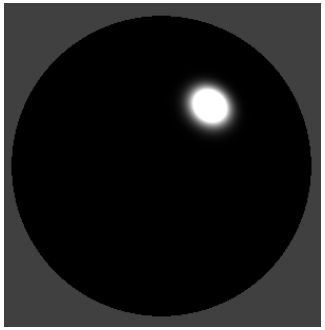
SPECULAR

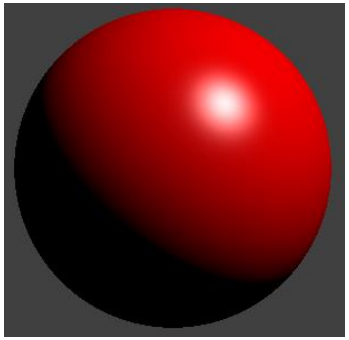


DIFFUSE + ROUGH SPECULAR









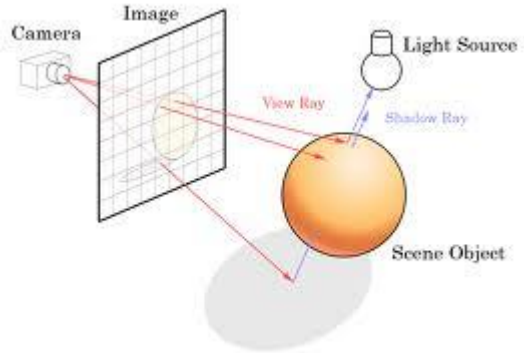
Single light source (top, right)

Inside a graphics engine

- Rendering equation

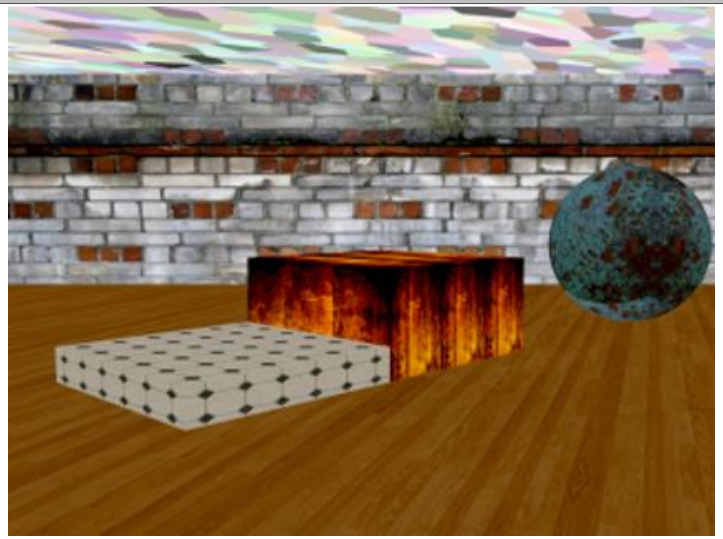
$$L_o(\mathbf{x}, \omega_o, \lambda, t) = L_e(\mathbf{x}, \omega_o, \lambda, t) + \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o, \lambda, t)\, L_i(\mathbf{x}, \omega_i, \lambda, t)\, (\omega_i \cdot \mathbf{n})\, \mathrm{d}\,\omega_i$$

- Ray tracing

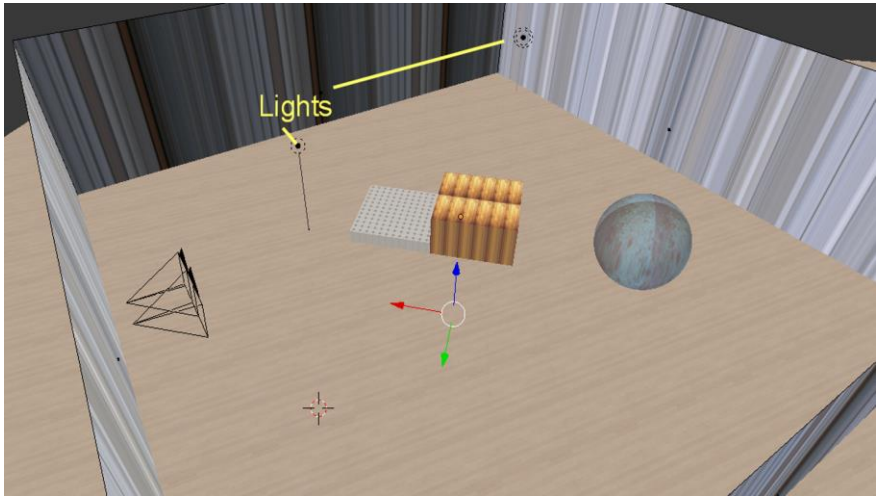


- Radiosity

Example

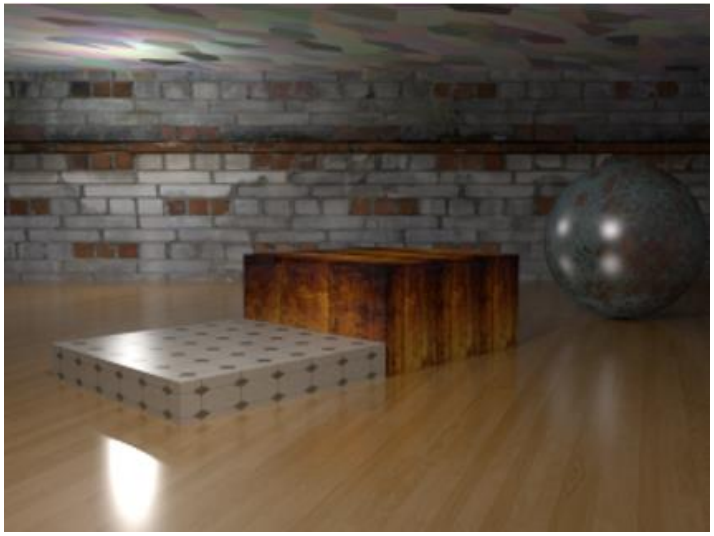


Texture only



3D scene in blender

Direct light

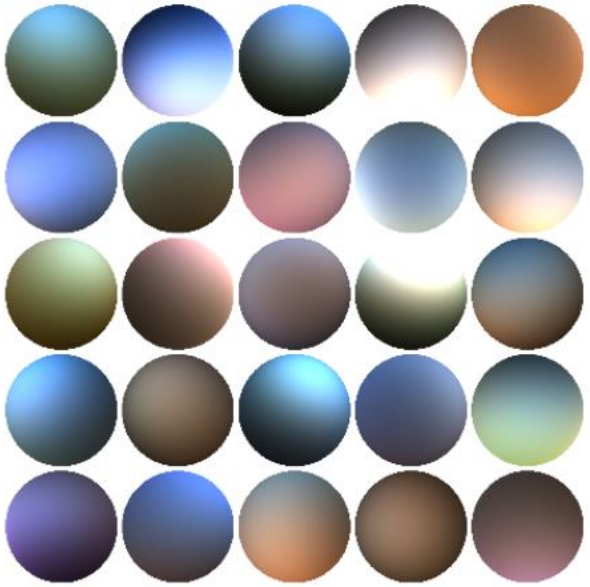


Global (direct + indirect) light

How to model light?

- Point light source
- Area light source
- Non-local illumination situation can be approximated with spherical harmonics

First 9 spherical harmonics (gray scale)



They can produce this

Capturig BRDF / BTF capture

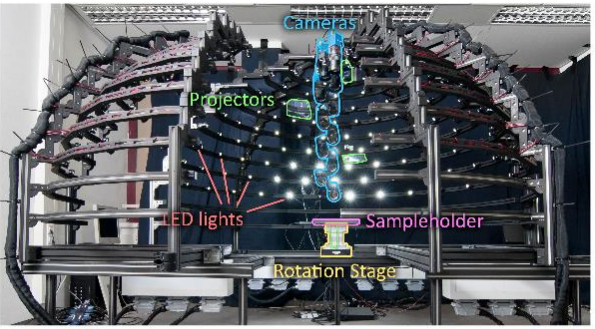
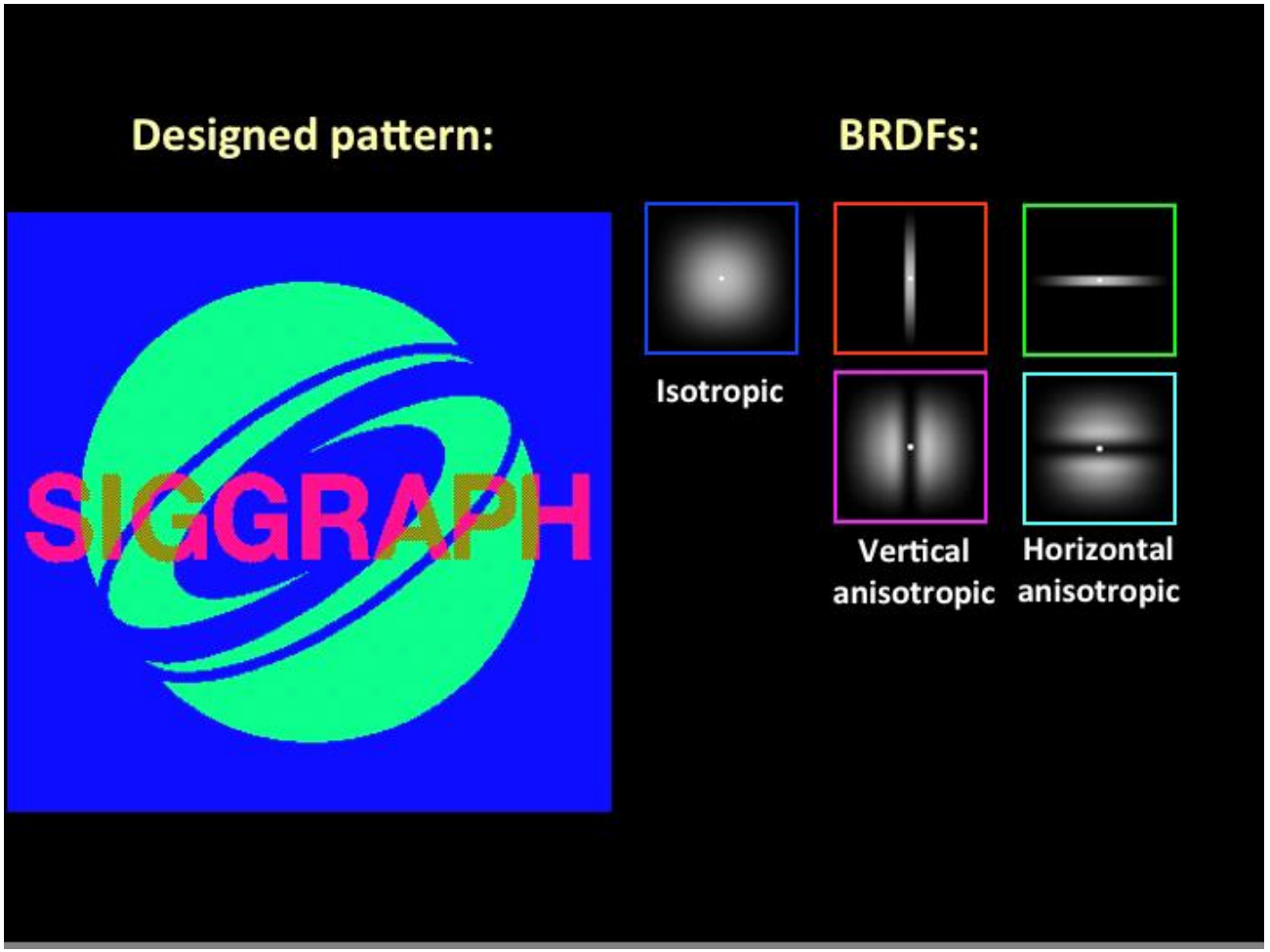


Figure 1: The DOME II BTF acquisition setup. One quarter has been slid open to expose the view on the inside.

BTF – Bidirectional Texture function. At every spatial location $\boldsymbol{x}$ a different BRDF

[Christopher Schwartz, Ralf Sarlette, Michael Weinmann and Reinhard Klein]

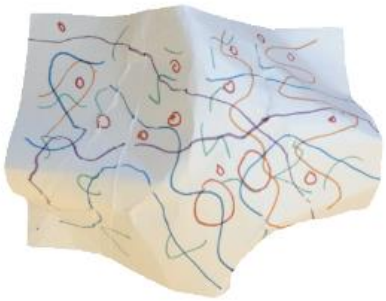
Fabricating BRDFs



[Levin et al. Siggraph 2013]

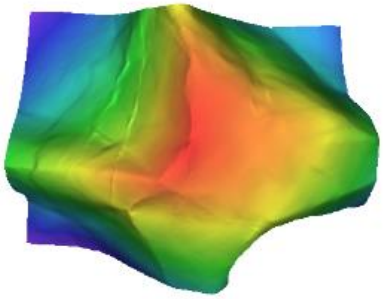
Single Image

Input:

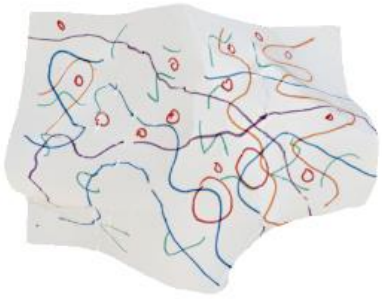


Image

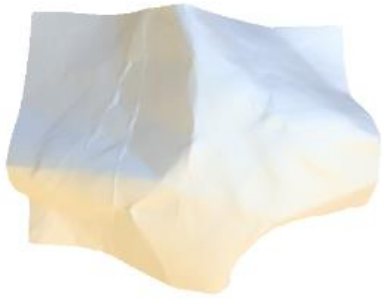
Output:



Shape



Reflectance

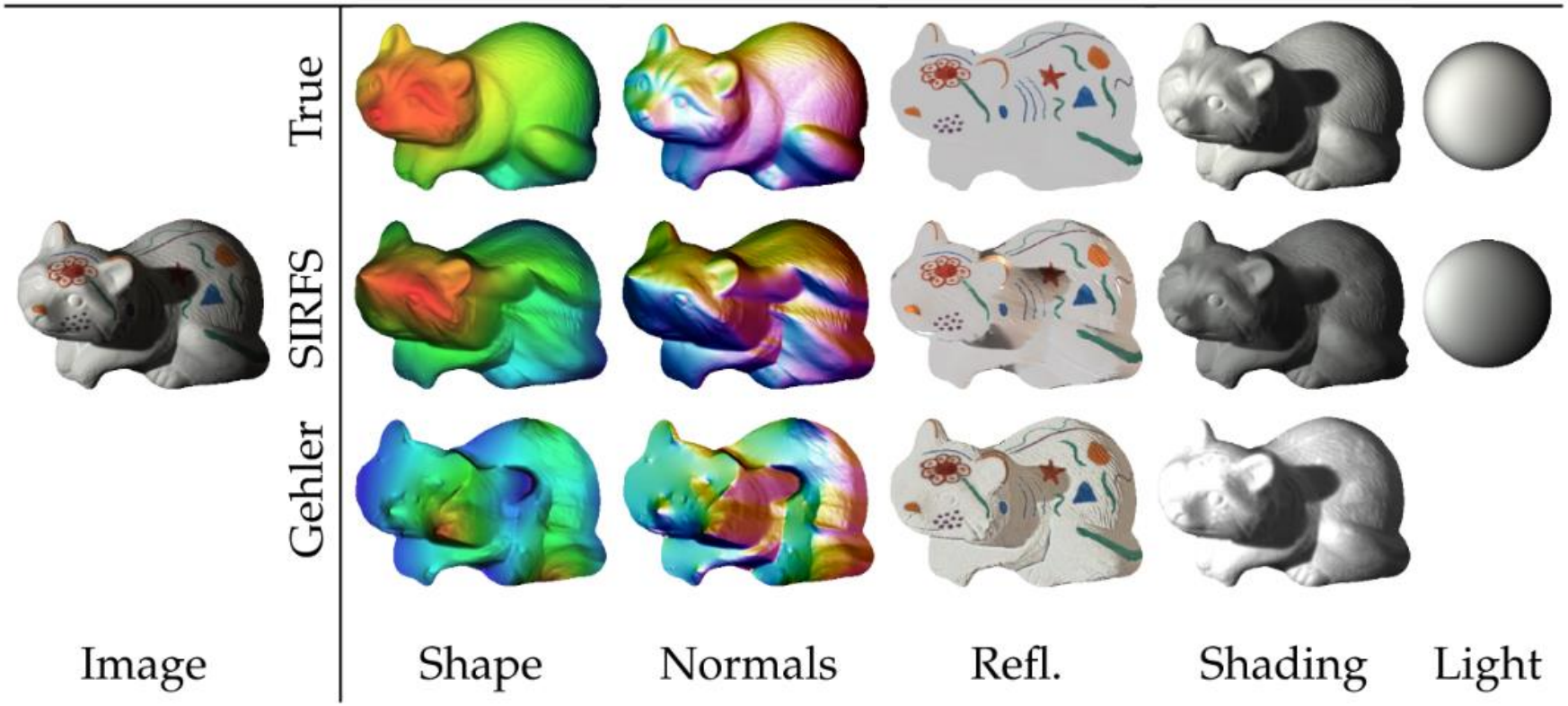


Shading

Illumination

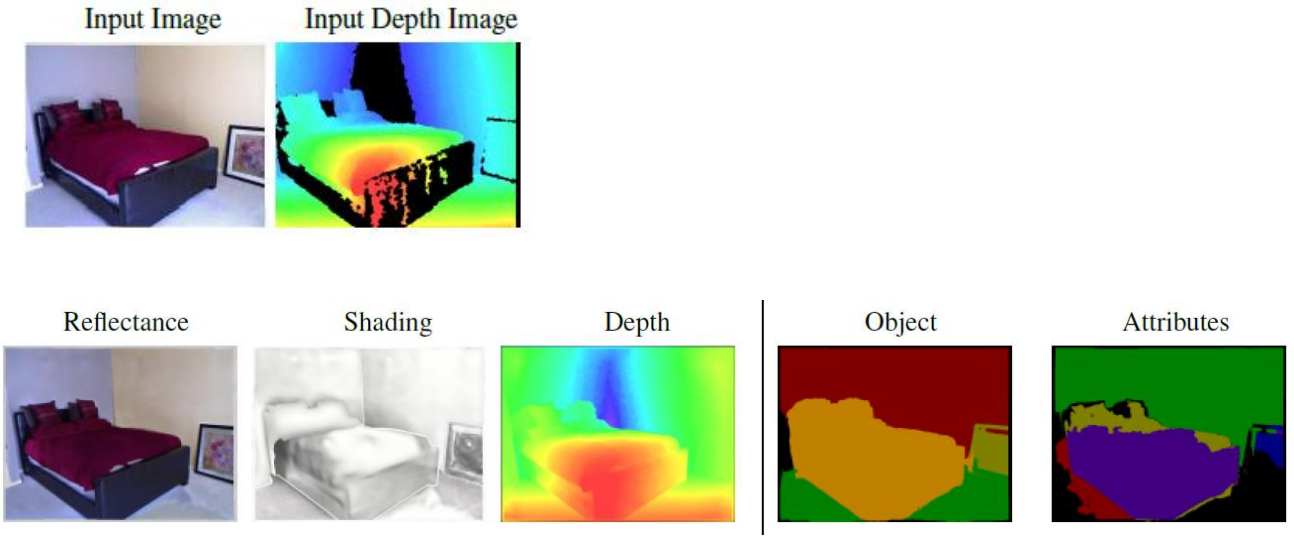
[Barron and Malik 2012]

Single Image … far from being solved



[Barron and Malik 2012]

Reconstruction and recognition

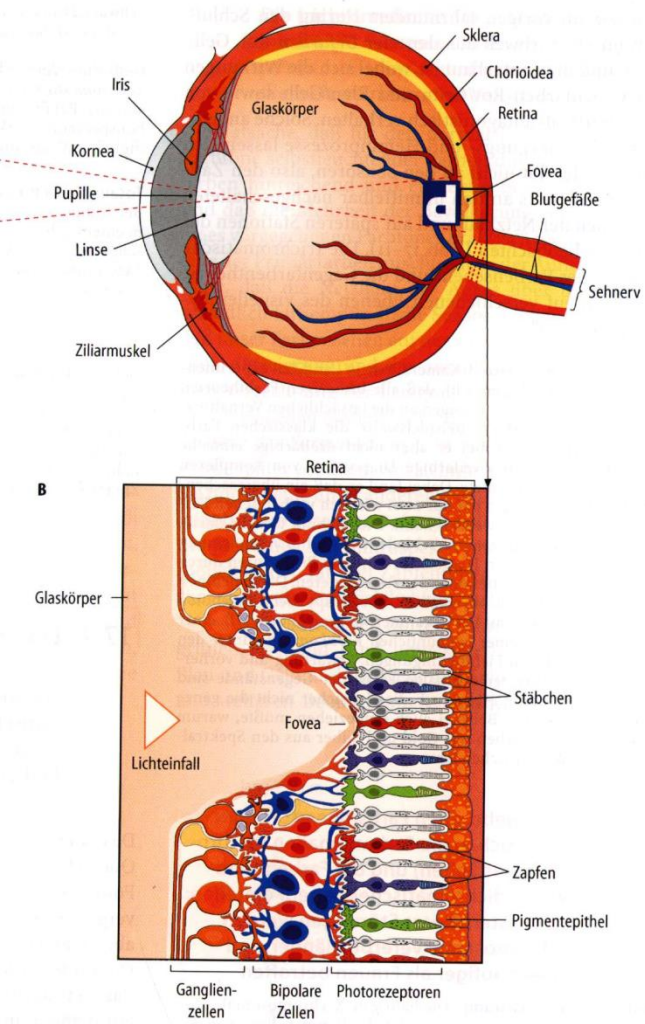


[Vinett, Rother, Torr, NIPS '13]

Roadmap this lecture

- Geometric primitives and transformations (sec. 2.1.1-2.1.4)
- Geometric image formation (sec 2.1.5, 2.1.6)
  - Pinhole camera
  - Lens effects
- Photometric image formation process (sec 2.2)
- The Human eye
- Camera Types and Hardware (sec 2.3)
- Appearance based matching

The Human eye



- The retina contains different types of sensors: cones “Zapfen” (colors, 6 million) and rods “Staebchen” (gray levels, 120 million)
- The resolution is much higher In fovea centralis
- Light first passes through the layer of neurons before it reaches photo sensors (smoothing). Only in the “fovea centralis” the light hits directly the photo sensors
- Signal goes out the other way:
  Retina → Ganglion cells (1 million) → Optic nerve → 1 MPixel Camera ?

Spatial resolution

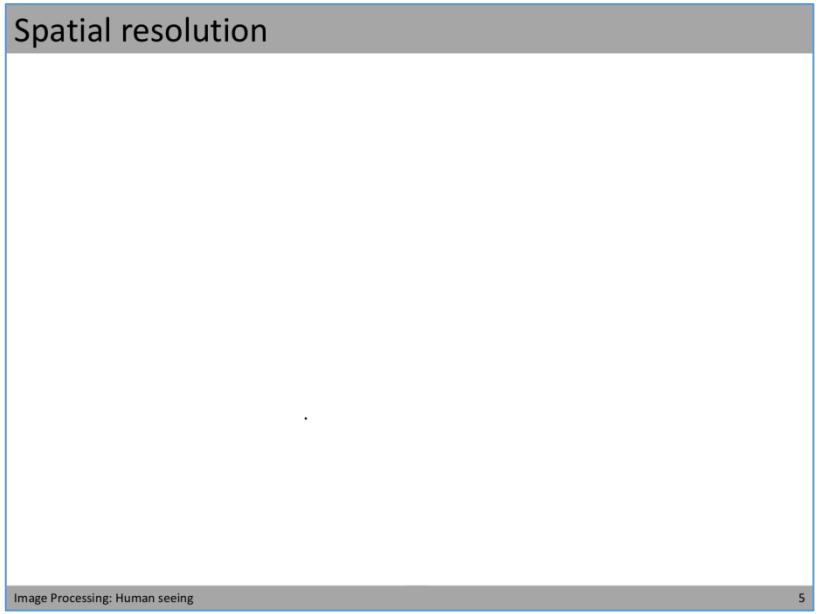
Spatial resolution

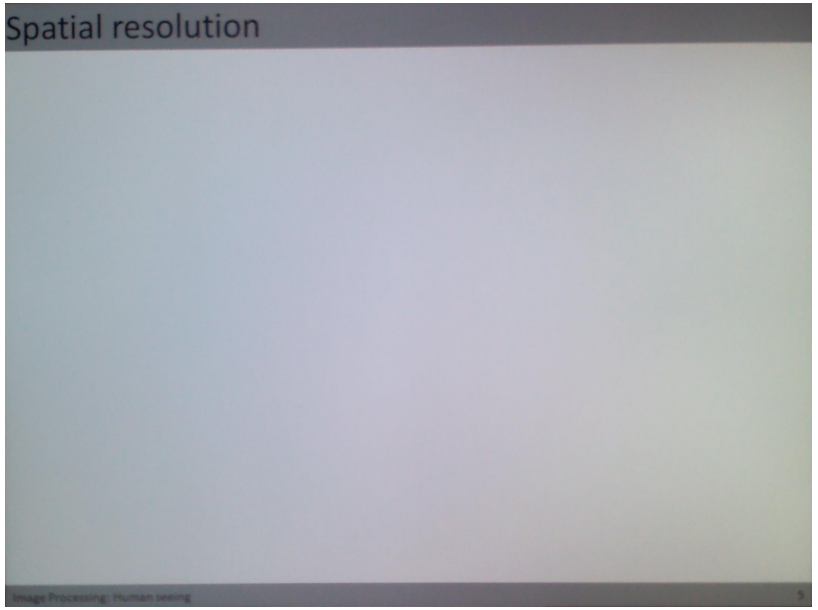
2MP Camera, far from the screen

Spatial resolution

5MP Camera, close to the screen

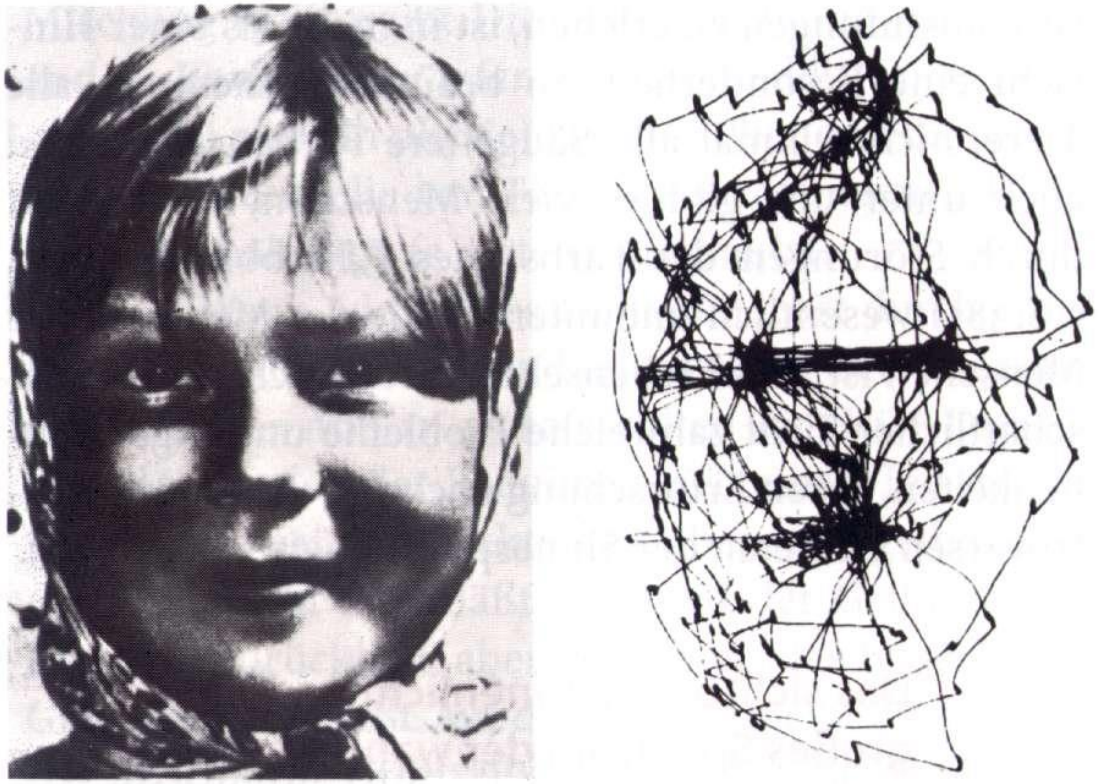
Spatial resolution (secrets)





- The resolution is much higher In fovea centralis
- The Information is pre-processed by Ganglion cells
  (Compare: 3072×2304=7MPixel, 2.4 MB RGB JPEG lossless)
- No still image, but a „Video“ (super-resolution)
- Scanning technique − Saccades

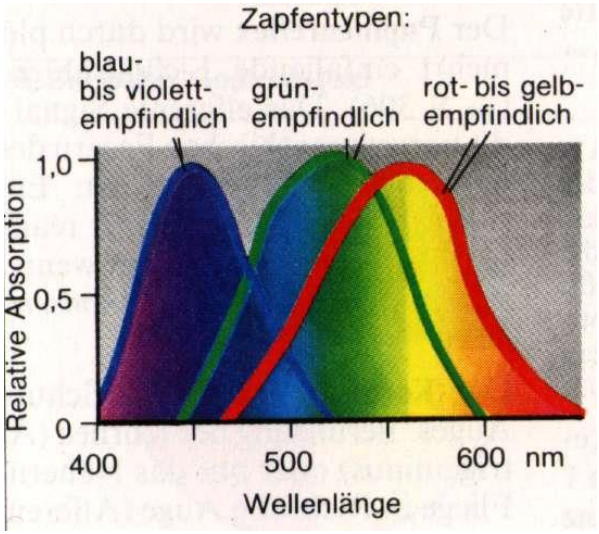
Eye Saccades

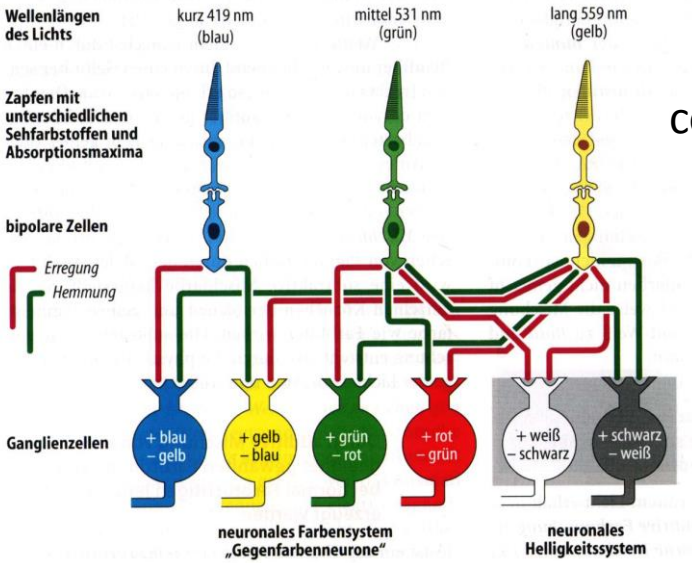


Eyes never move uniformly, but jump in **saccades**
(approximately 15-100 ms duration between fixation points)

Saccades are driven by the “importance” of the scene parts
(eyes, mouth etc).

The Human eye





cones “Zapfen”

What is light?

Spectrum, i.e. a function of the wavelength

Different colors can be computed by adding /subtracting signal of rods

Spectral resolution of the eye is relatively bad due to projection $\infty \rightarrow 3$

Roadmap this lecture

- Geometric primitives and transformations (sec. 2.1.1-2.1.4)
- Geometric image formation (sec 2.1.5, 2.1.6)
  - Pinhole camera
  - Lens effects
- Photometric image formation process (sec 2.2)
- The Human eye
- Camera Types and Hardware (sec 2.3)
- Appearance based matching

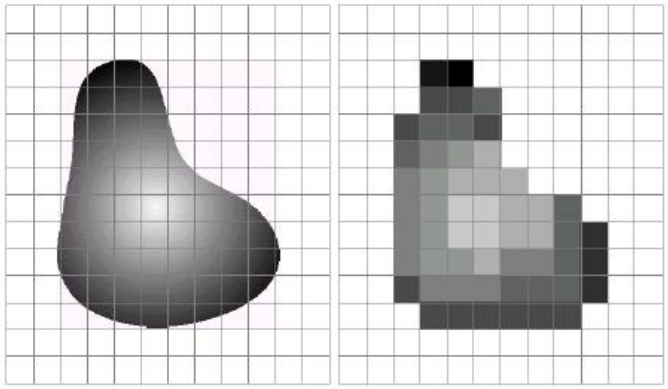
Camera types

- RGB cameras

- Depth cameras:
  - Passive RGB stereo
  - Active Structured light
  - Active Time of Flight

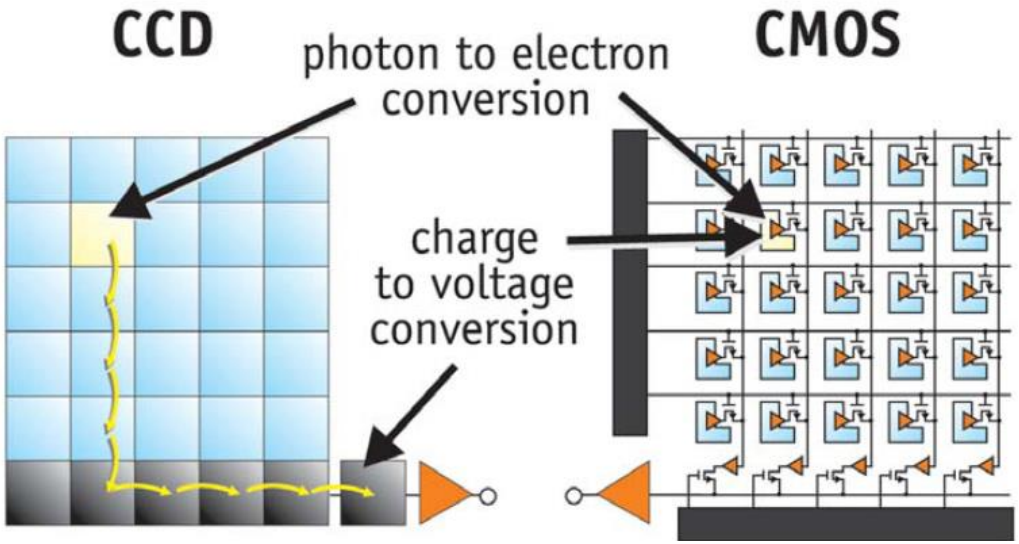
- Lightfield cameras

Digital RGB cameras



- A digital camera replaces film with a sensor array
- Each cell in the sensor array is light-sensitive diode that converts photons to electrons
- Two common types:
  - Charge Coupled Device (CCD)
  - Complementary metal oxide semiconductor (CMOS)

CCD versus CMOS

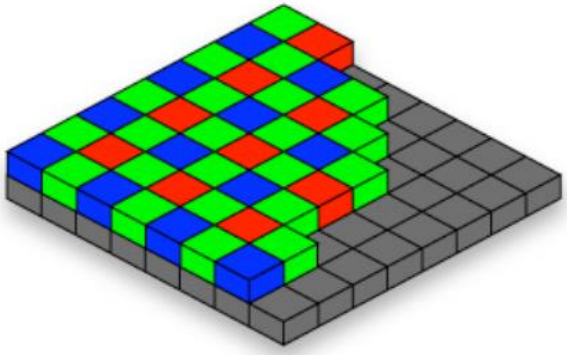


- CCD: move charge from pixel to pixel and then converts to voltage and digital signal
  - Negative: more expensive
- CMOS: converts to voltage inside the pixel
  - Negative: slower to read out image ("rolling shutter" effect)

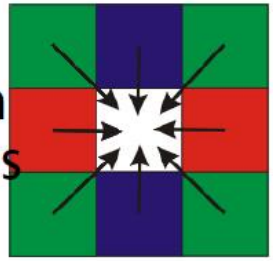
See details on: http://www.dalsa.com/shared/content/pdfs/CCD_vs_CMOS_Litwiller_2005.pdf

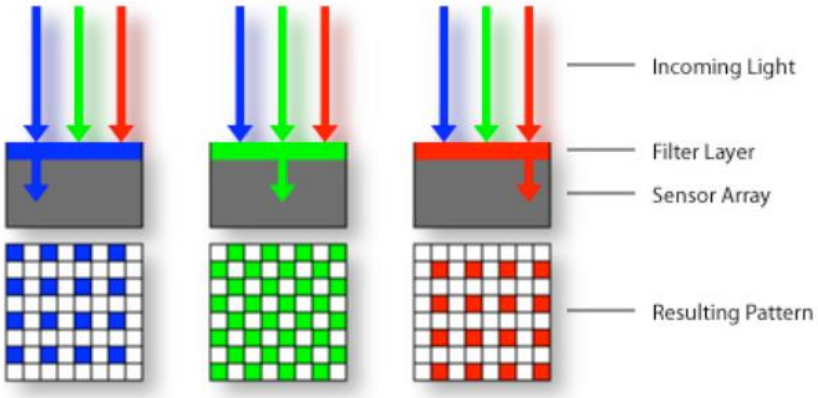
Color - Filters

Bayer grid

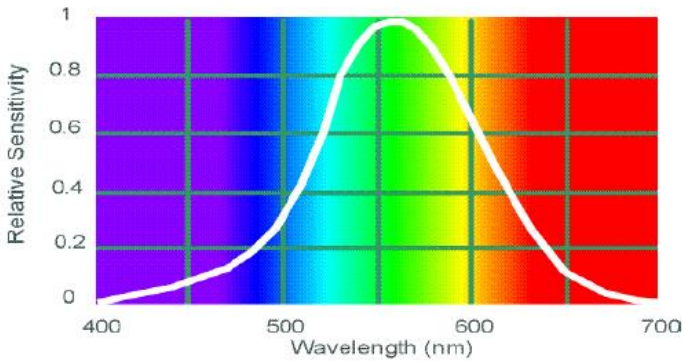


Estimate missing components from neighboring values (demosaicing)





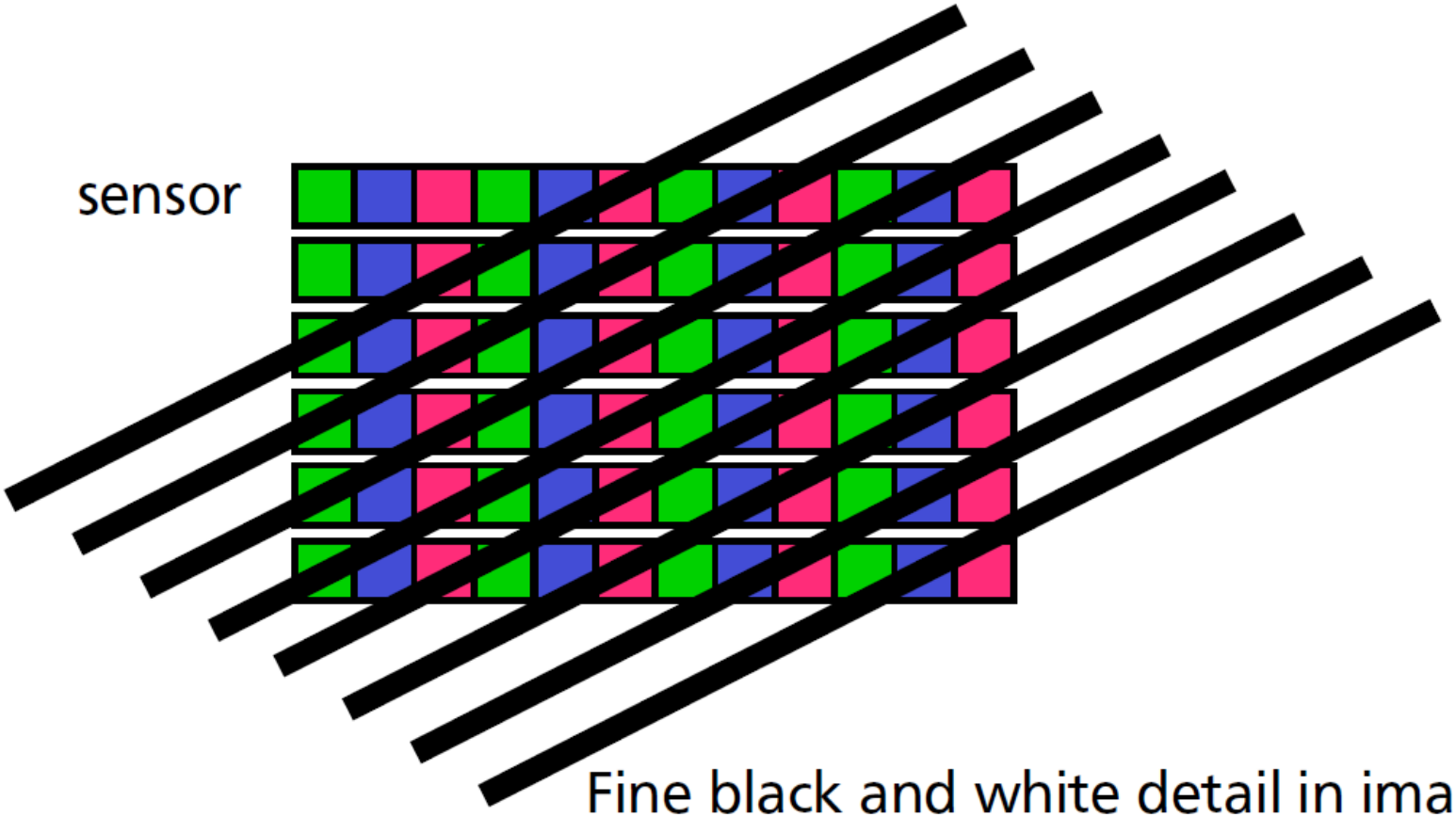
Why more green?



Human Luminance Sensitivity Function

Problem with de-mosaicing: color moiré

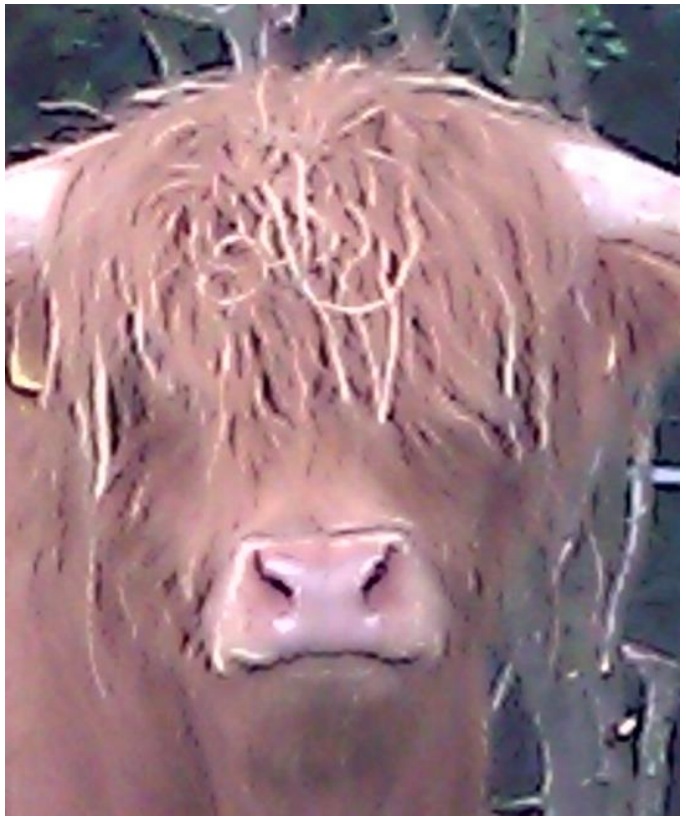
Cause of color moiré



Fine black and white detail in image misinterpreted as color information

General problems with RGB cameras

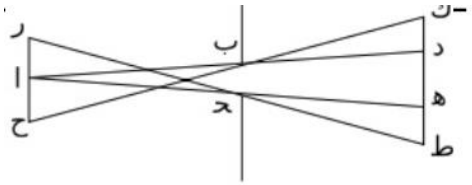
- **Noise**
  - low light is where you most notice noise
  - light sensitivity (ISO) / noise tradeoff
  - stuck pixels
- **Resolution: Are more megapixels better?**
  - requires higher quality lens
  - noise issues
- **In-camera processing**
  - oversharpening can produce halos
- **RAW vs. compressed**
  - file size vs. quality tradeoff
- **Blooming**
  - charge overflowing into neighboring pixels
- **More info online:**
  - http://electronics.howstuffworks.com/
  - http://www.dpreview.com/
  - http://www.dxomark.com/



In-camera processing

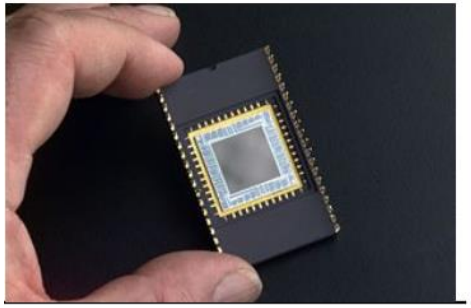
Historical context of cameras

- Pinhole model: Mozi (470-390=1 BCE), Aristoteles (384-322 BCE)
- Principles of optics (including lenses): Alhazen (965-1039 CE)
- Camera obscura: Leonardo da Vinci (1452-1519), Johann Zahn (1631-1707)
- First photo: Joseph Nicephore Niepce (1822)
- Daguerréotypes (1839)
- Photographic film (Eastman, 1889)
- Cinema (Lumière Brothers, 1895)
- Color Photography (Lumière Brothers, 1908)
- Television (Baird, Farnsworth, Zworykin, 1920s)
- First consumer camera with CCD: Sony Mavica (1981)
- First fully digital camera: Kodak DCS100 (1990)



Alhazen's notes

Niepce, "La Table Servie," 1822



Camera types

- RGB cameras

- Depth cameras:
  - Passive RGB stereo
  - Active Structured light
  - Active Time of Flight

- Lightfield cameras

Passive Depth Camera –RGB stereo

Easy to match

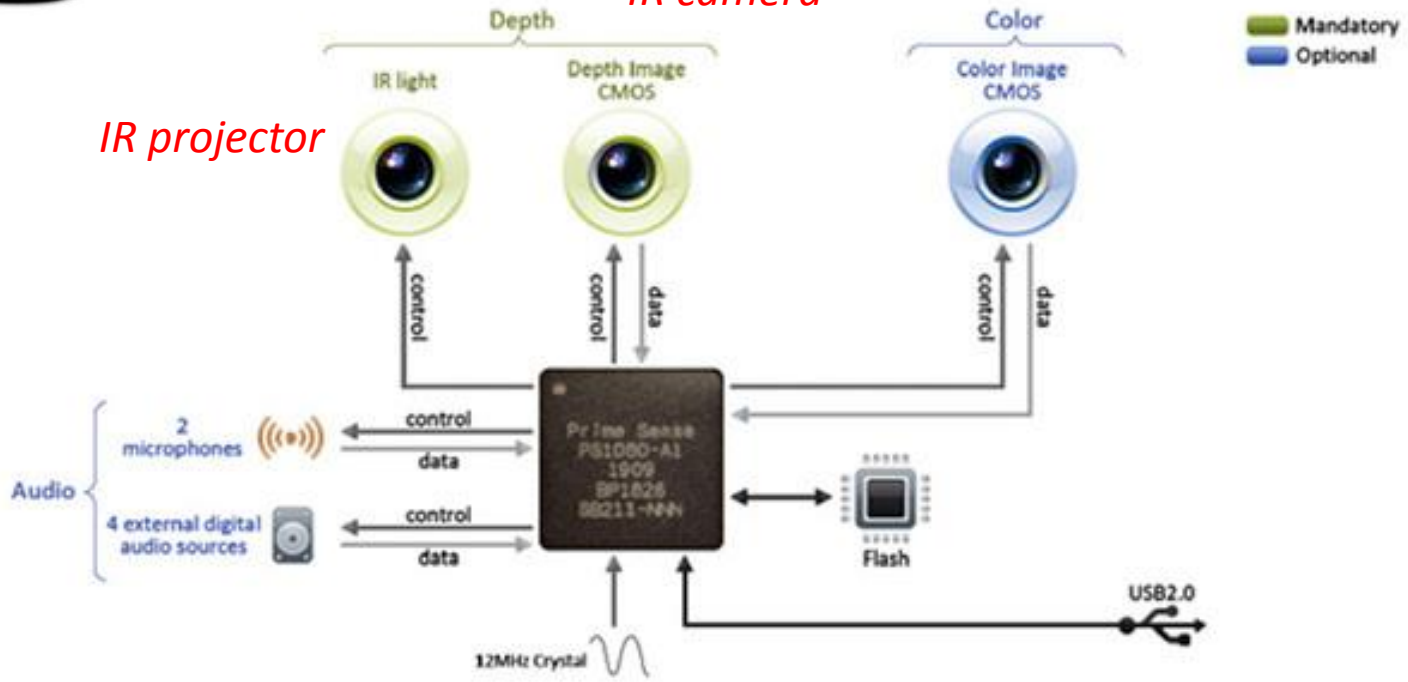
hard to match

Active Depth Camera – structured light

KINECT
for XBOX 360

IR camera

IR projector



Depth Camera – structured light

Reference image from IR projector

- We have to perform matching (as with passive stereo camera) but now we have a very textured image
- Only works well when external IR light is not too string (not under sunlight)

Depth Camera – structure Light

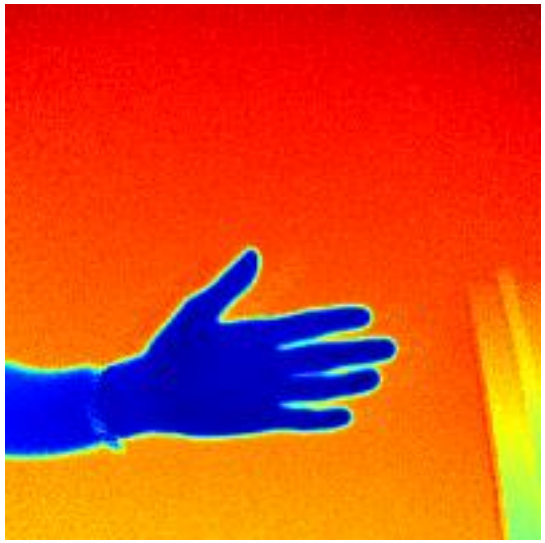
Halfway

3 Minutes Break

Question?

Depth camera – time of flight

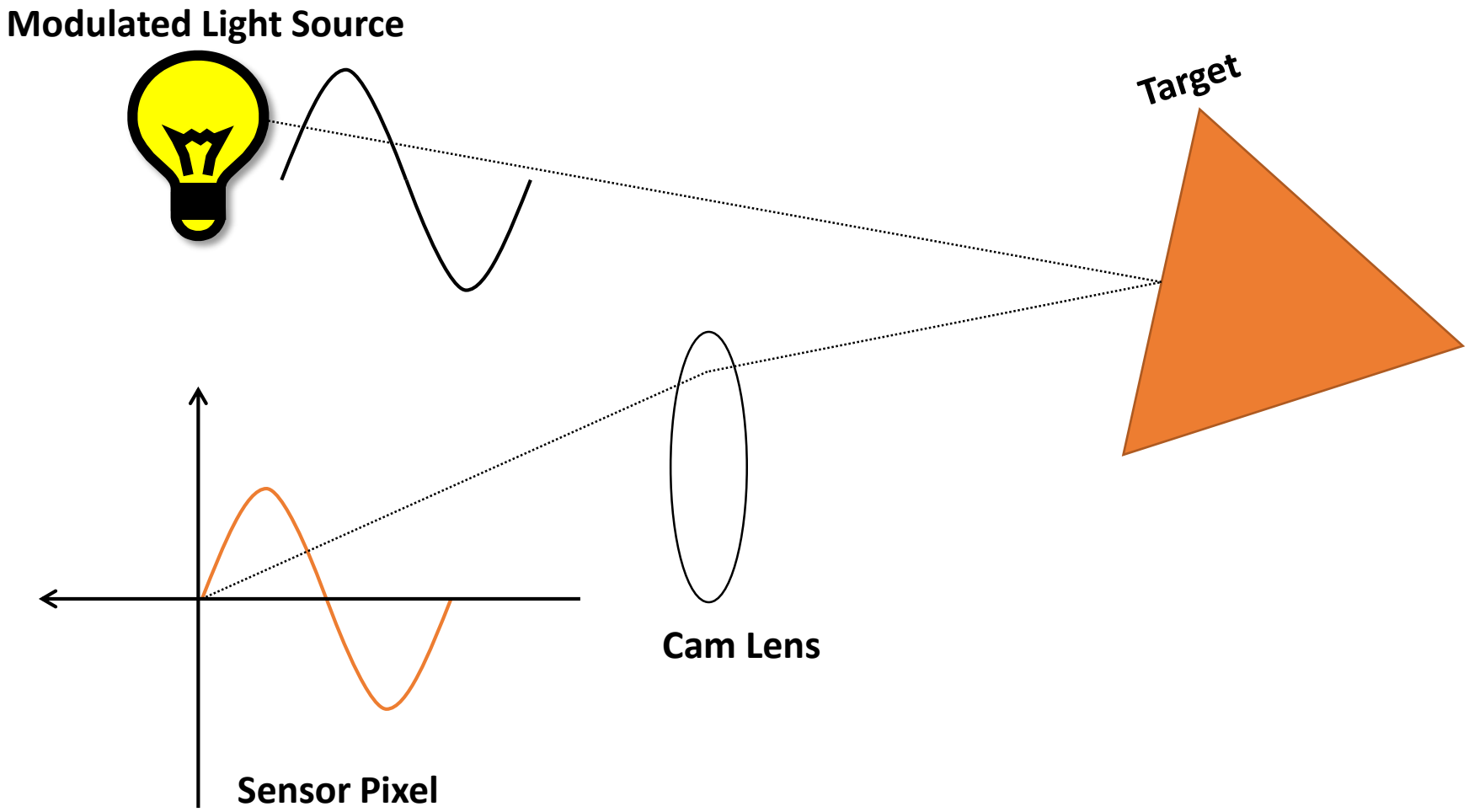
Intensity image



Depth Image

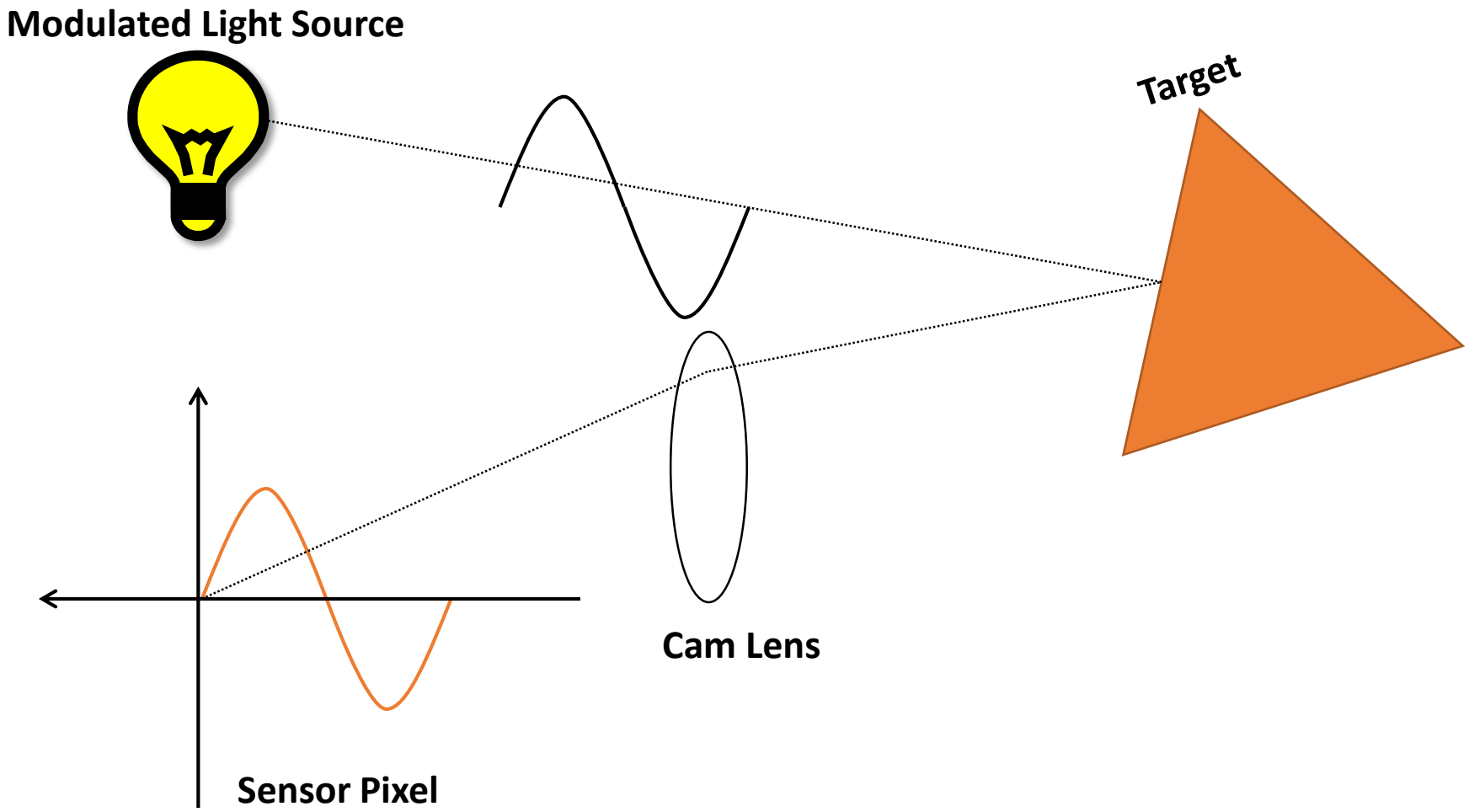
PMD Camera

Principle Time of Flight



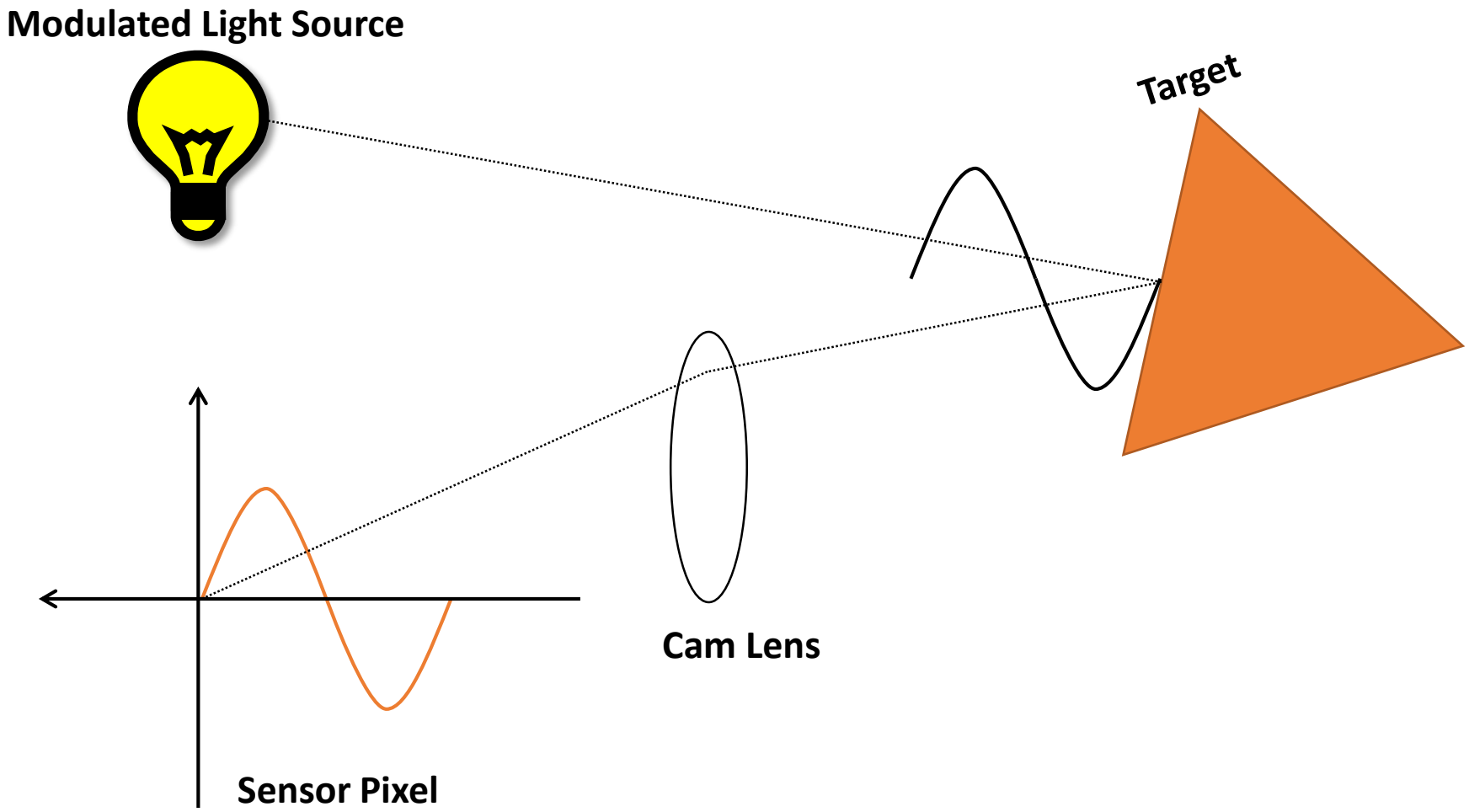
[slide credits: Rahul Nair, Daniel Kondermann]

Principle Time of Flight



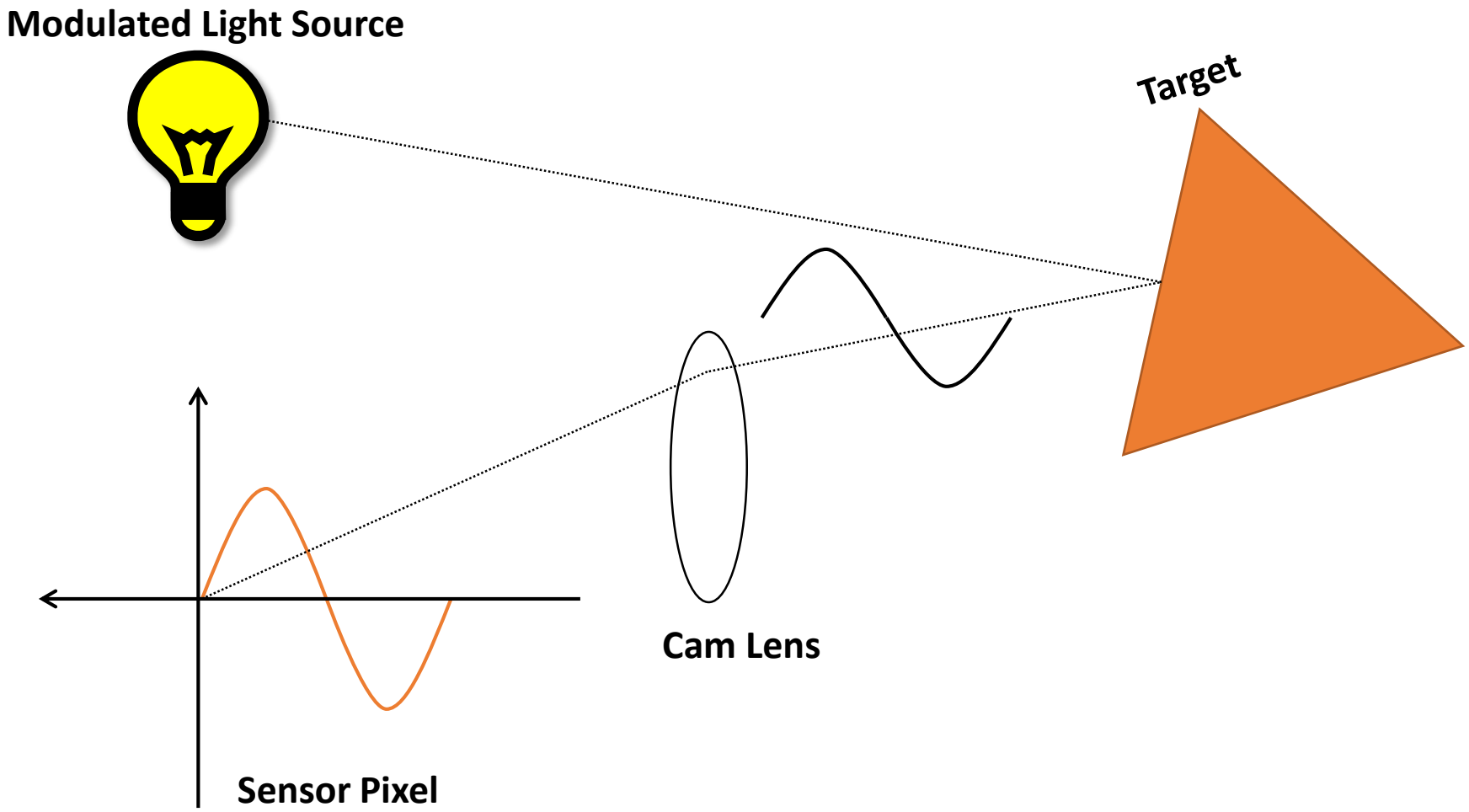
[slide credits: Rahul Nair, Daniel Kondermann]

Principle Time of Flight



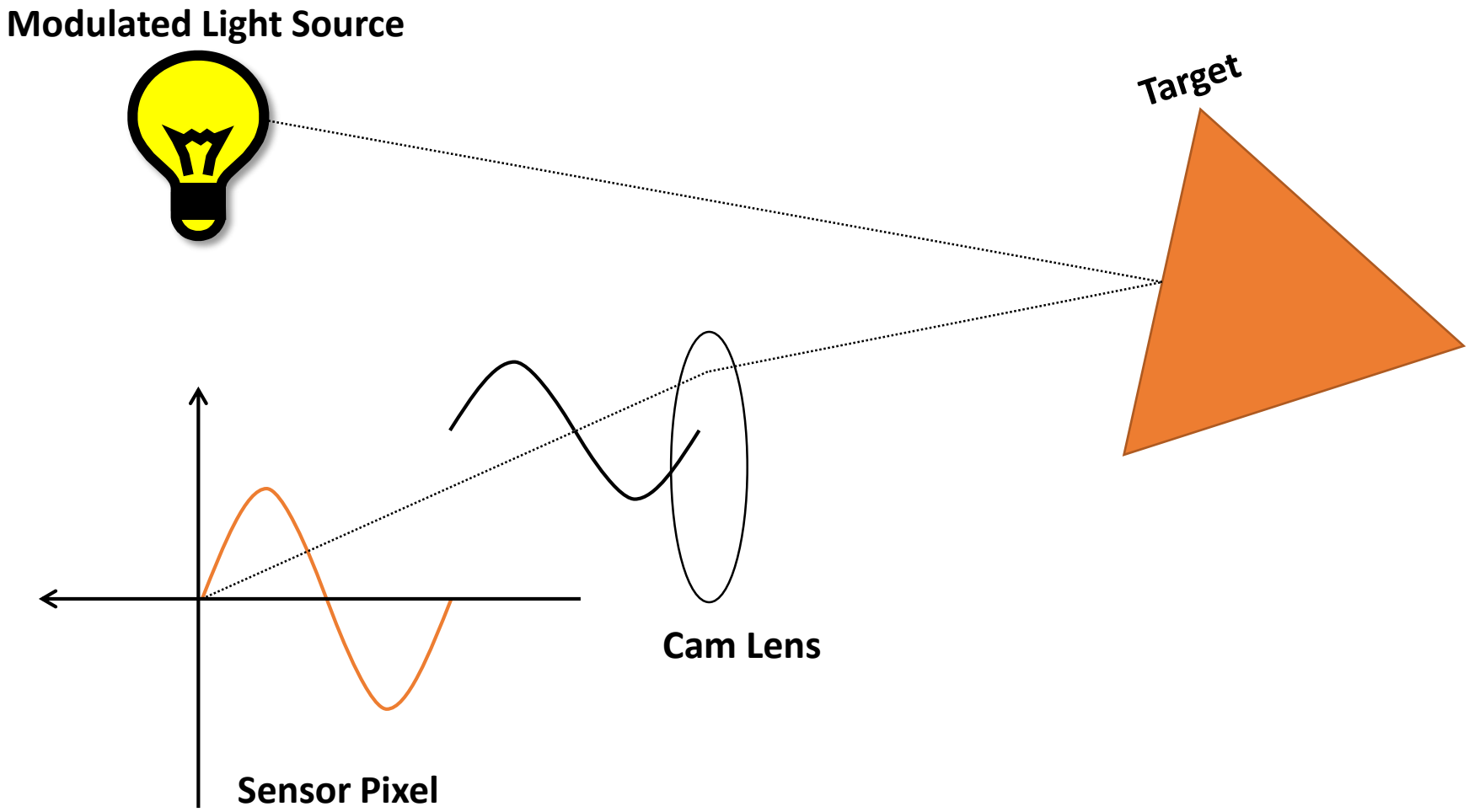
[slide credits: Rahul Nair, Daniel Kondermann]

Principle Time of Flight



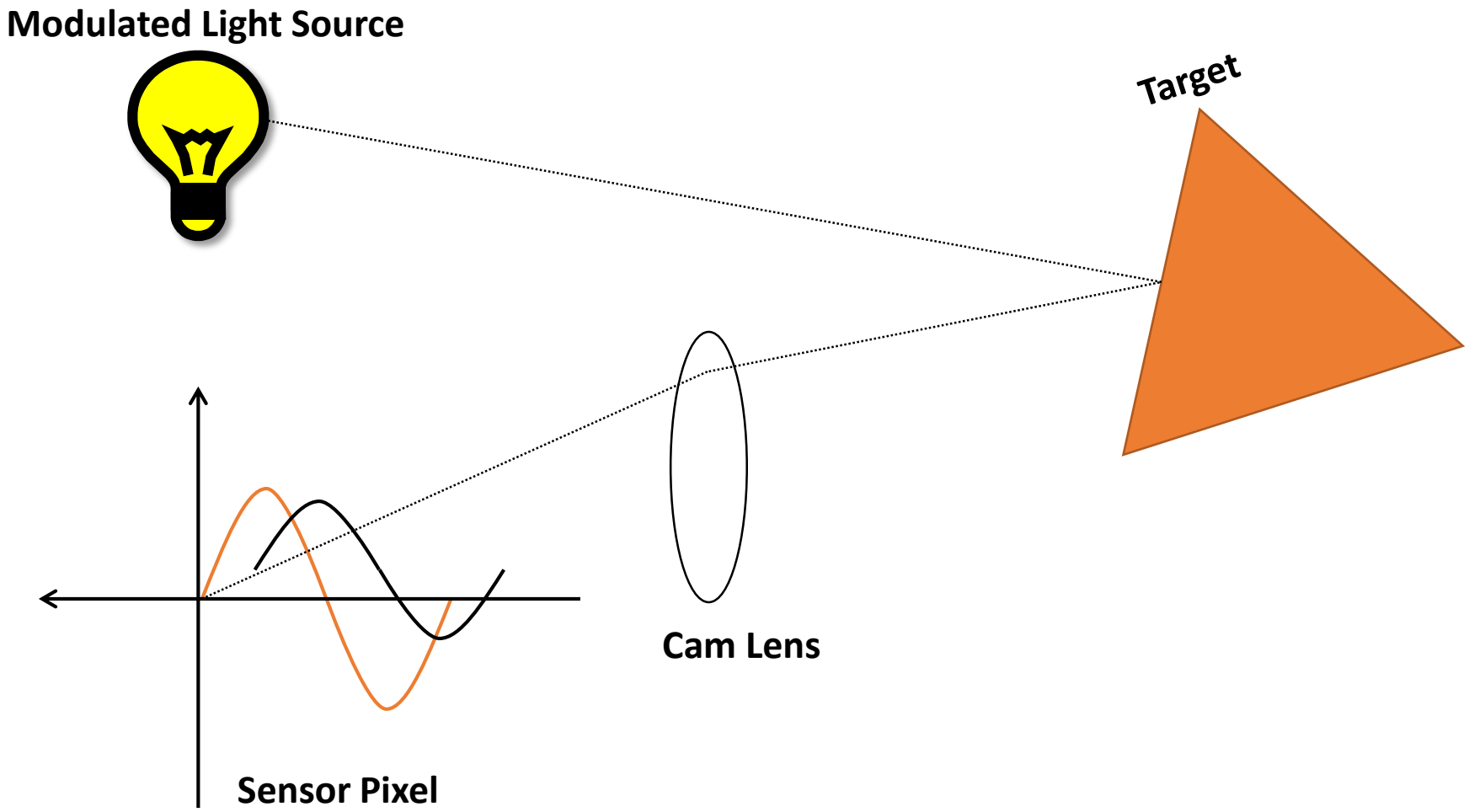
[slide credits: Rahul Nair, Daniel Kondermann]

Principle Time of Flight



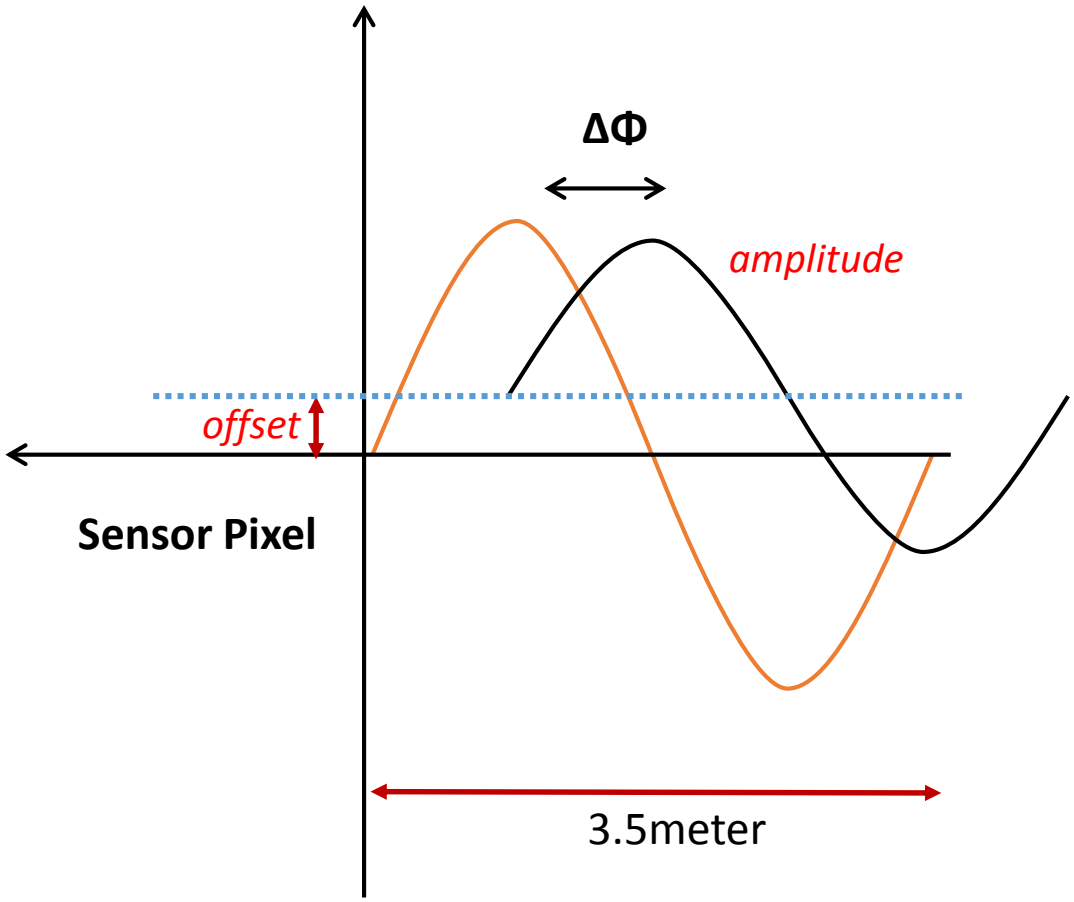
[slide credits: Rahul Nair, Daniel Kondermann]

Principle Time of Flight



[slide credits: Rahul Nair, Daniel Kondermann]

Principle Time of Flight



ΔΦ means d= 0.3m or 3.8m or 7.3m or …
(take differently modulated frequences)

Measure:

- Phase shift
- Amplitude and offset

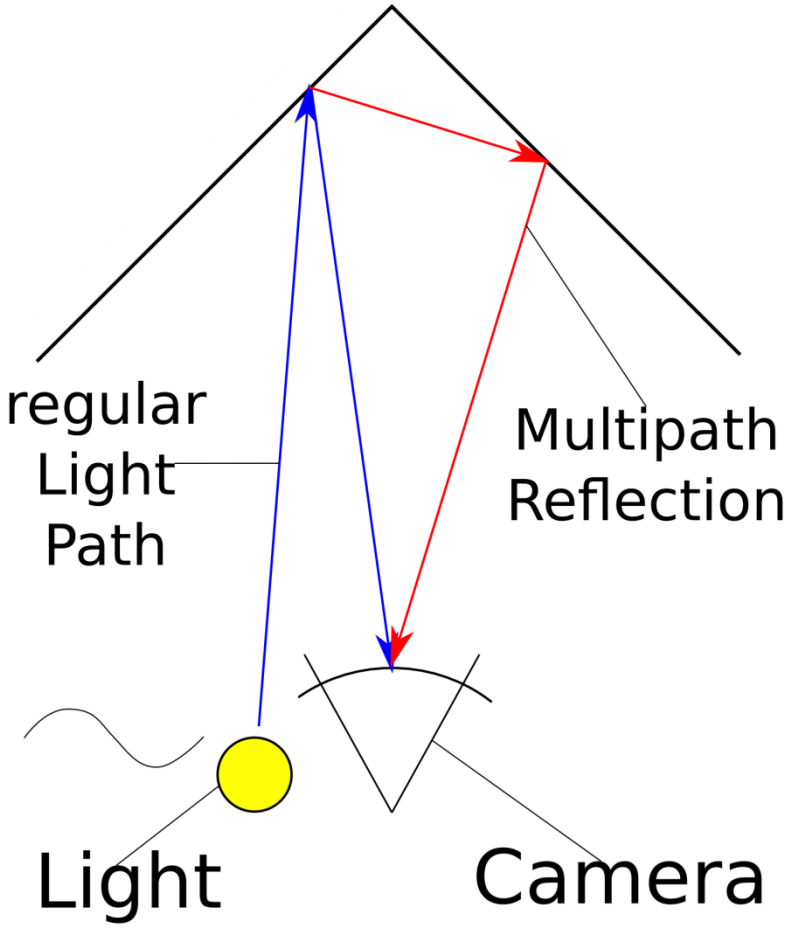
Output:

- **Depth Image**
  (from phase shift)
  (wavelength determines the range of depth values; around 3.5 meter)
- **Intensity image** (from amplitude and offset)

[slide credits: Rahul Nair, Daniel Kondermann]

Depth camera – time of flight

One of the biggest problems is multi-path



Lightfield cameras

Capture all light:

http://www.lytro.com/camera/

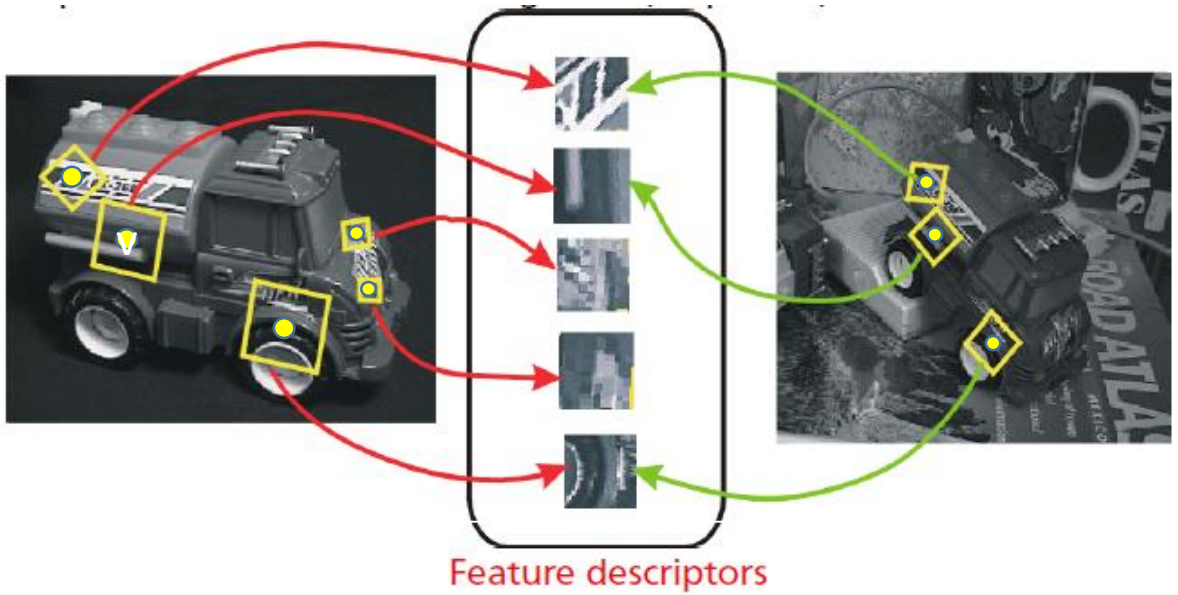
Refocus and change perspective with one image

Roadmap this lecture

- Geometric primitives and transformations (sec. 2.1.1-2.1.4)
- Geometric image formation (sec 2.1.5, 2.1.6)
  - Pinhole camera
  - Lens effects
- Photometric image formation process (sec 2.2)
- The Human eye
- Camera Types and Hardware (sec 2.3)
- Appearance based matching

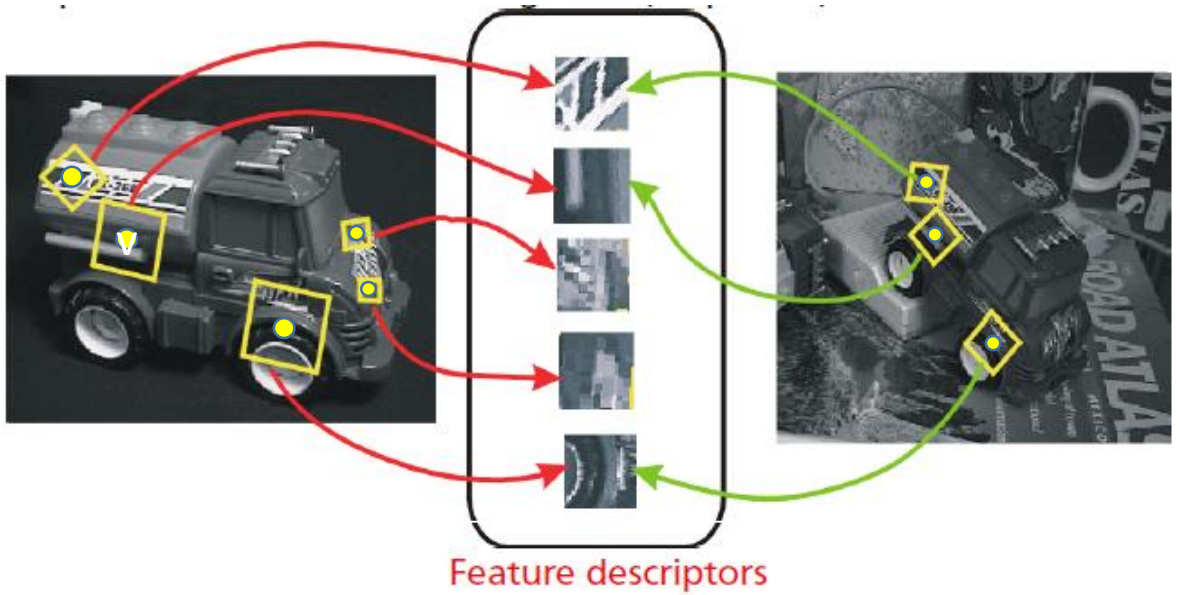
Roadmap: matching 2 Images (appearance & geometry)

- Find interest points
- Find orientated patches around interest points to capture appearance
- Encode patch appearance in a descriptor
- Find matching patches according to appearance (similar descriptors)
- Verify matching patches according to geometry

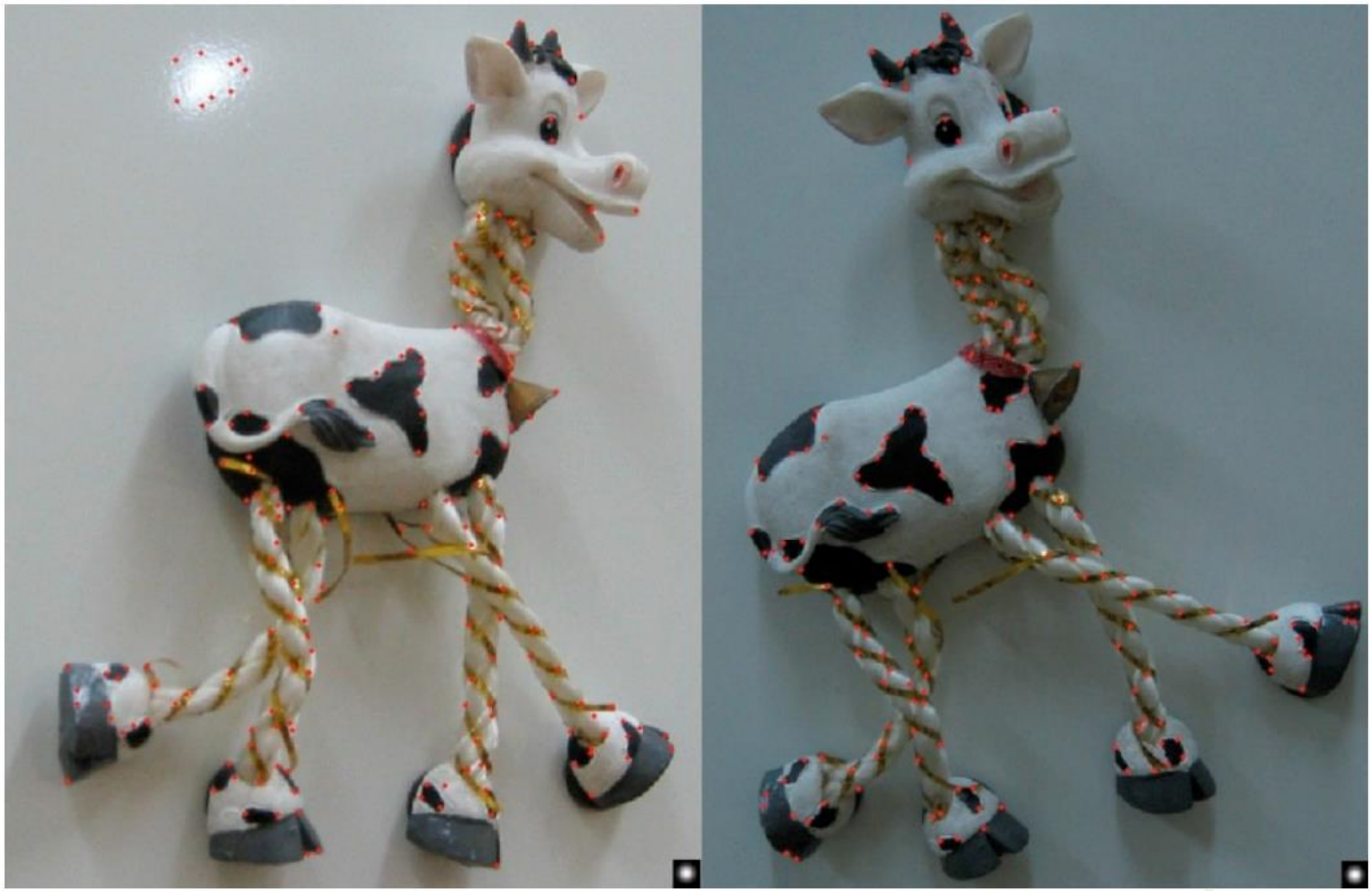


Roadmap: matching 2 Images (appearance & geometry)

- Find interest points
- Find orientated patches around interest points to capture appearance
- Encode patch appearance in a descriptor
- Find matching patches according to appearance (similar descriptors)
- Verify matching patches according to geometry



Reminder Lecture 3: Harris Corner Detector



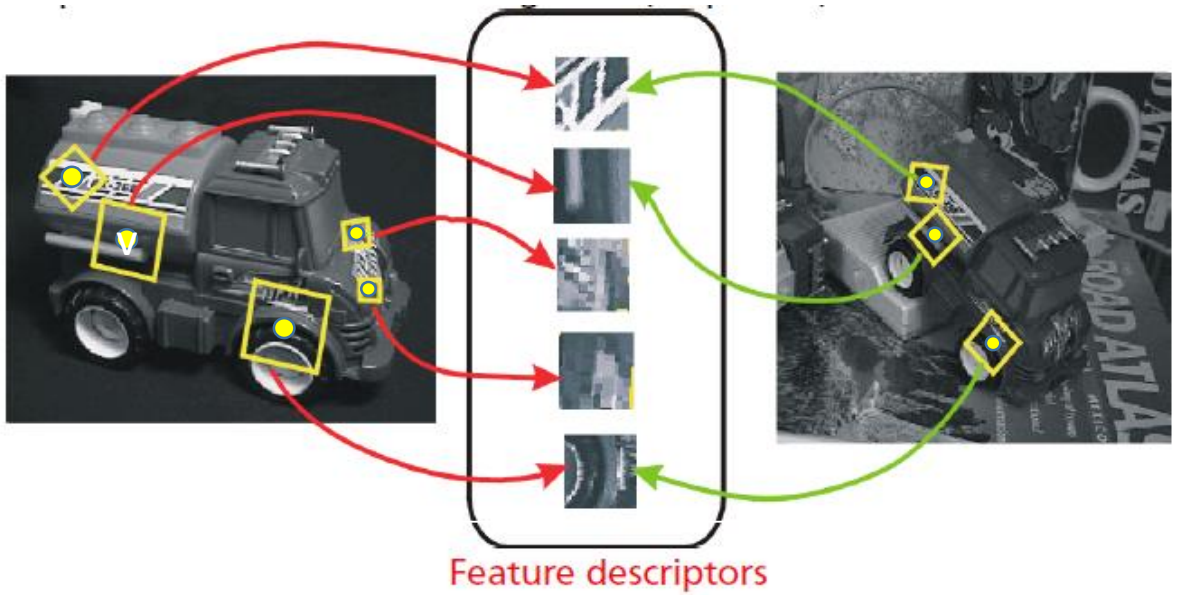
Auto-correlation function: $c(x, y, \Delta x, \Delta y) \approx [\Delta x, \Delta y]\, Q(x, y) \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$

$$\lambda_1 \lambda_2 = \det Q(x, y) = AC - B^2, \quad \lambda_1 + \lambda_2 = \text{trace} Q(x, y) = A + C$$

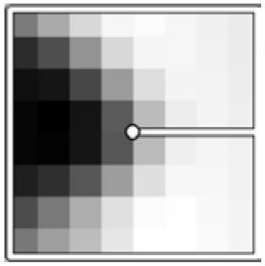
Harris measure: $H = \lambda_1 \lambda_2 - 0.04(\lambda_1 + \lambda_2)^2$

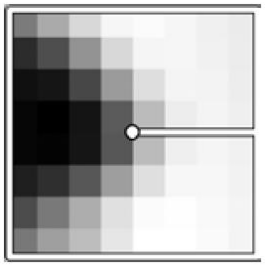
Roadmap: matching 2 Images (appearance & geometry)

- Find interest points
- Find orientated patches around interest points to capture appearance
- Encode patch appearance in a descriptor
- Find matching patches according to appearance (similar descriptors)
- Verify matching patches according to geometry



How to deal with orientation

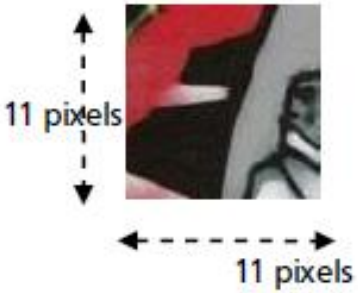


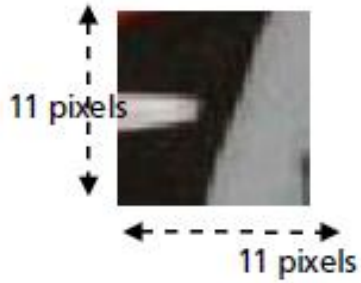
Orientate with image gradient:

$$\nabla I = (I_x, I_y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

$$\theta = \text{atan}(I_y, I_x)$$

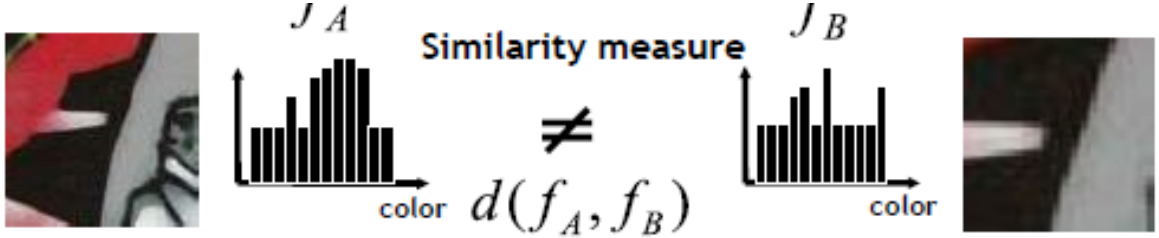
Choose a patch around each point





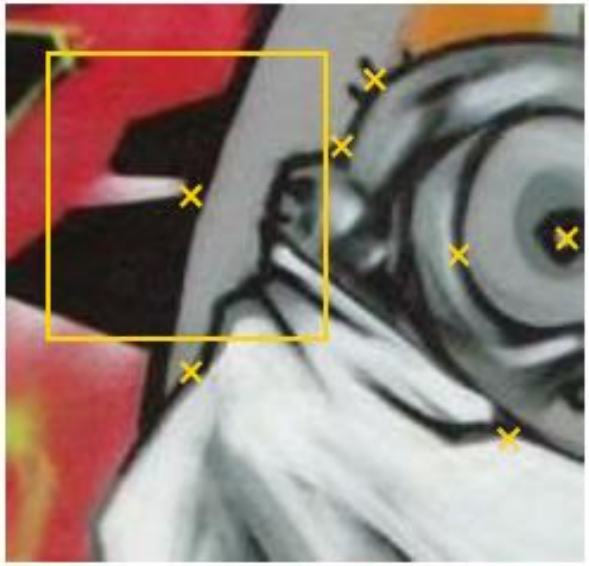
How to deal with scale?

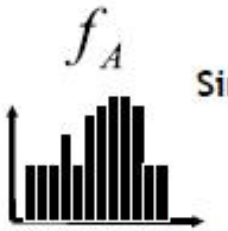
Choose a patch around each point



How to deal with scale?

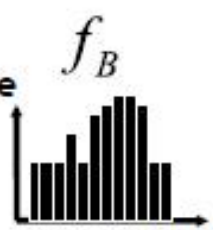
Choose a patch around each point





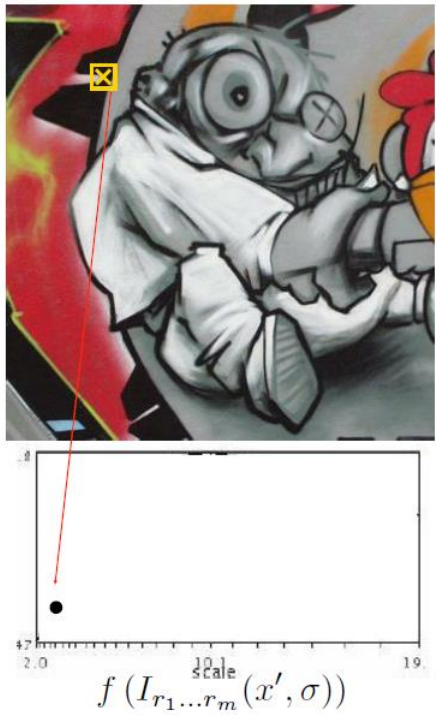
Similarity measure

$= d(f_A, f_B)$



How to deal with scale?

Scale selection (illustration)



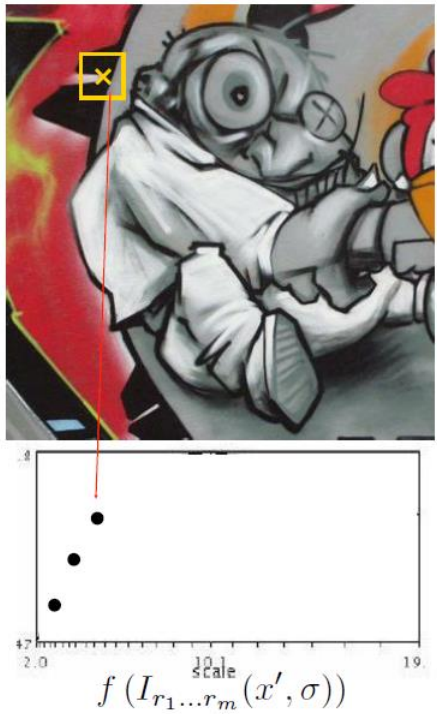
f *is Laplacian of Gaussian (LoG) operator.*
Measures an average edge-ness in all directions

$$\nabla^2(G(\sigma) * I) = \frac{\partial^2\,(G(\sigma) * I)}{\partial x^2} + \frac{\partial^2\,(G(\sigma) * I)}{\partial y^2}$$

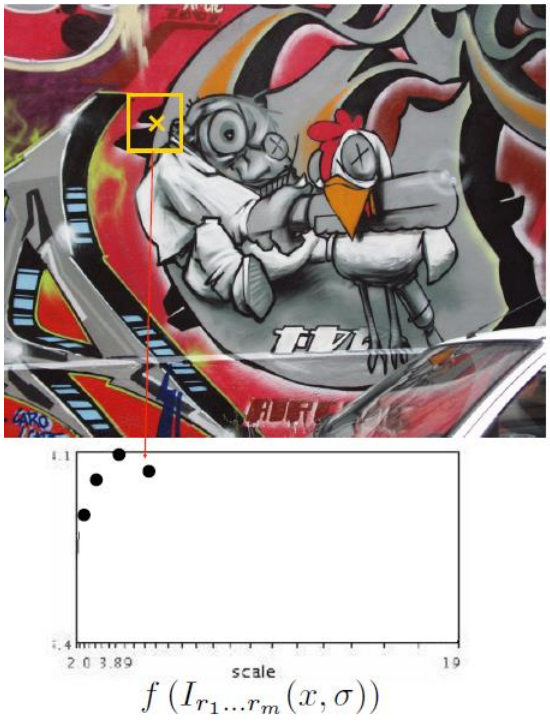
(details on page 191)

Scale selection (illustration)

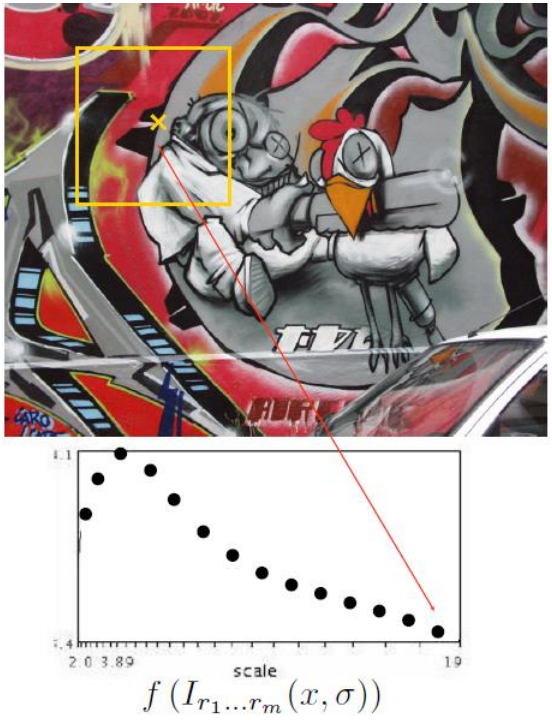
Scale selection (illustration)

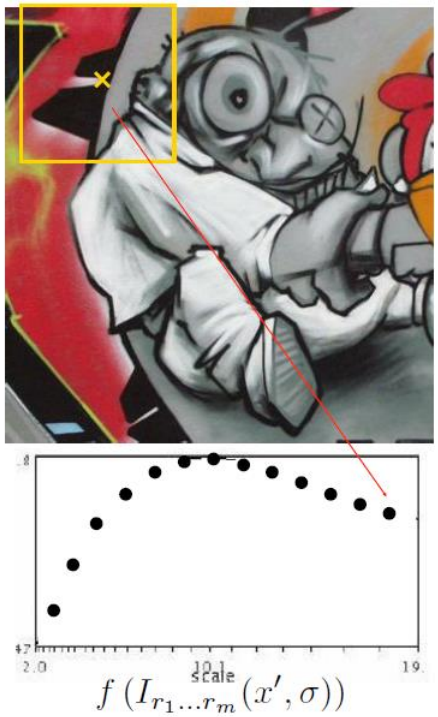


Scale selection (illustration)



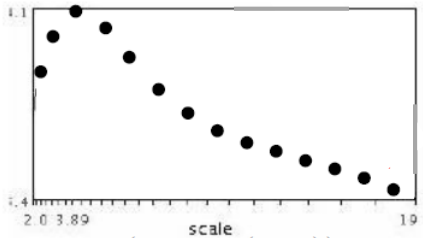
Scale selection (illustration)



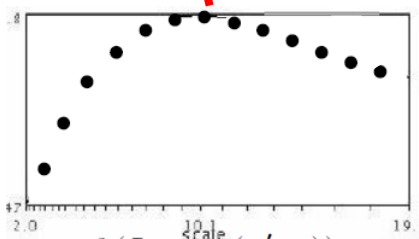


We could match-up these curves and find unique corresponding points

Scale selection (illustration)



$$f\left(I_{r_1 \ldots r_m}(x, \sigma)\right)$$



$$f\left(I_{r_1 \ldots r_m}(x', \sigma)\right)$$

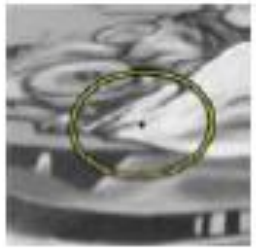
Simpler: Find maxima /minima in image

Extensions: general affine transformations

$$\boldsymbol{x}_0 \rightarrow \boldsymbol{A}_0^{-1/2}\boldsymbol{x}'_0$$

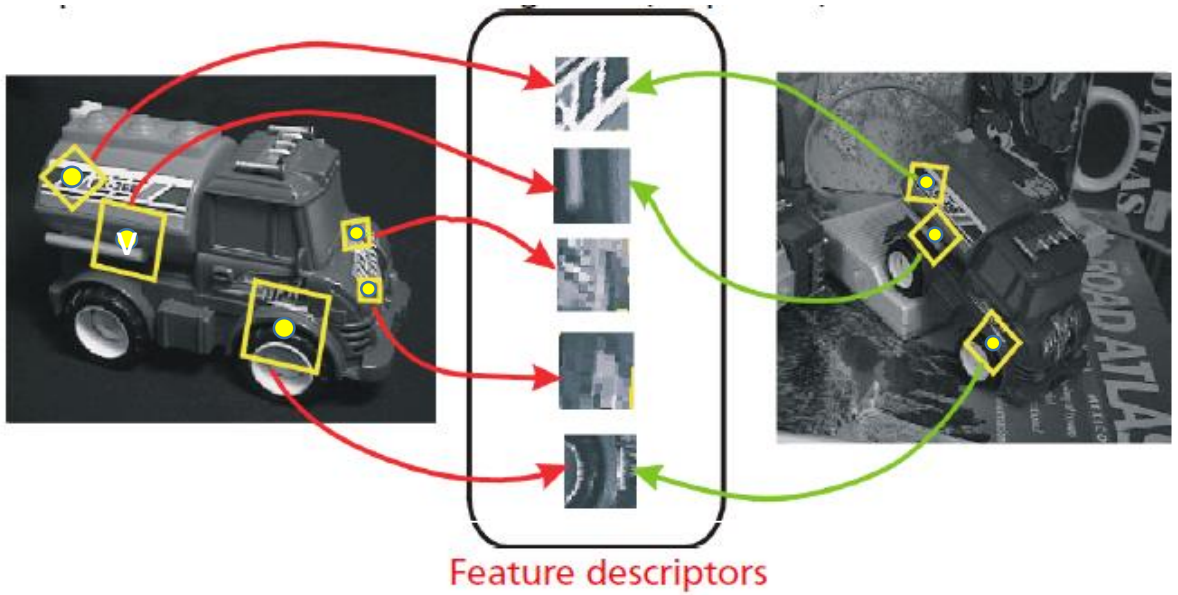
$$\boldsymbol{x}'_0 \rightarrow \boldsymbol{R}\boldsymbol{x}'_1$$

$$\boldsymbol{A}_1^{-1/2}\boldsymbol{x}'_1 \leftarrow \boldsymbol{x}_1$$

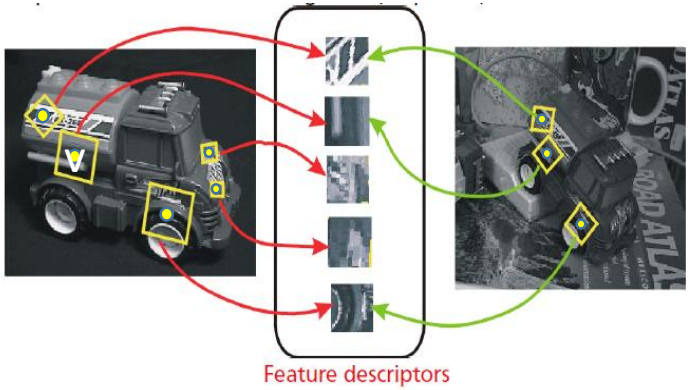


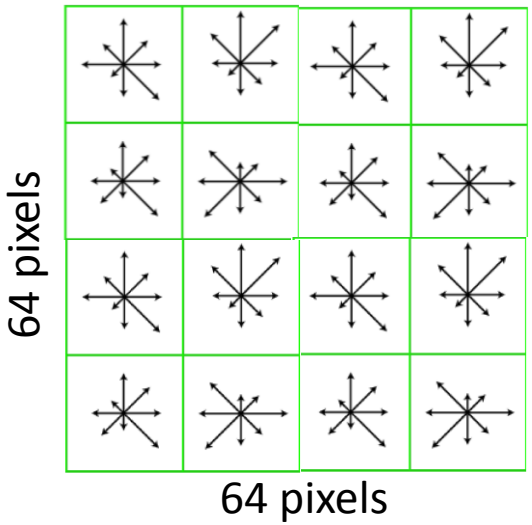
Roadmap: matching 2 Images (appearance & geometry)

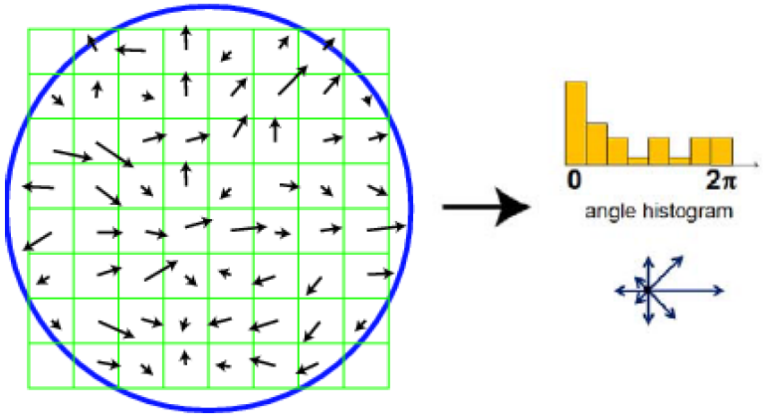
- Find interest points
- Find orientated patches around interest points to capture appearance
- Encode patch appearance in a descriptor
- Find matching patches according to appearance (similar descriptors)
- Verify matching patches according to geometry



SIFT feature







- 4*4=16 cells
- Each cell has an 8 bin histogram (smoothed across cells)
- In total: 16*8 values, i.e. 128D vector

A cell has 16x16 pixels
(here 8x8 for illustration only)
(blue circle shows center weighting)

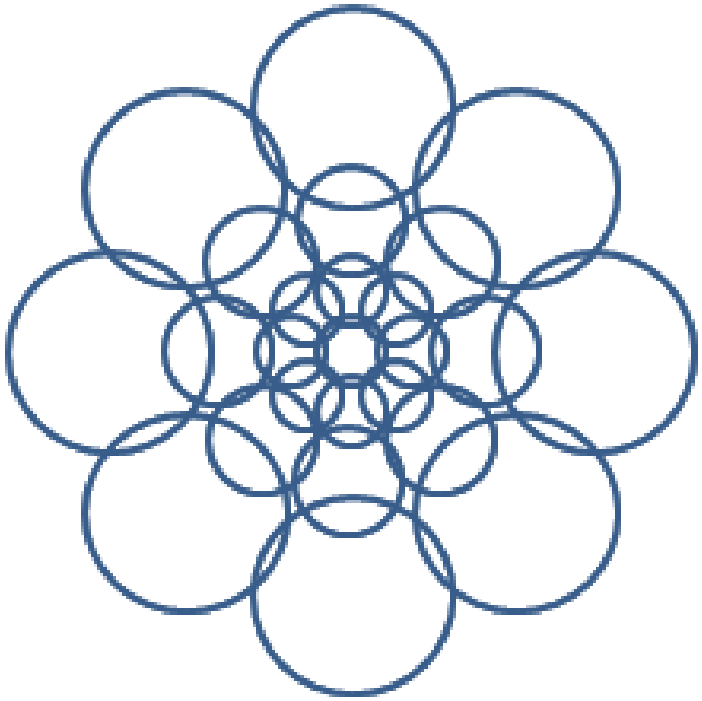
[Lowe 2004]

SIFT feature is very popular

- Fast to compute
- Can handle large changes in viewpoint well (up to 60^o out of plane rotation)
- Can handle photometric changes (even day versus night)

Many other feature descriptors

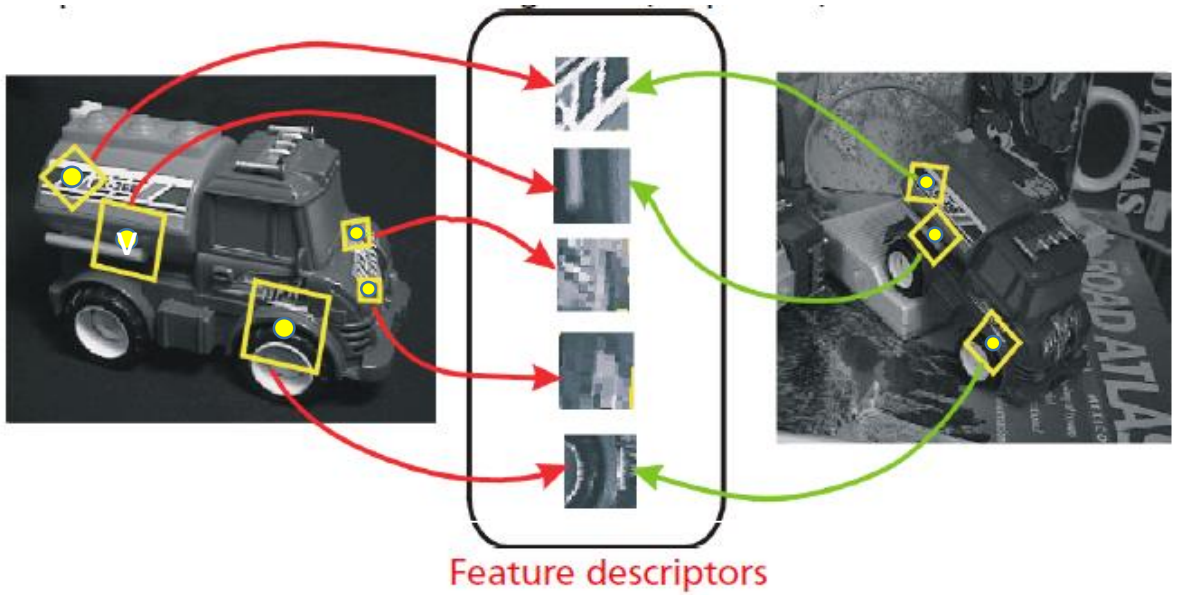
- MOPS [Brown, Szeliski and Winder 2005]
- SURF [Herbert Bay et al. 2006]
- DAISY [Tola, Lepetit, Fua 2010]
- Shape Context
- ….



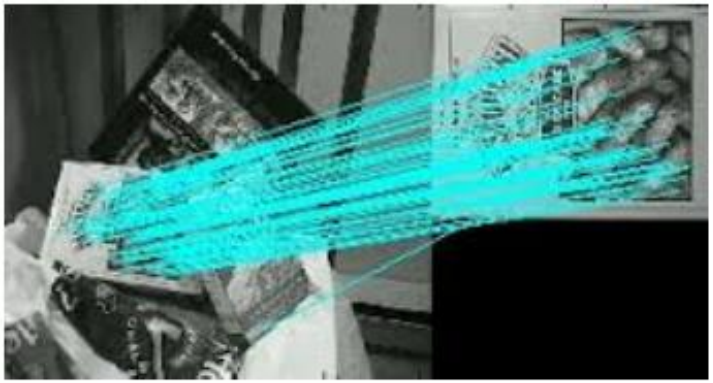
DAISY

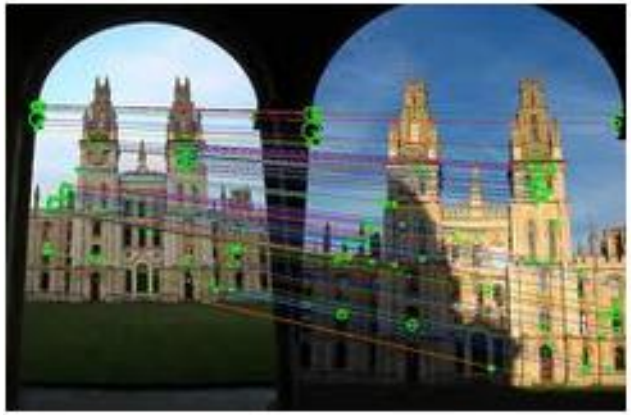
Roadmap: matching 2 Images (appearance & geometry)

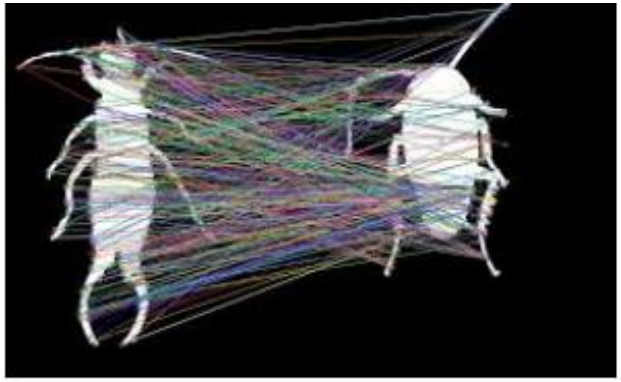
- Find interest points
- Find orientated patches around interest points to capture appearance
- Encode patch appearance in a descriptor
- Find matching patches according to appearance (similar descriptors)
- Verify matching patches according to geometry

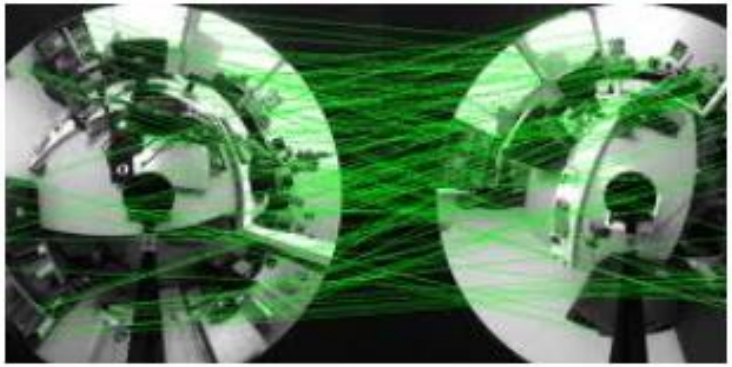


Appearance-based matching

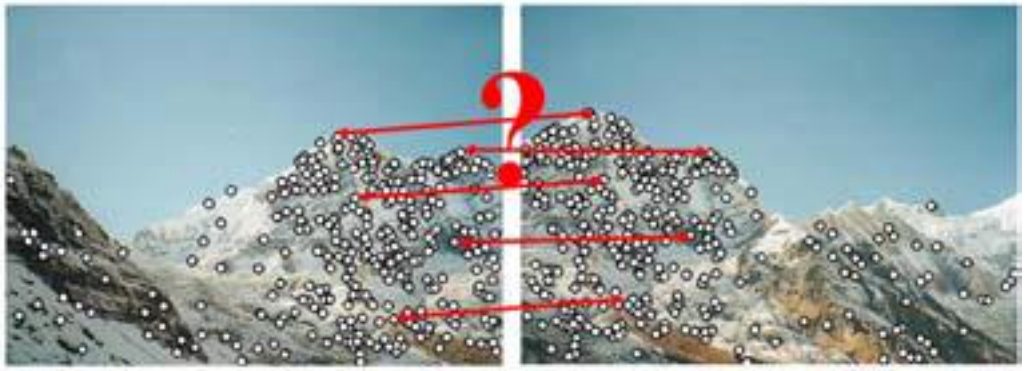








Appearance-based matching

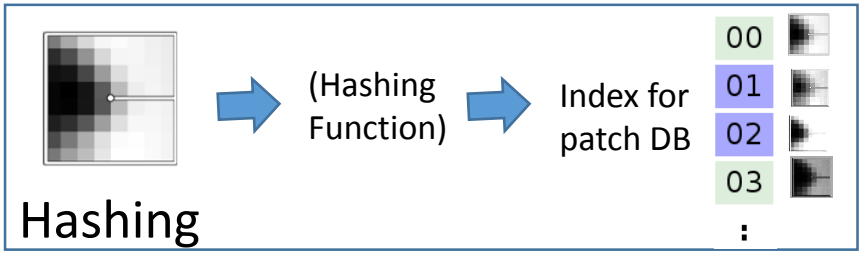


N patches
(e.g. N = 1000)

N patches
(e.g. N = 1000)

Goal:

1) Find for each patch in left image the closest in right image
2) Accept all matches where descriptors are similar enough



Methods:

- Naïve: N^2 tests (here 1 Million)
- Hashing (locality sensitive hashing)
- Kd-tree; on average NlogN tests (here 10,000)

Subtask: Search for one patch

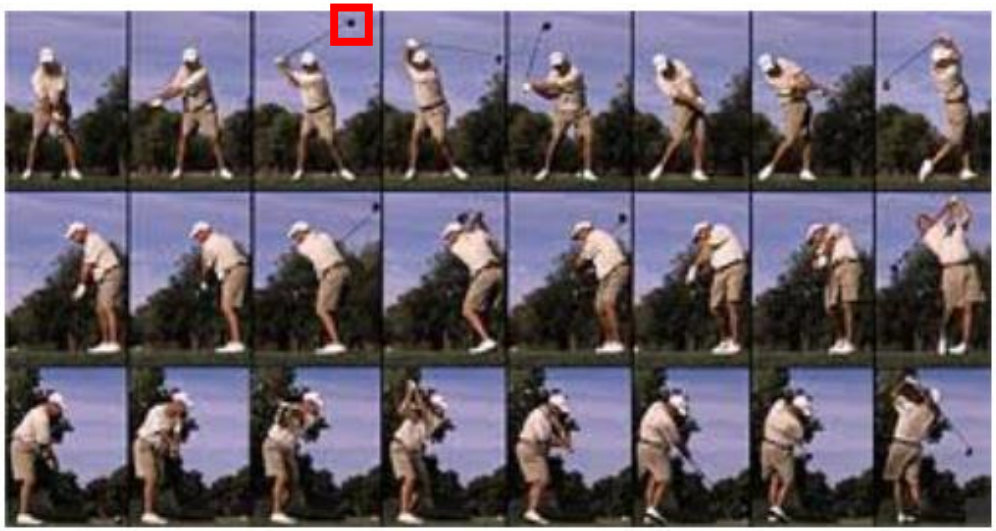
Query patch

?

Database image

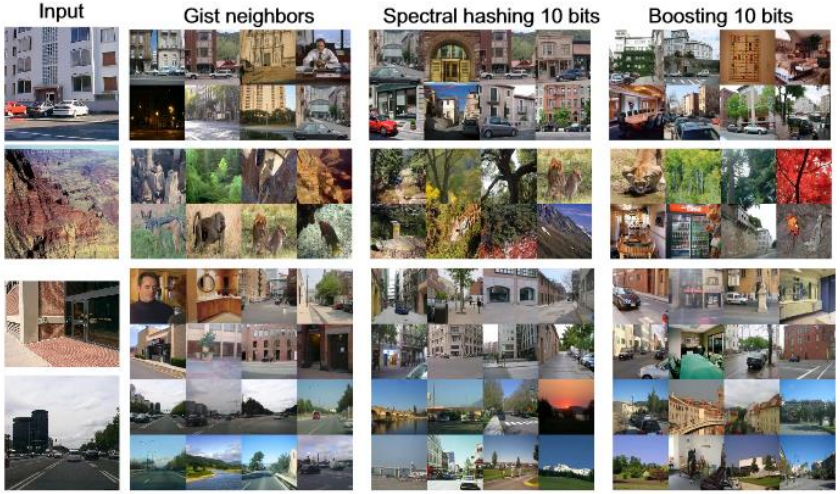
Nearest Neighbor Search

- Tracking in Video



The video is the Database

- Image retrieval



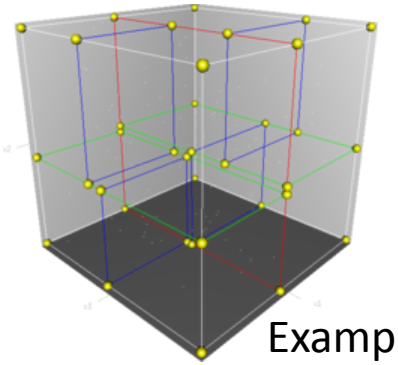
- Whole image has one descriptor
- Database is an image collection

Kd-tree (d stands for dimension)

Query patch

?

Database image



Example in 3D

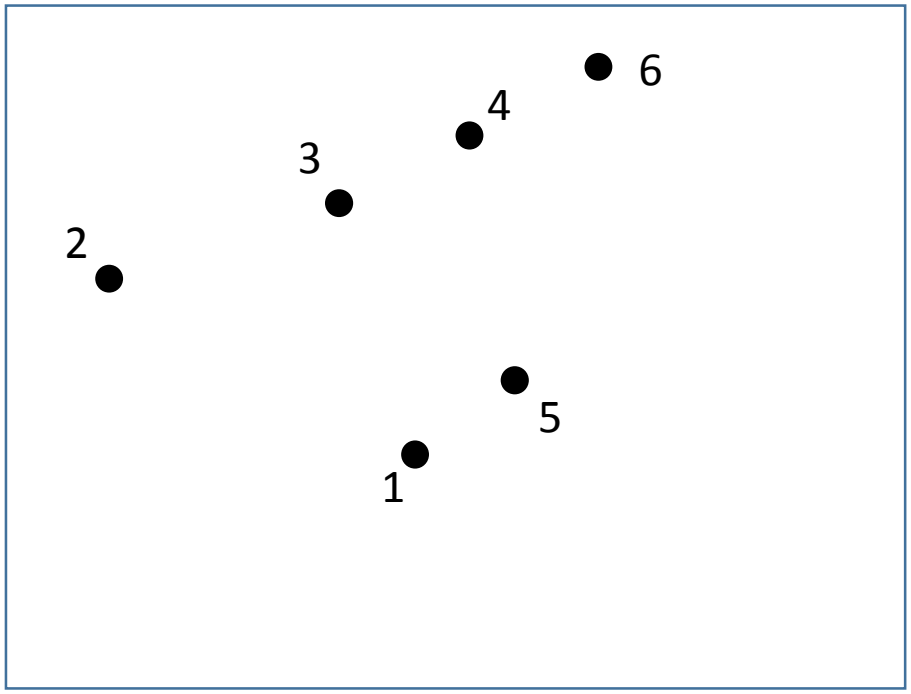
- Build the tree over database image:
  1) Cycle over dimensions: x,y,z,x,y,z,….
  2) Put in axis-aligned hyper-planes (split at median of point set)

- Result: balanced tree

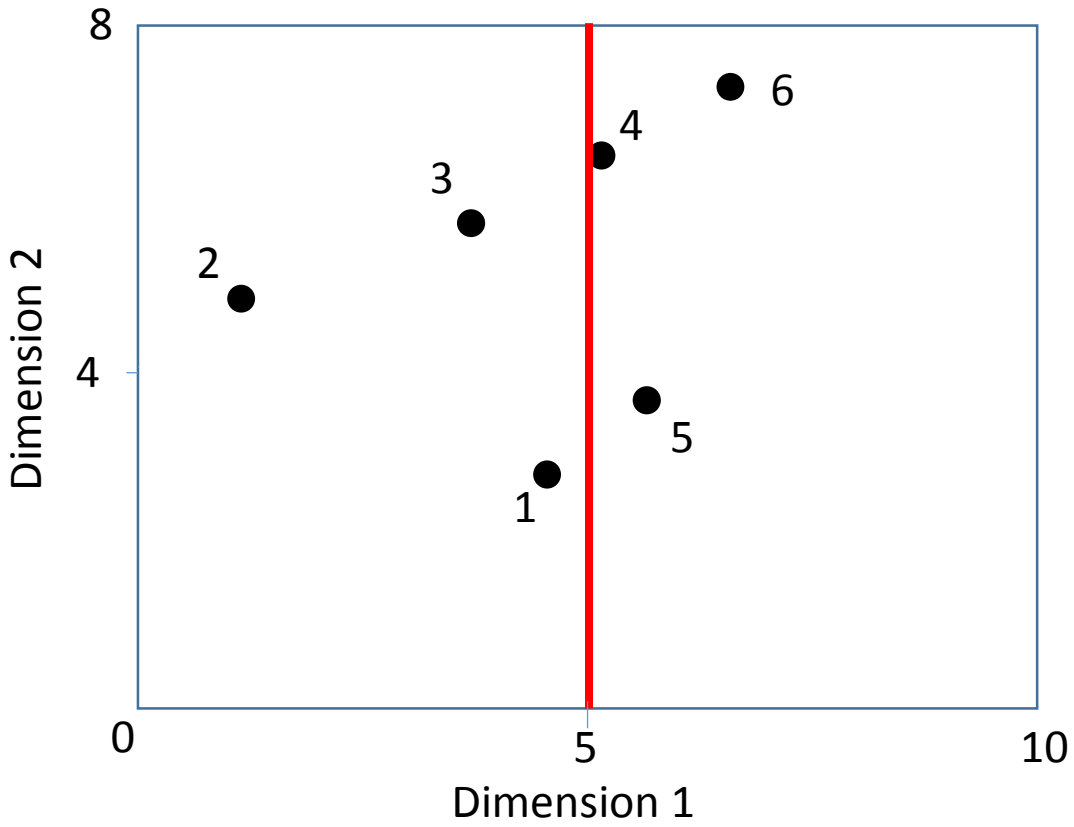
- Nearest Neighbour search (to come)

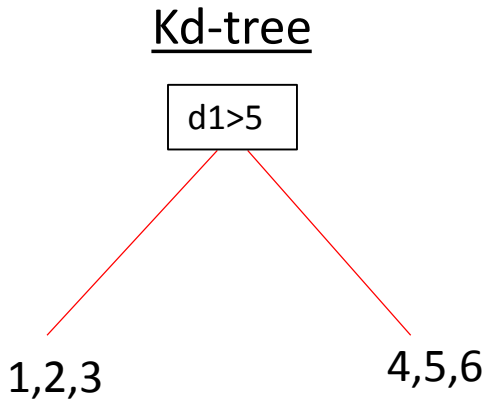
[Invented by Jon Louis Bentley 1975]

Examples

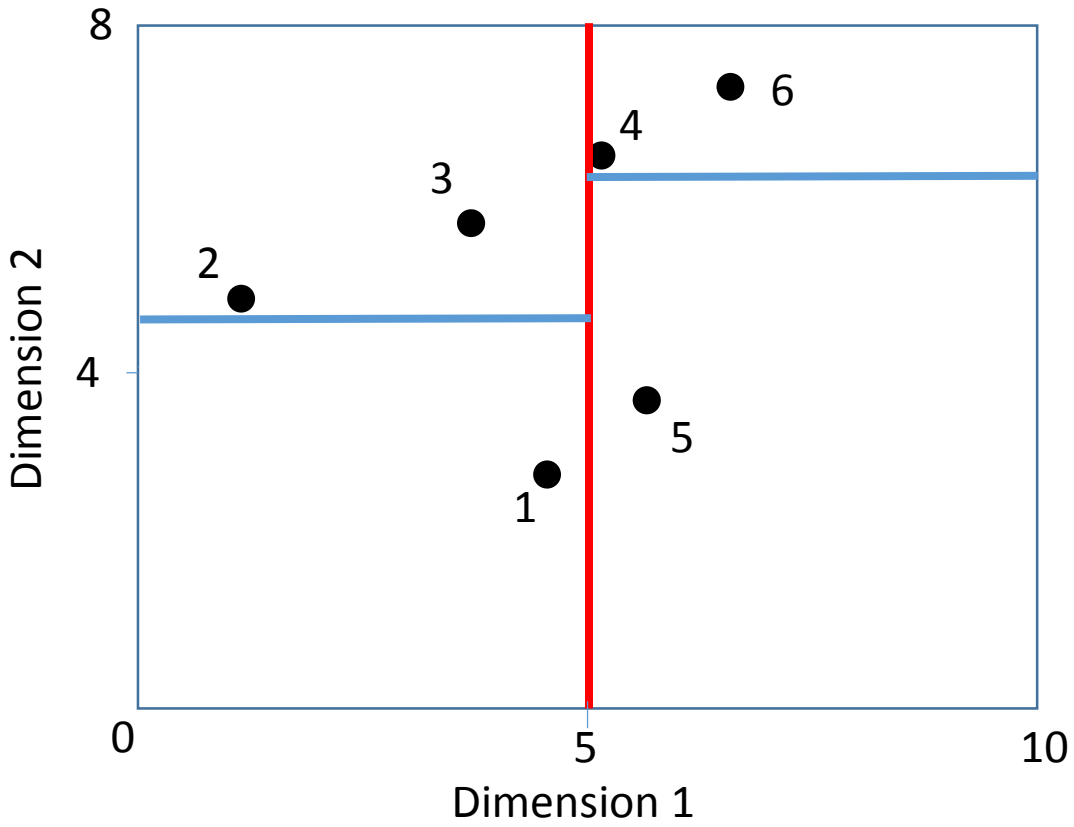


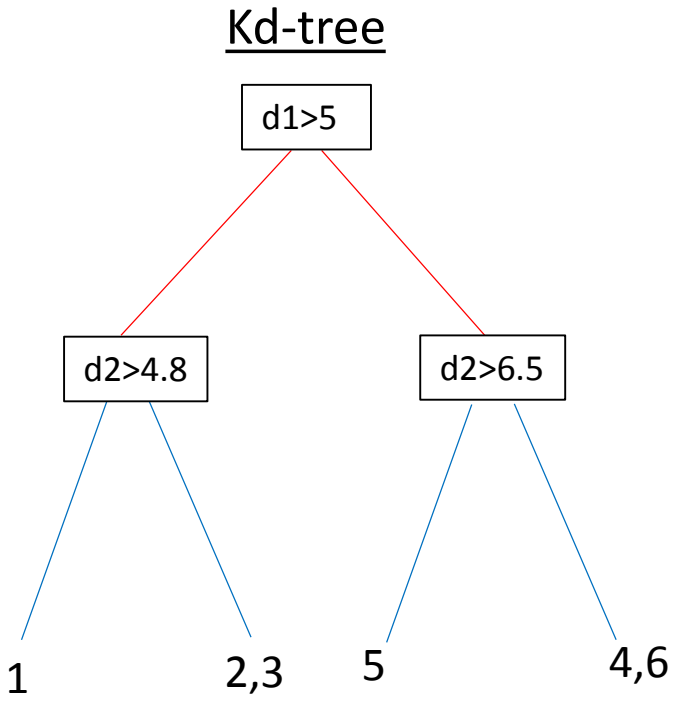
Examples



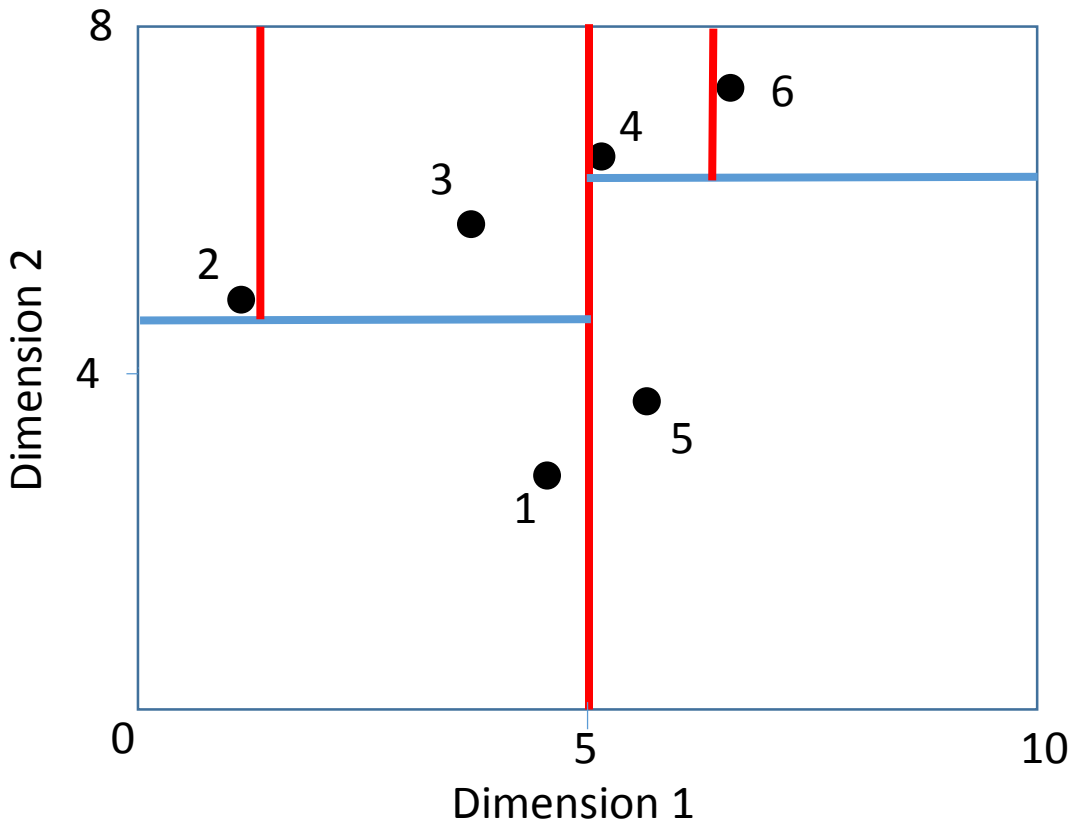


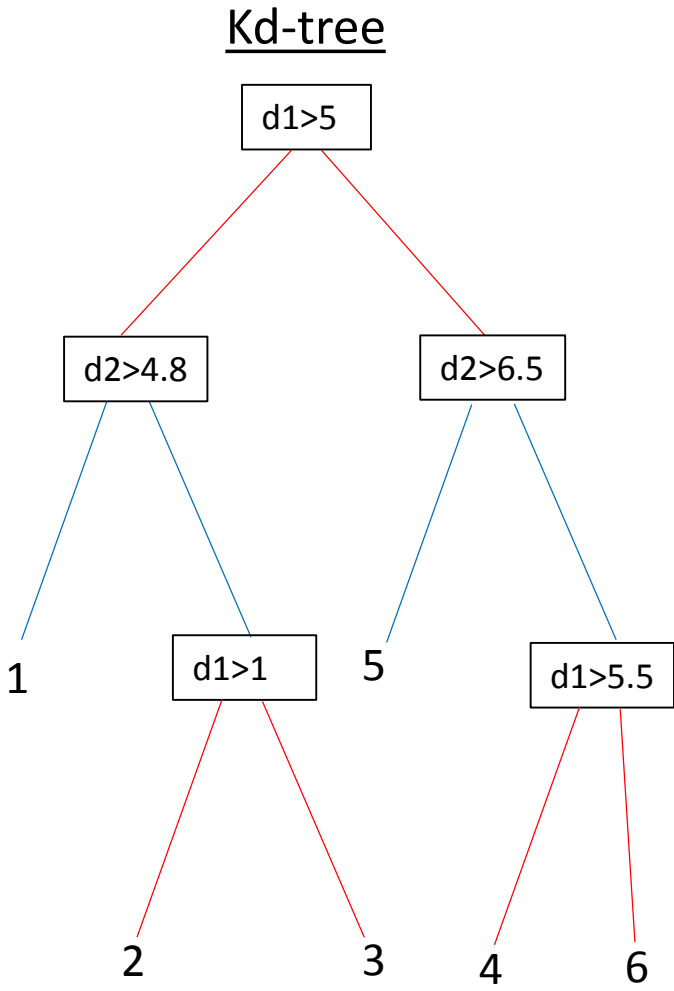
Examples


8
4
0
5
10
Dimension 2
Dimension 1
1
2
3
4
5
6

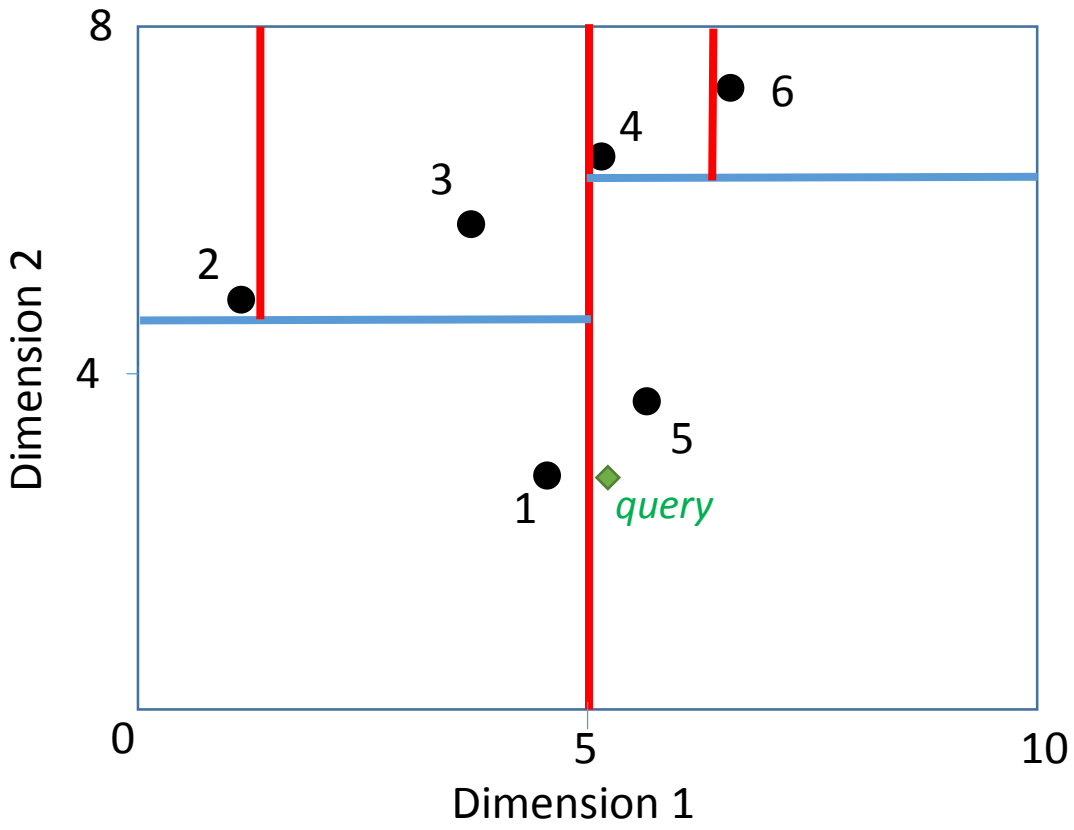

Kd-tree
d1>5
d2>4.8
d2>6.5
1
2,3
5
4,6

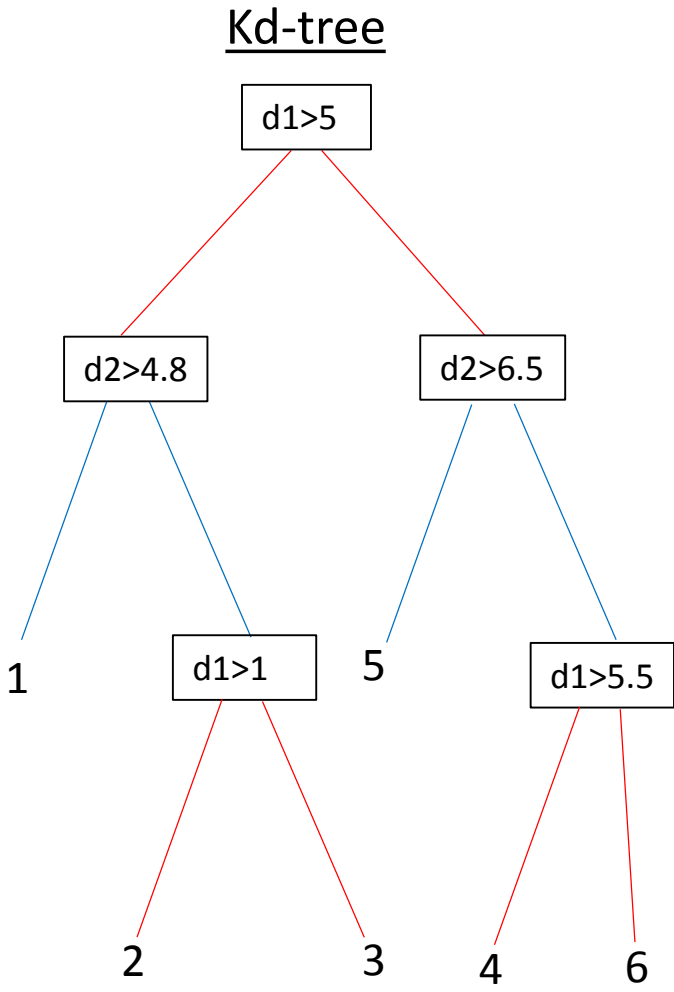
Examples


8
4
0
5
10
Dimension 2
Dimension 1
1
2
3
4
5
6

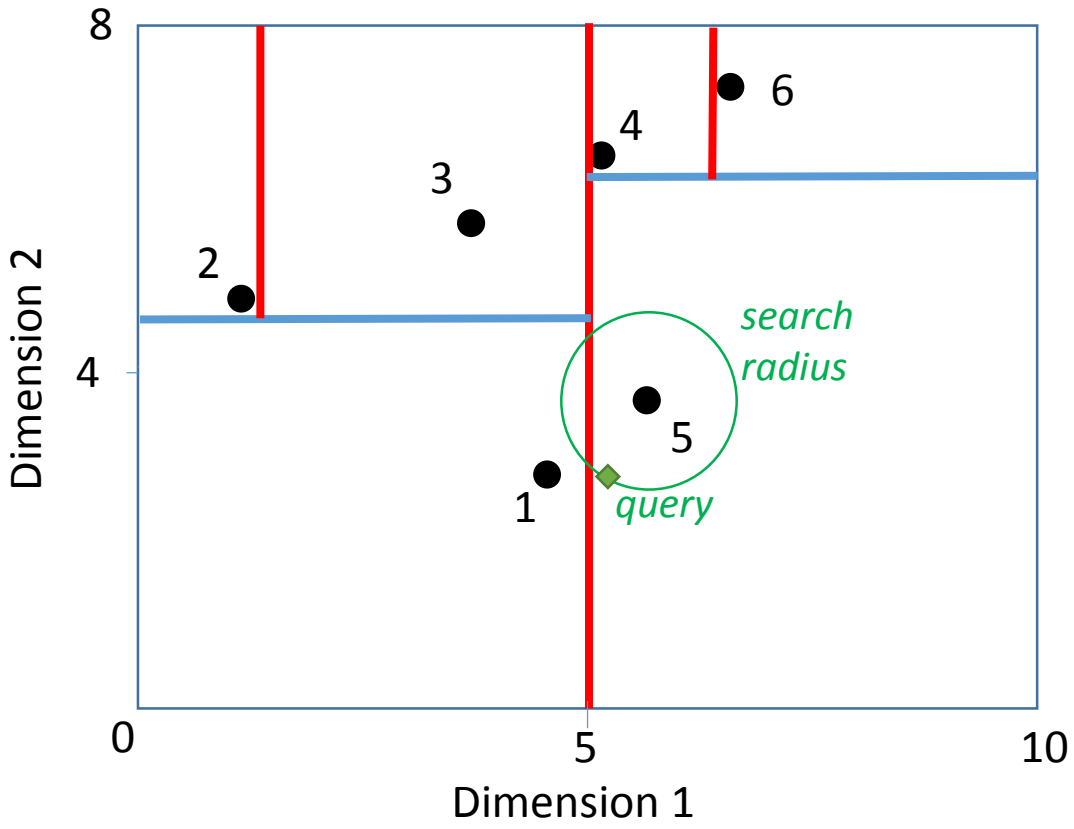

Kd-tree
d1>5
d2>4.8
d2>6.5
d1>1
d1>5.5
1
5
2
3
4
6

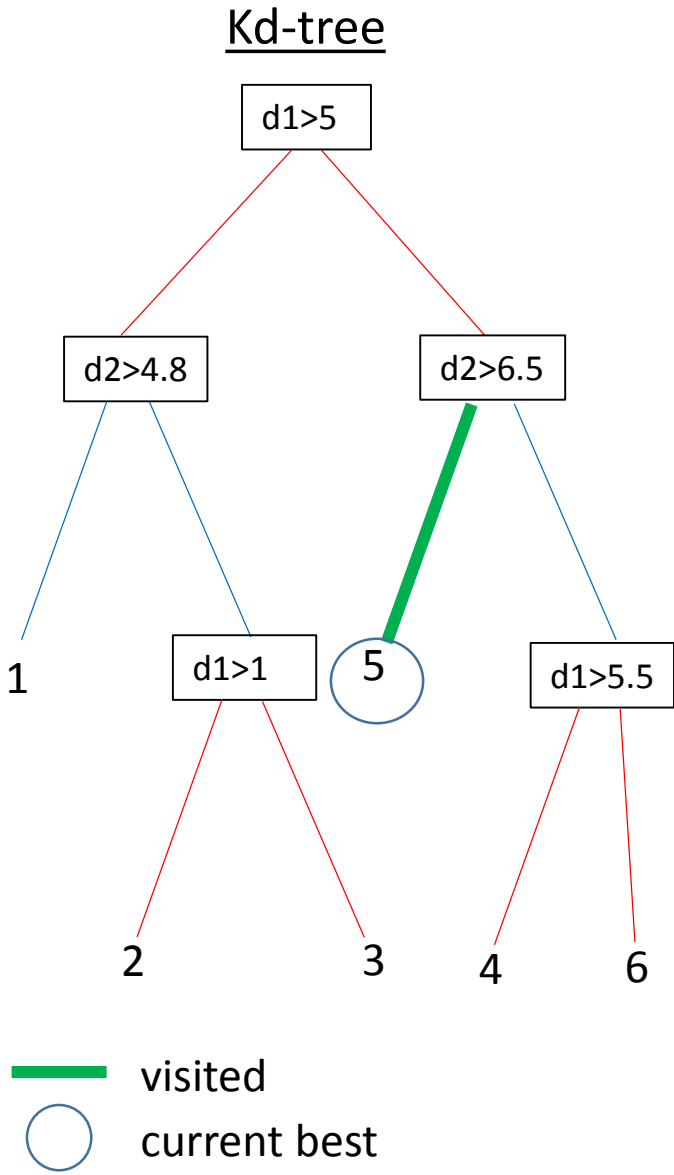
Examples: nearest neighbor search



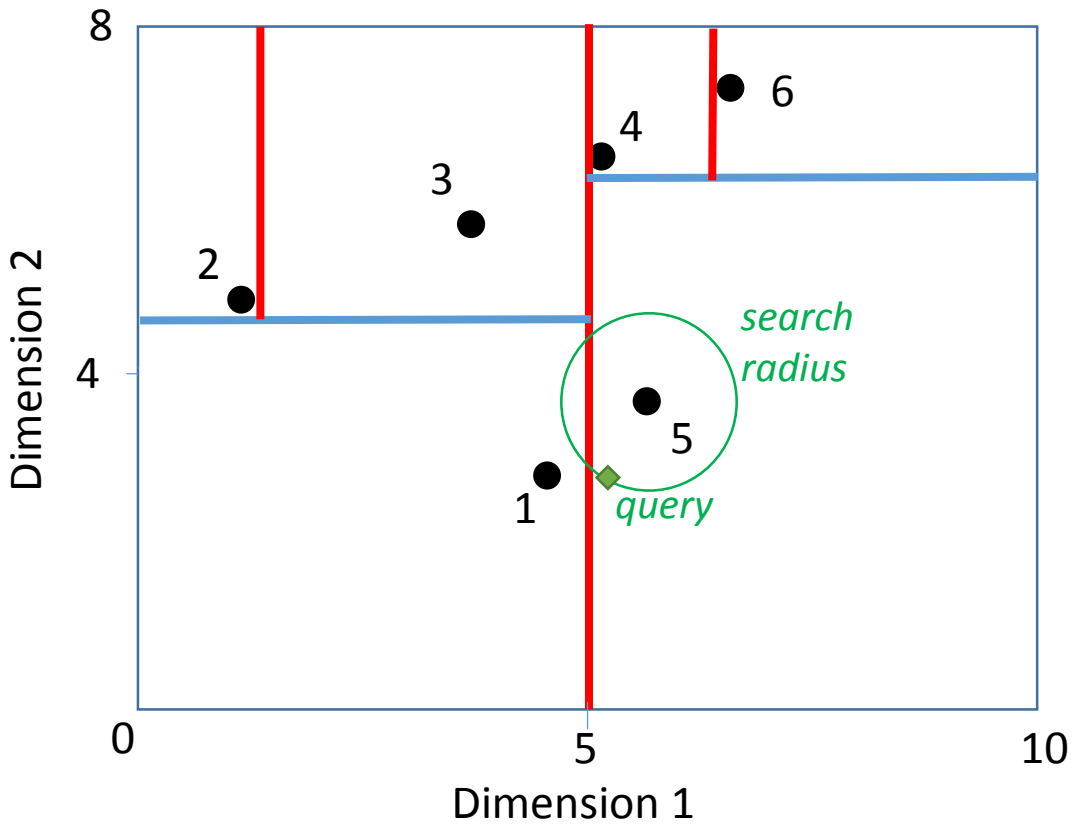


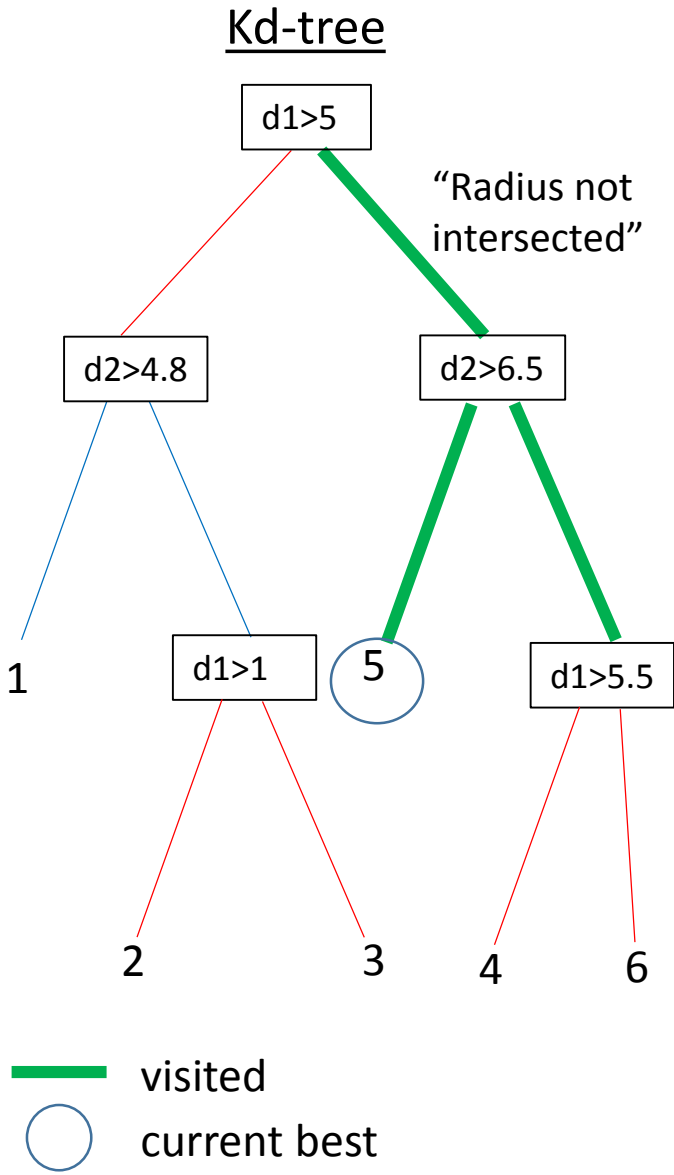
Examples: nearest neighbor search



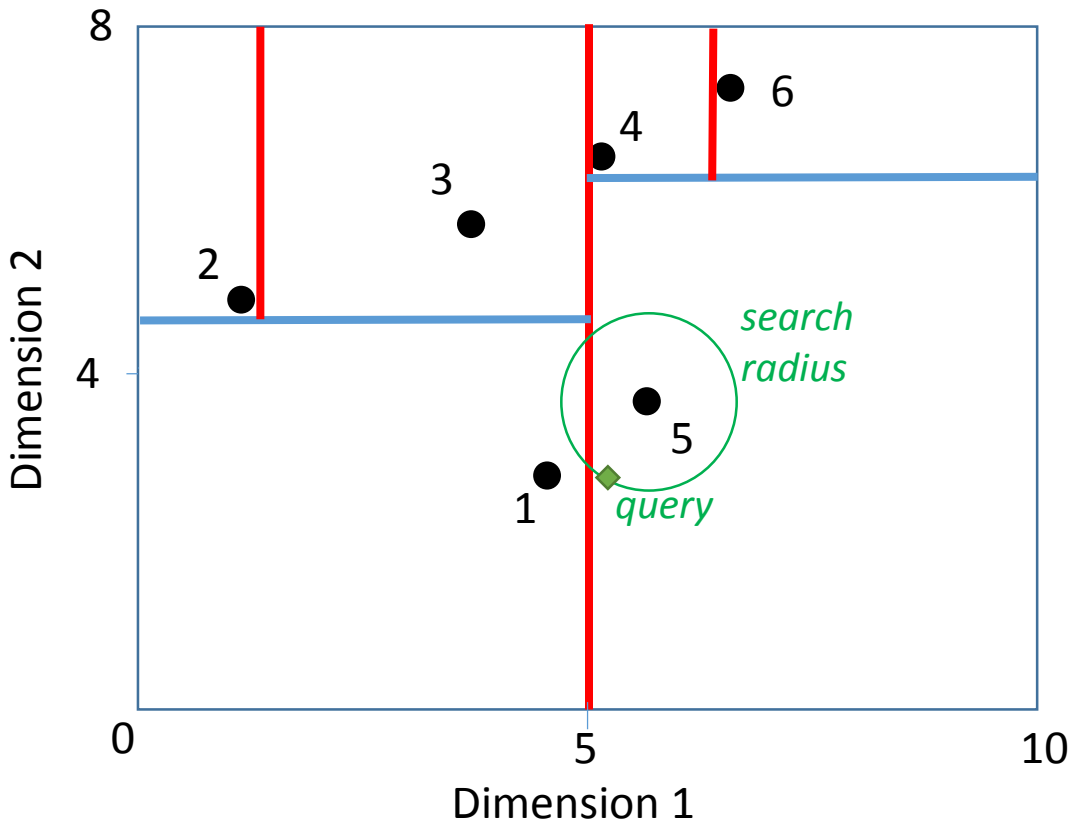


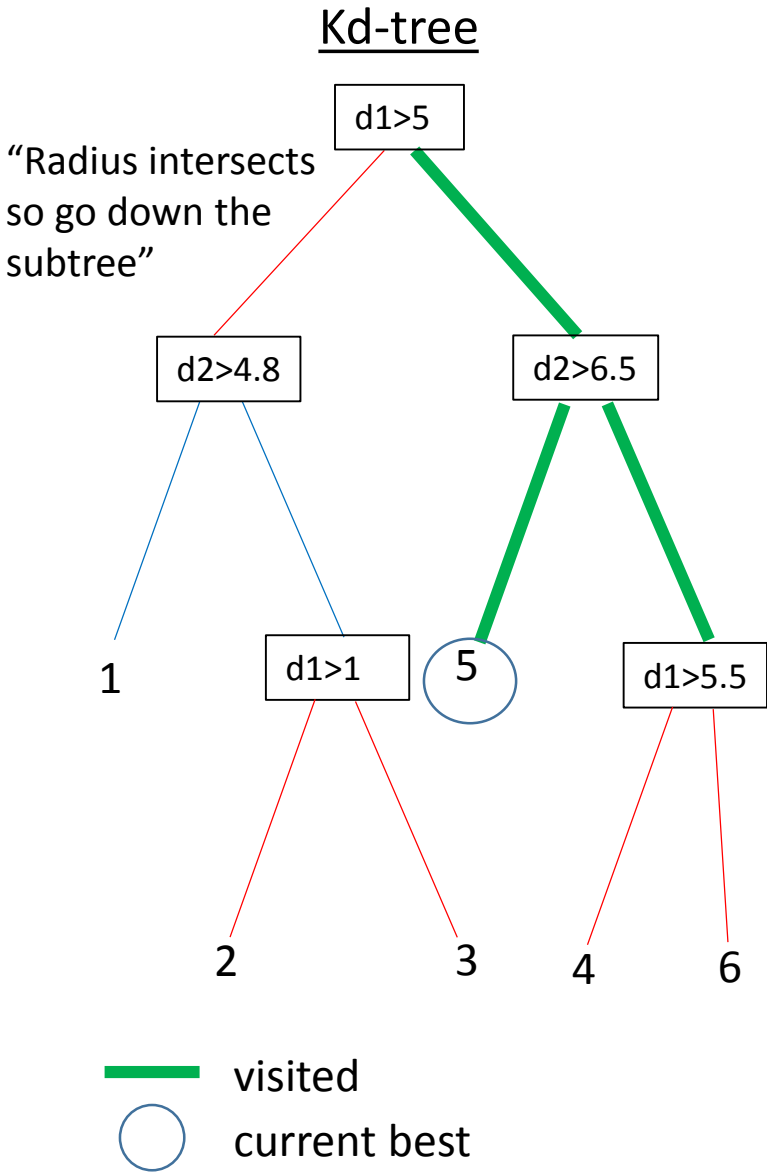
Examples: nearest neighbor search



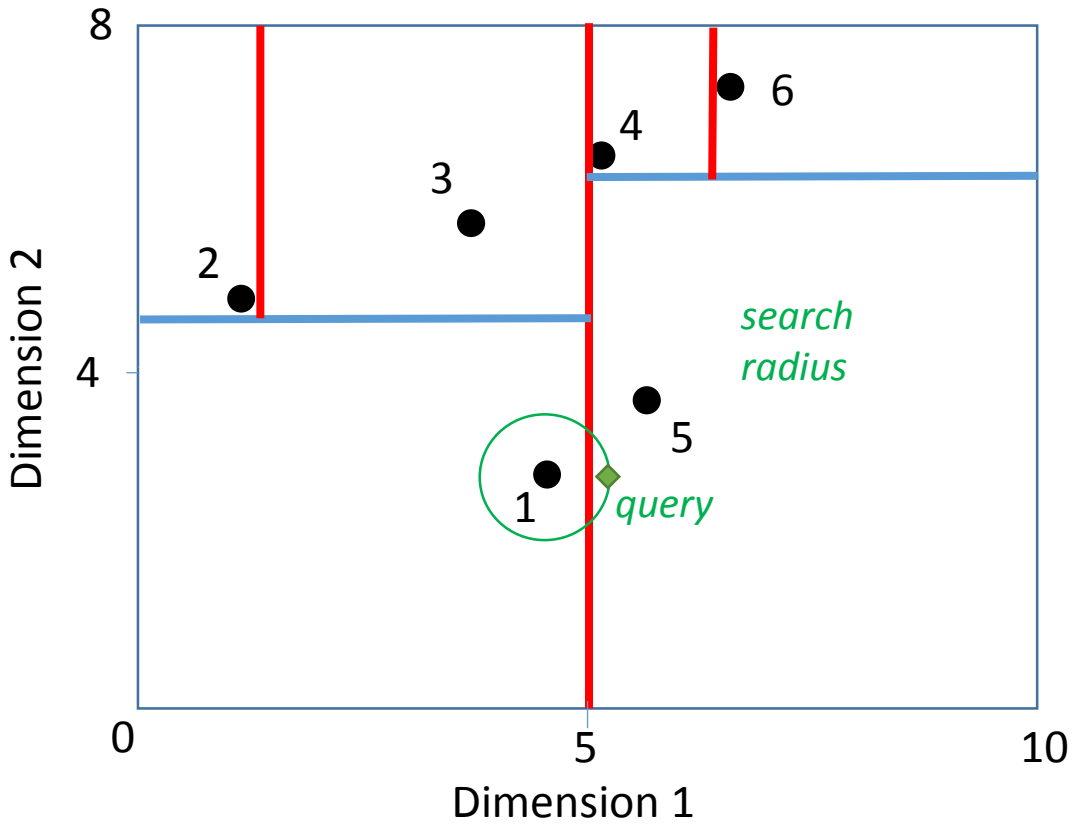


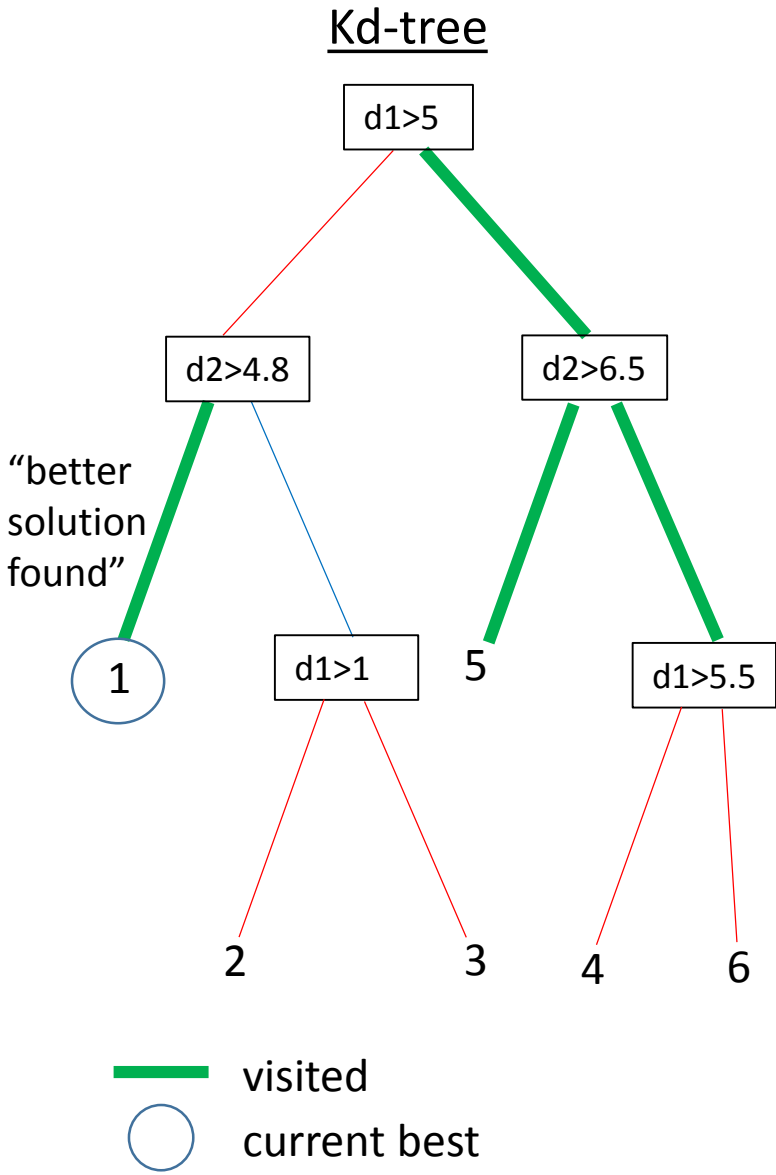
Examples: nearest neighbor search



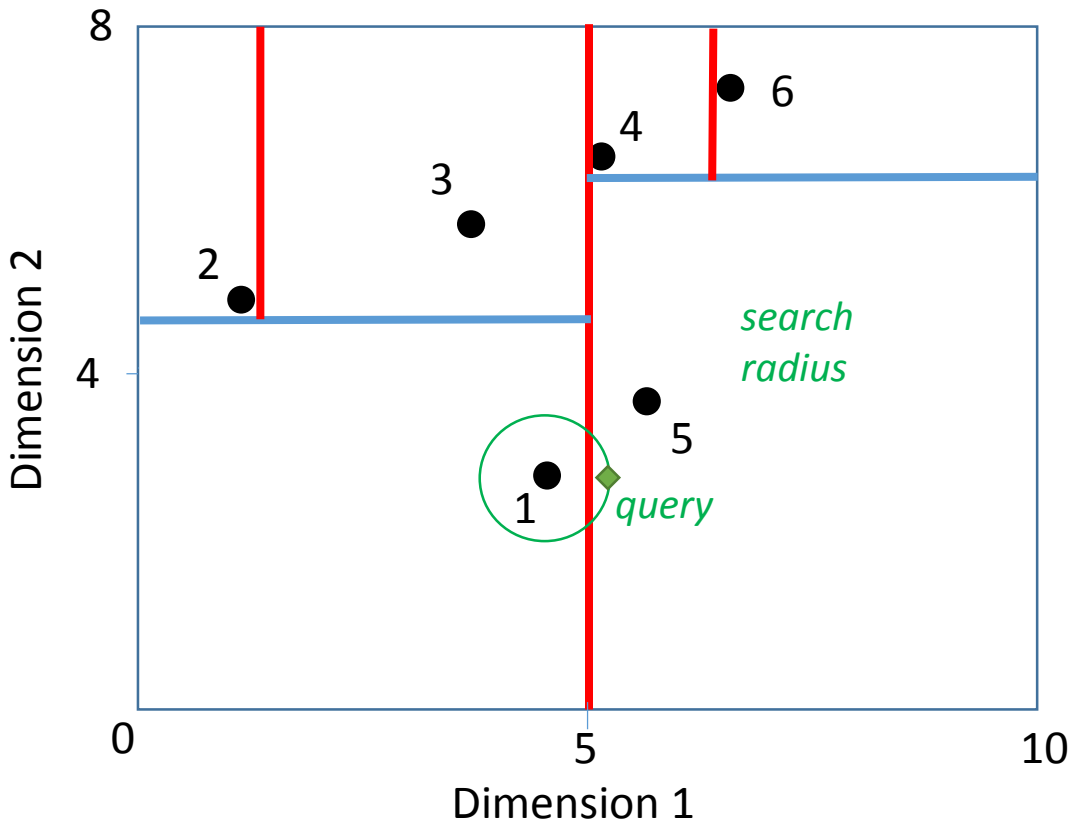


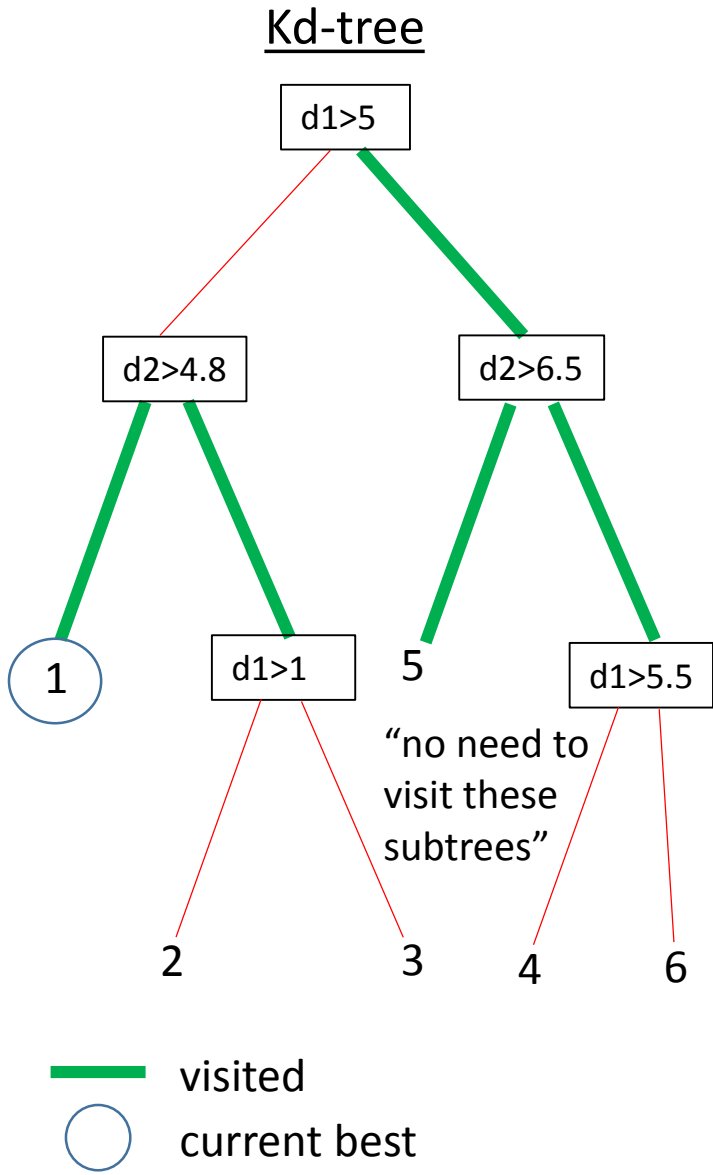
Examples: nearest neighbor search



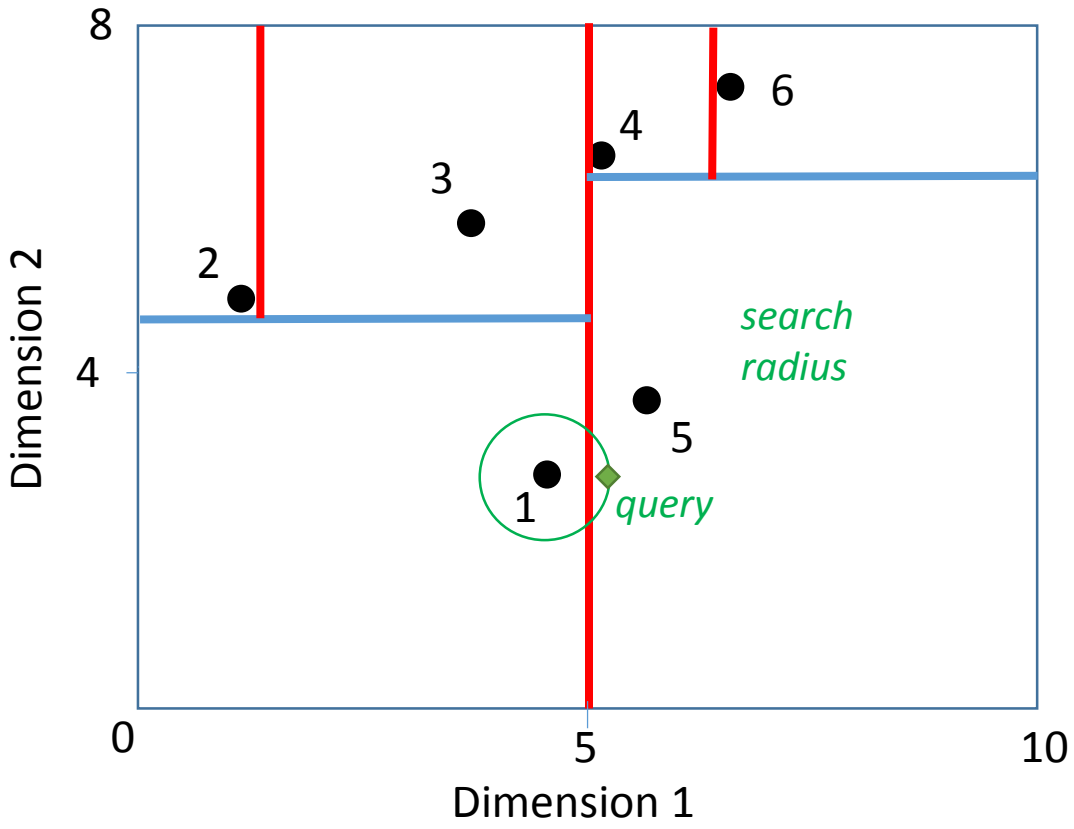


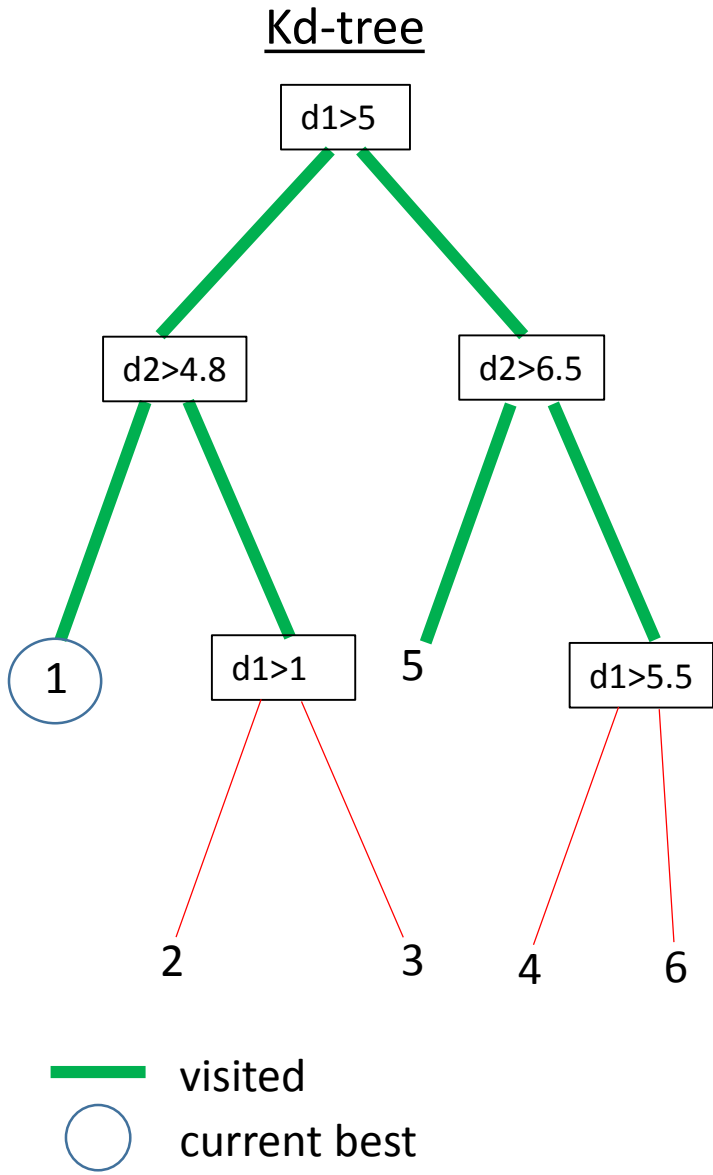
Examples: nearest neighbor search





Examples: nearest neighbor search





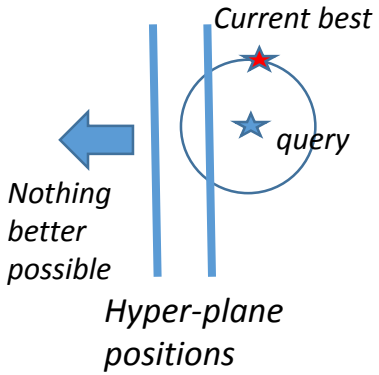
Done since root-node is marked in both ways

Nearest neighbour search – pseudo code

Input: query point

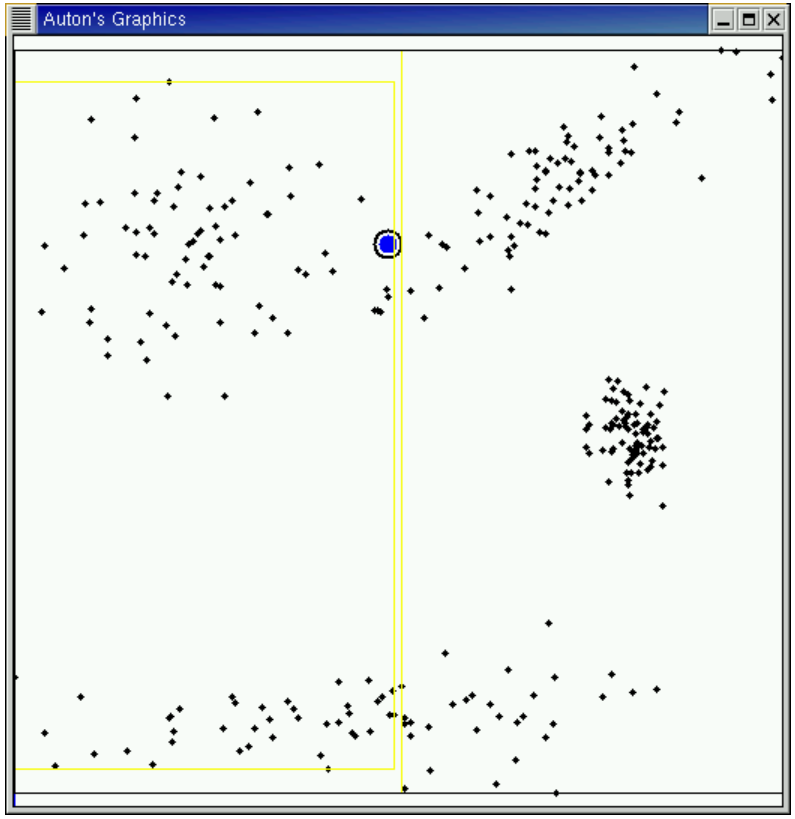
Pseudo code

Step 1: Find leave node (bucket) with query point
Step 2: Make hyper-sphere with radius
(current best and query point)
Step3: go up the tree and see if hyper-plane intersects hyper-sphere
Step 3a: ***no intersection:*** mark tree branch as visited
since no better point can be found there.
If node is root node then stop.
Step 3b: ***intersection:*** go down the branch to find
potentially a better point. If so, mark as
current best and go to Step 2.



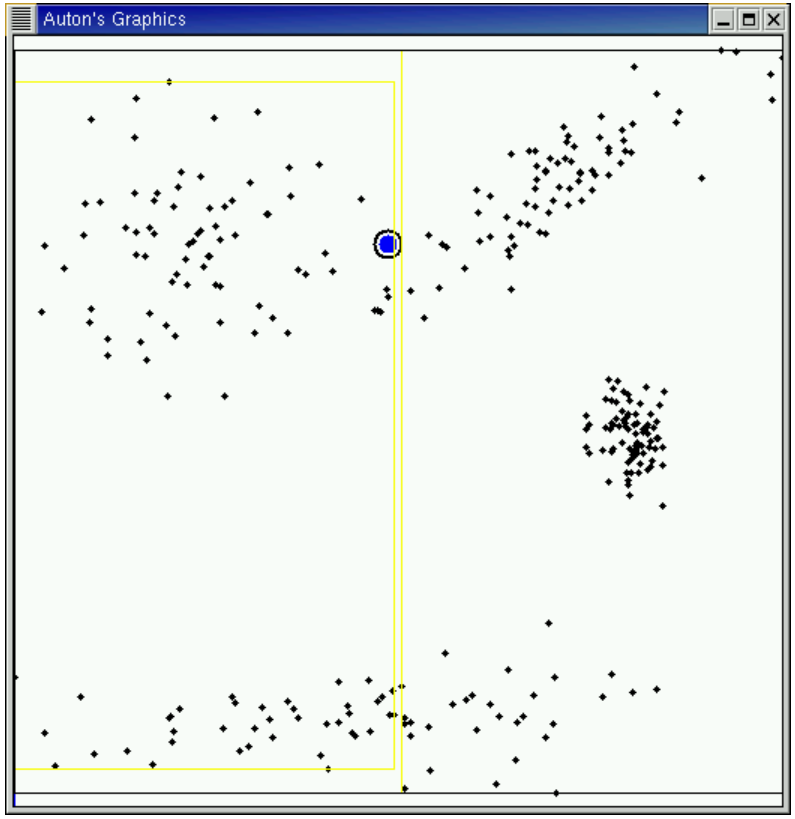
On average O(log N)

Example with many points in 2D

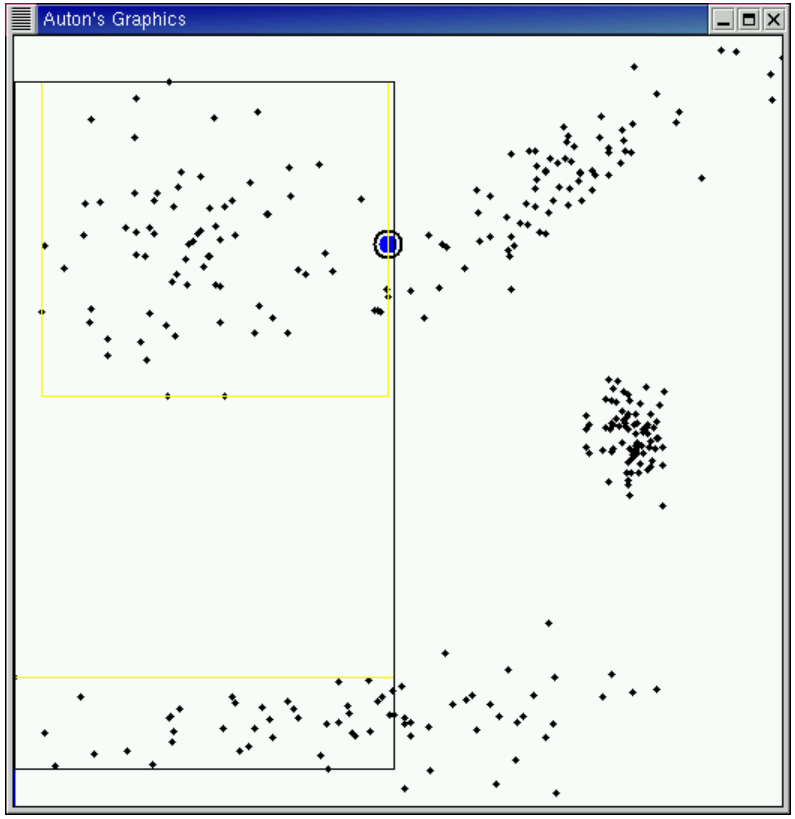


From Andrew Moore: http://www.cs.cmu.edu/~awm/animations/kdtree/

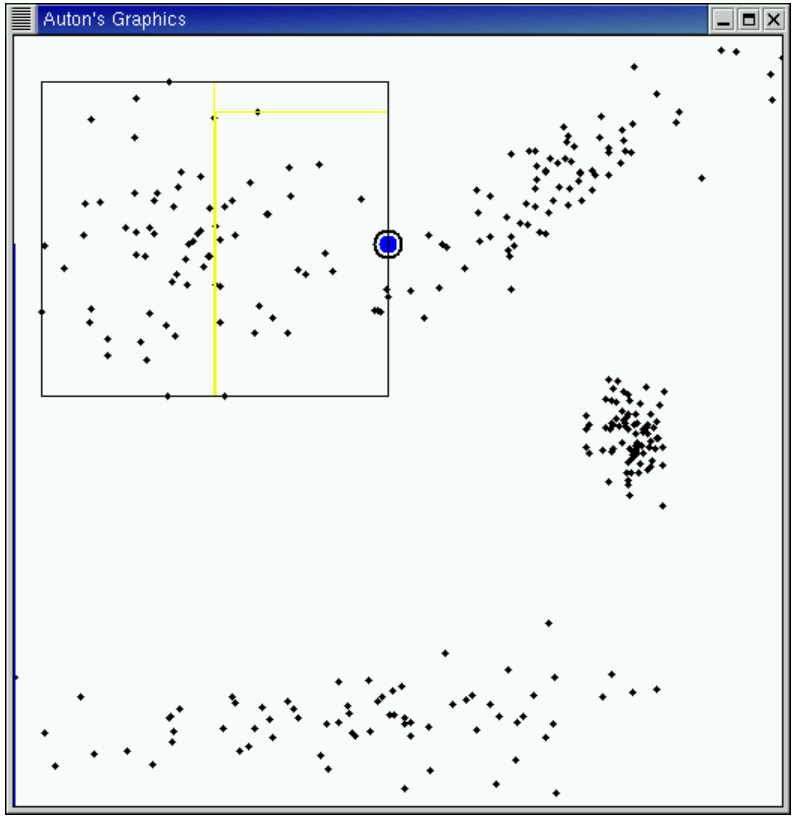
Example with many points in 2D



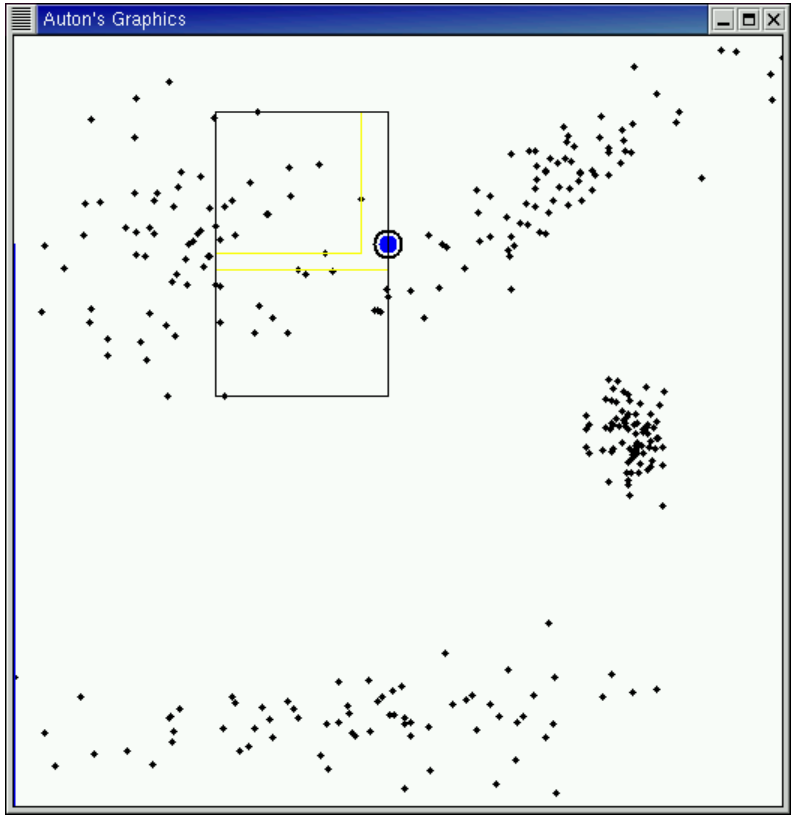
Example with many points in 2D



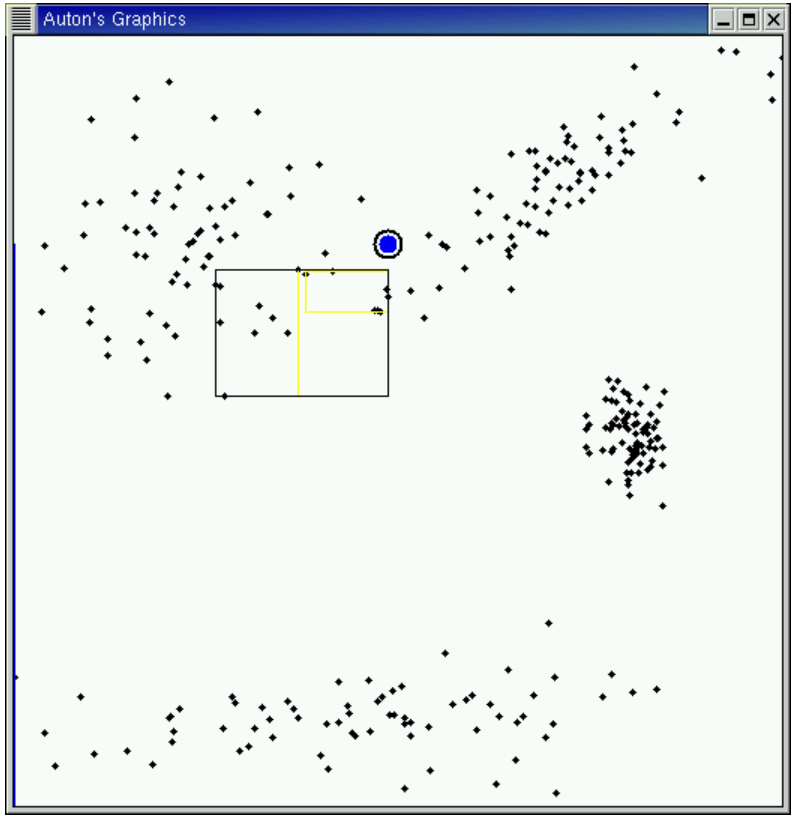
Example with many points in 2D



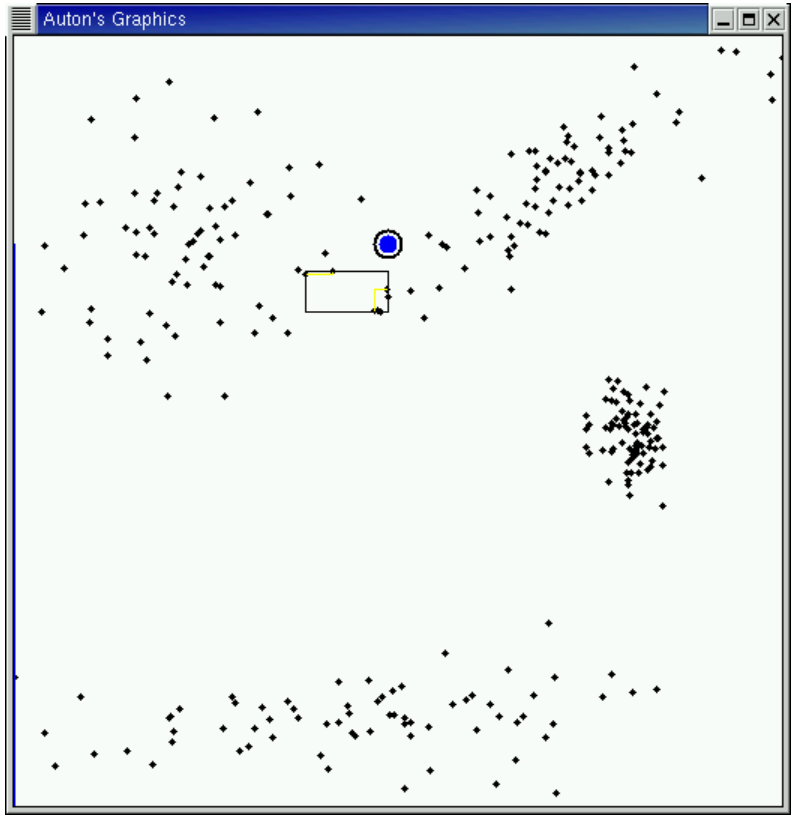
Example with many points in 2D



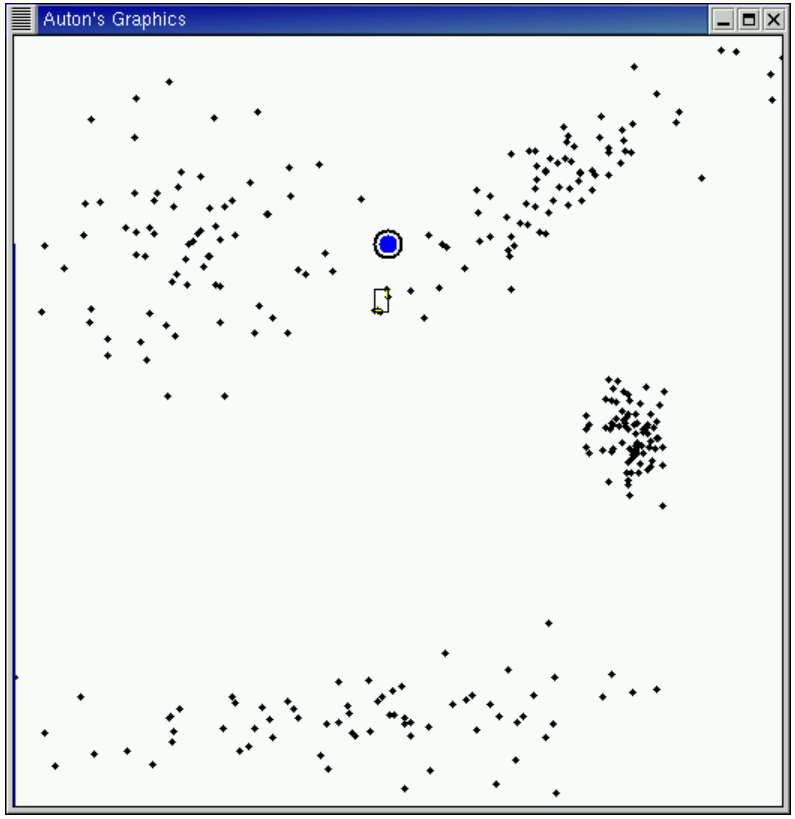
Example with many points in 2D



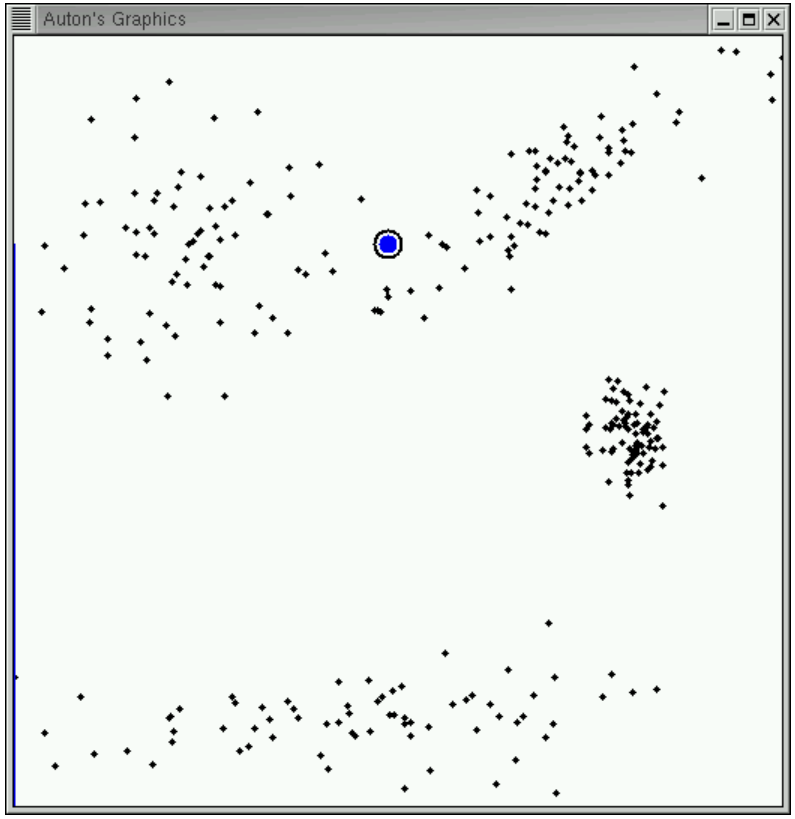
Example with many points in 2D



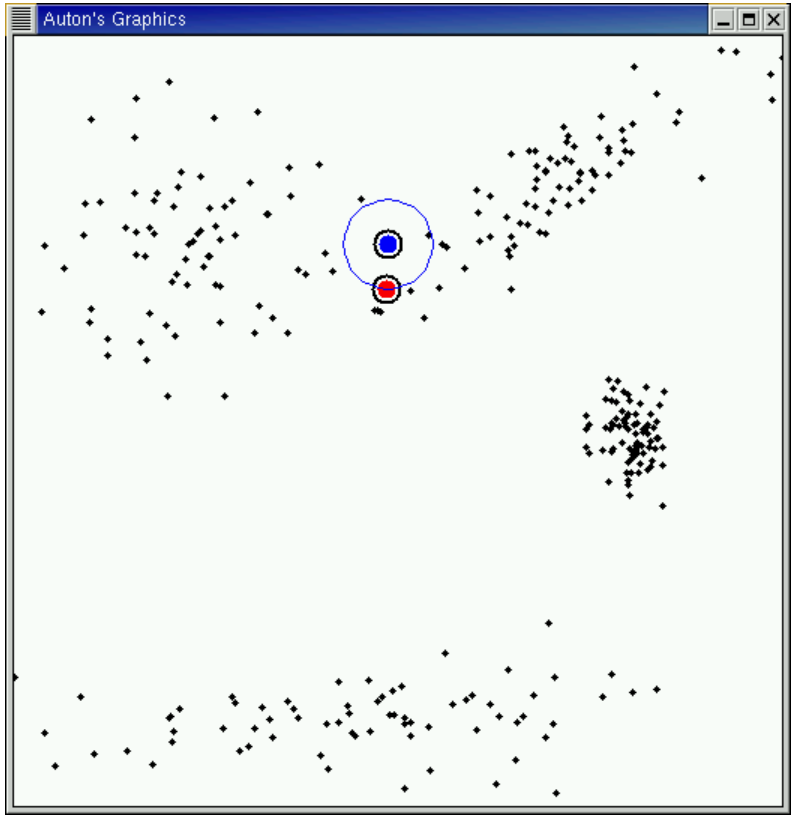
Example with many points in 2D



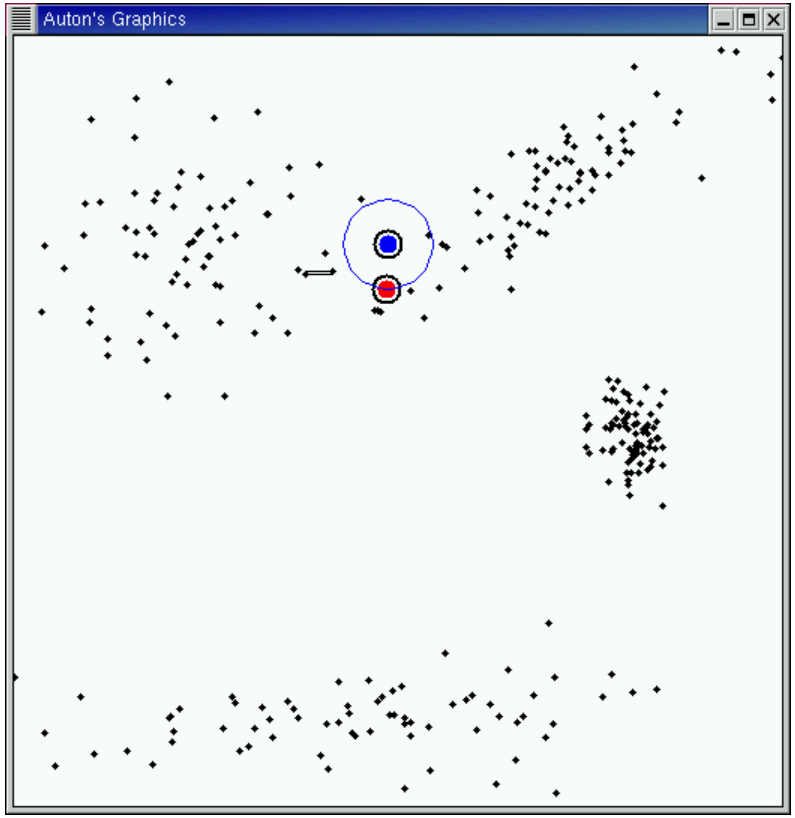
Example with many points in 2D



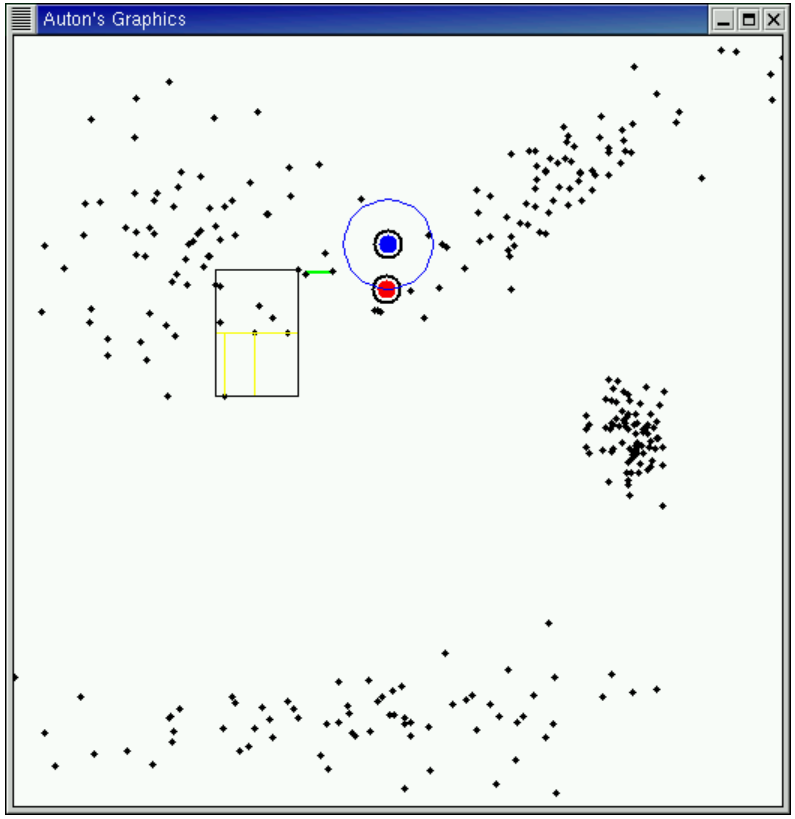
Example with many points in 2D



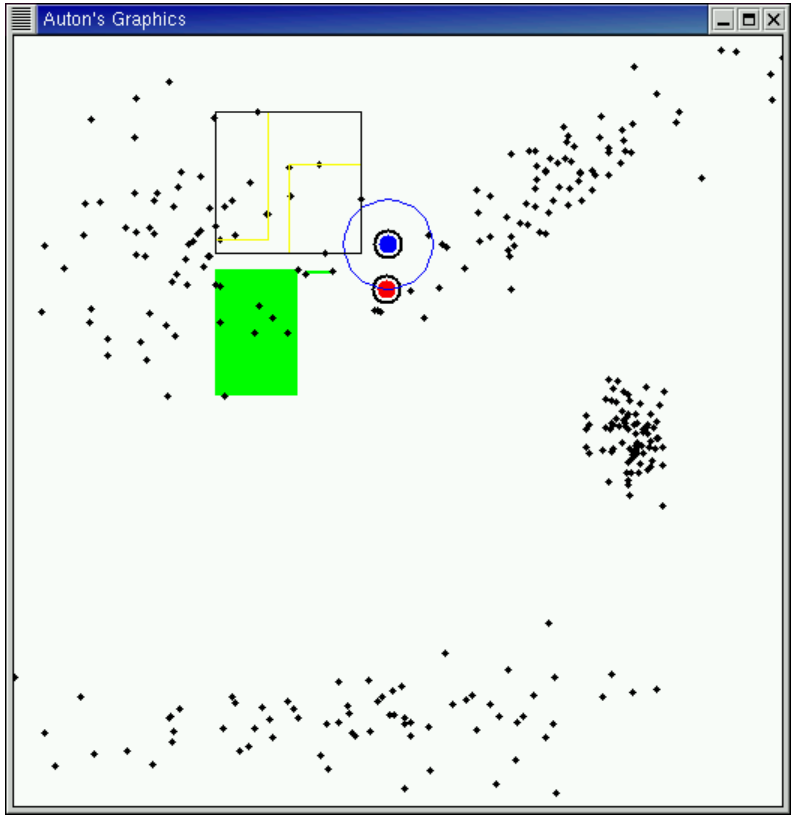
Example with many points in 2D



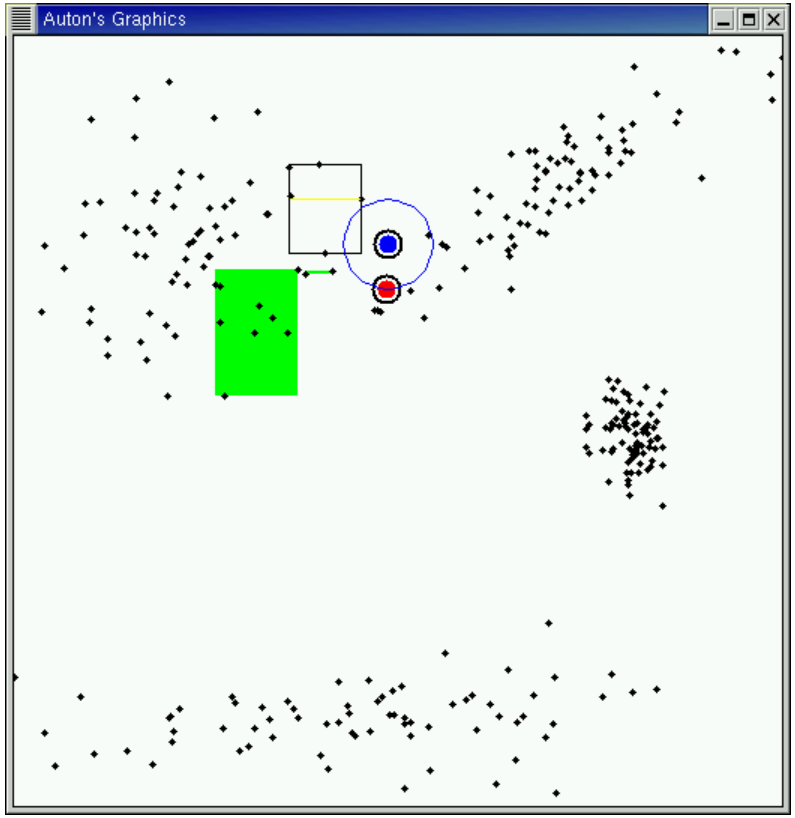
Example with many points in 2D



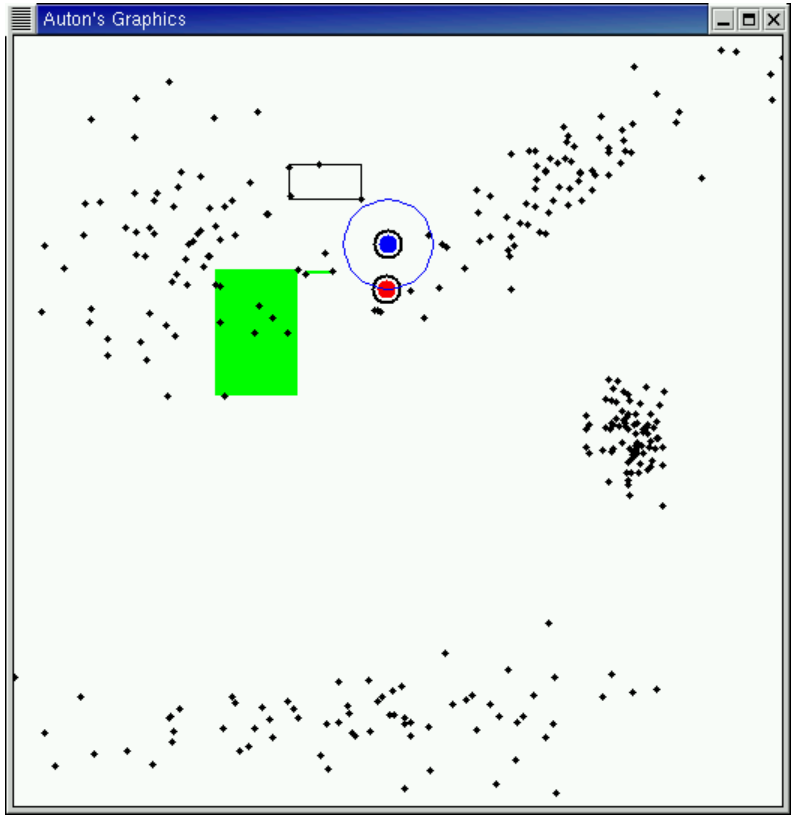
Example with many points in 2D



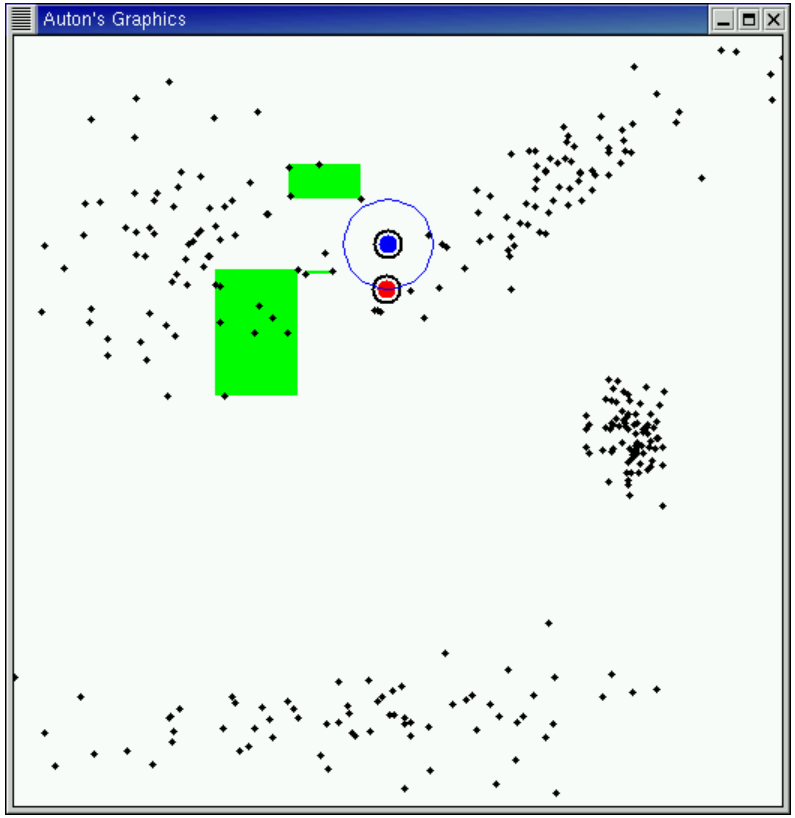
Example with many points in 2D



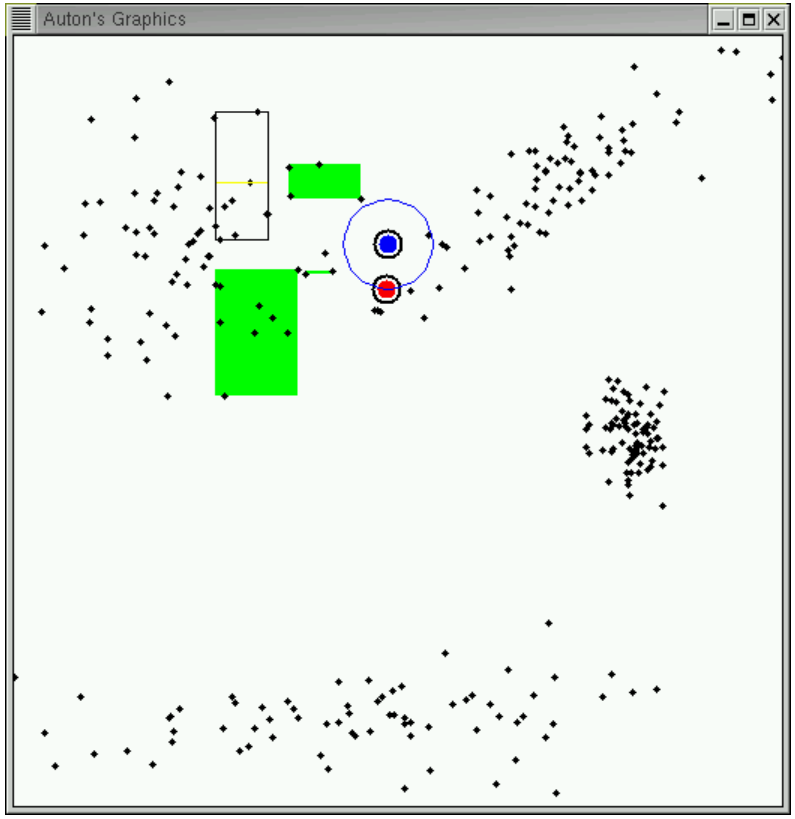
Example with many points in 2D



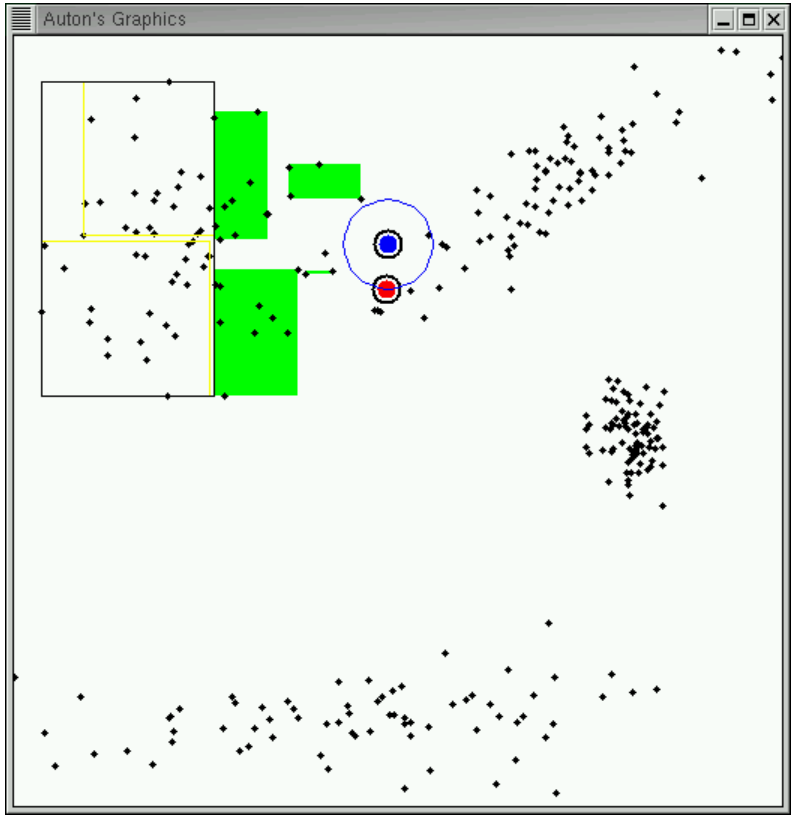
Example with many points in 2D



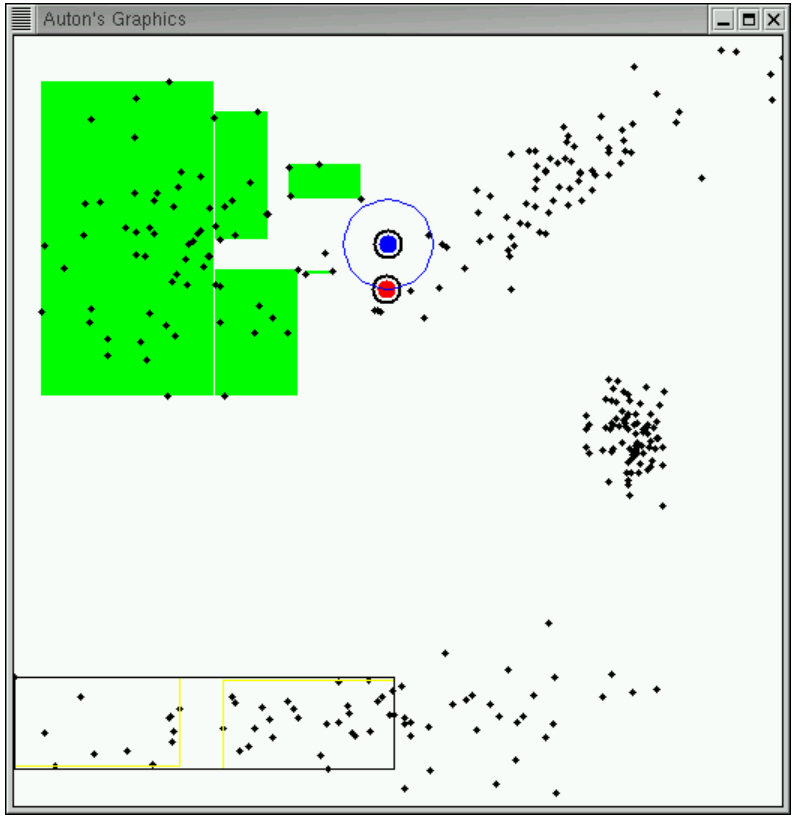
Example with many points in 2D



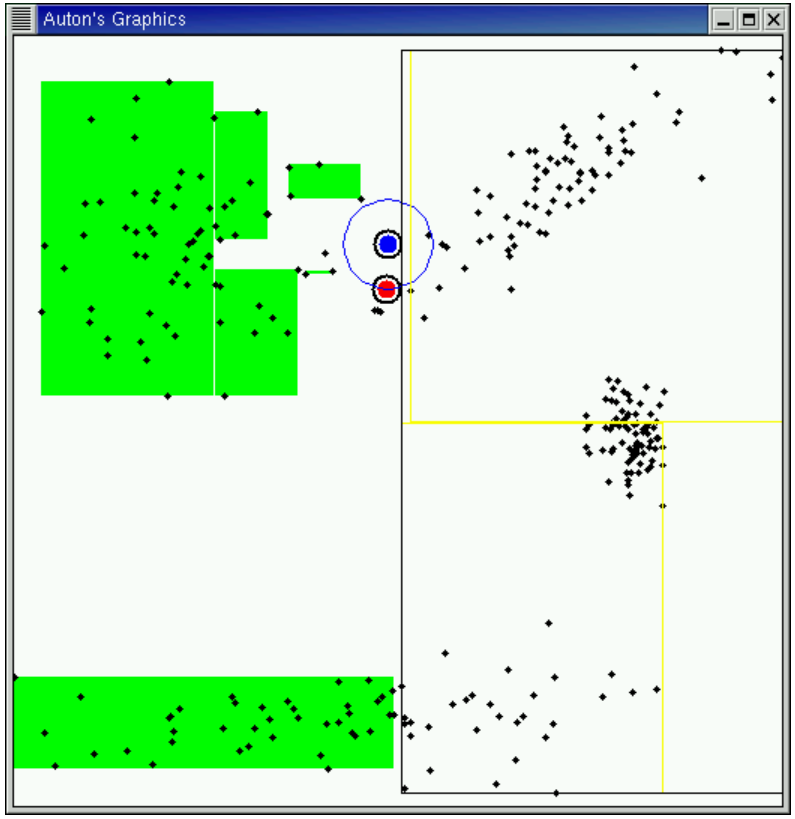
Example with many points in 2D



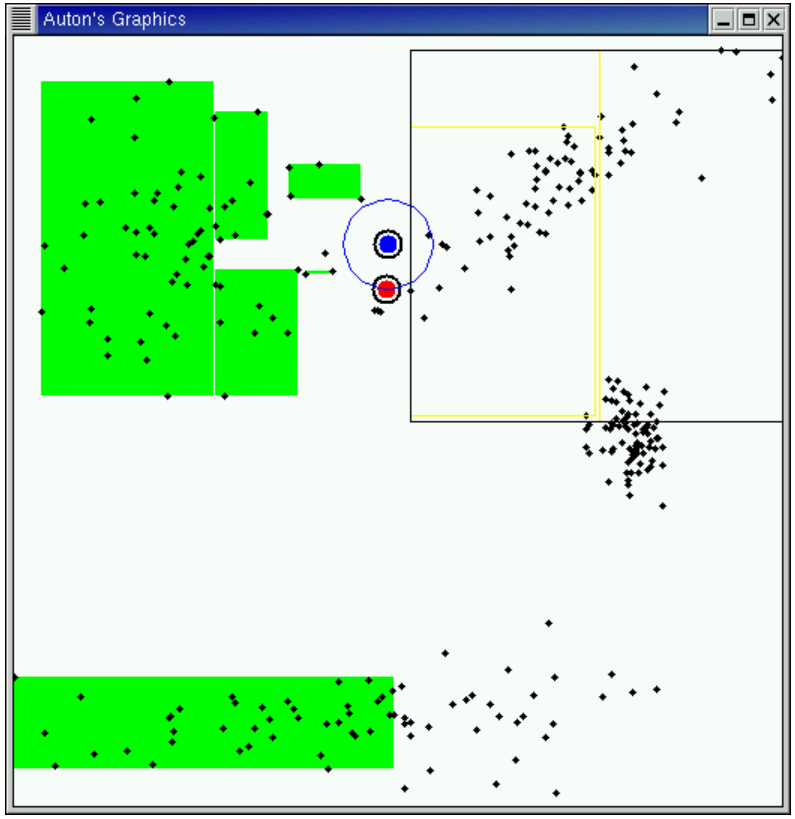
Example with many points in 2D



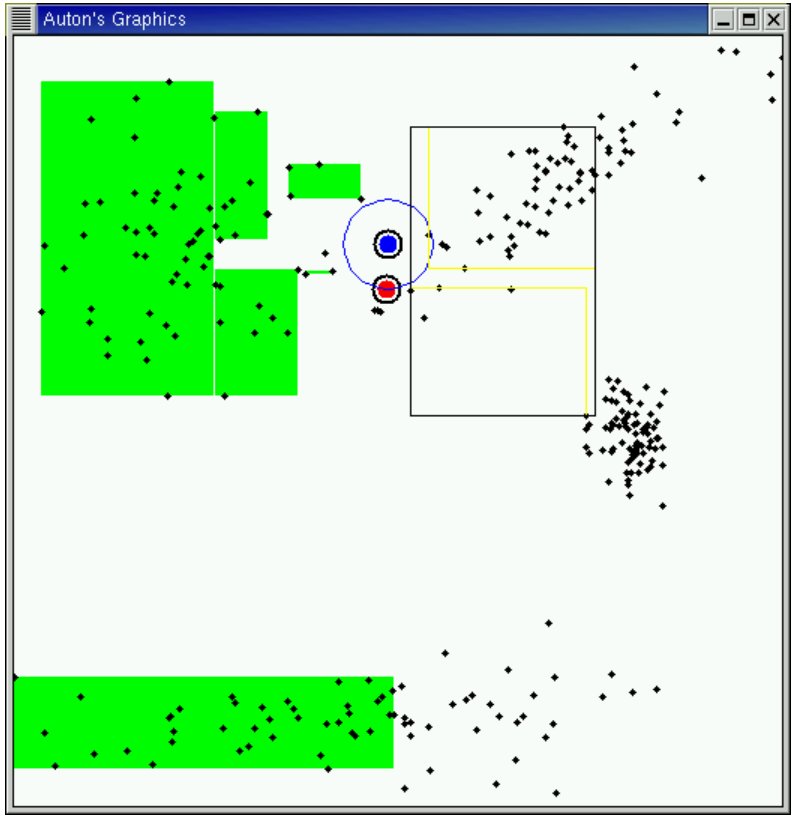
Example with many points in 2D



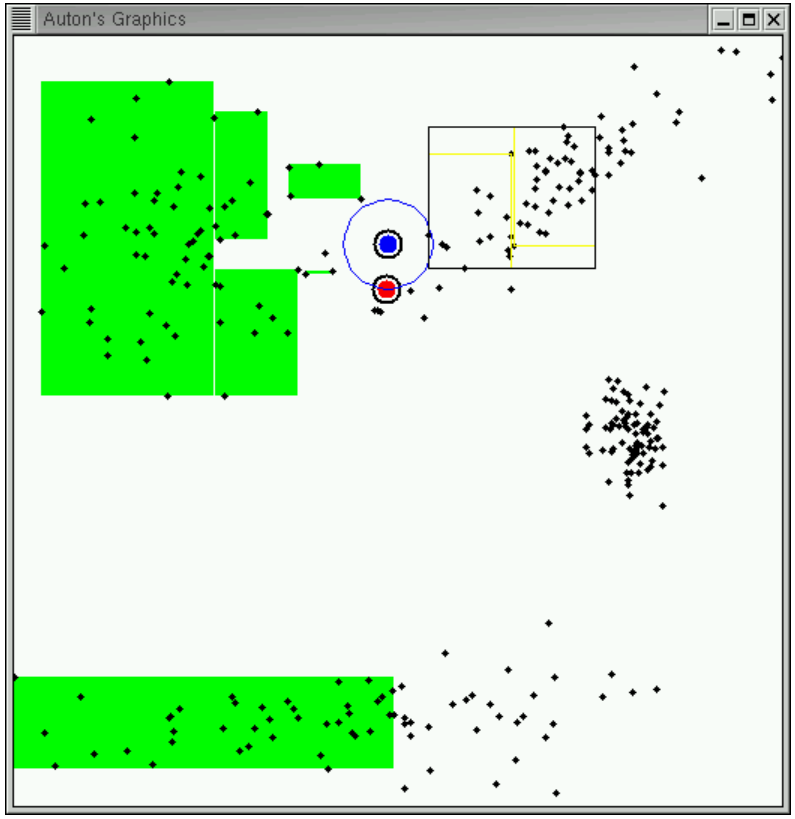
Example with many points in 2D



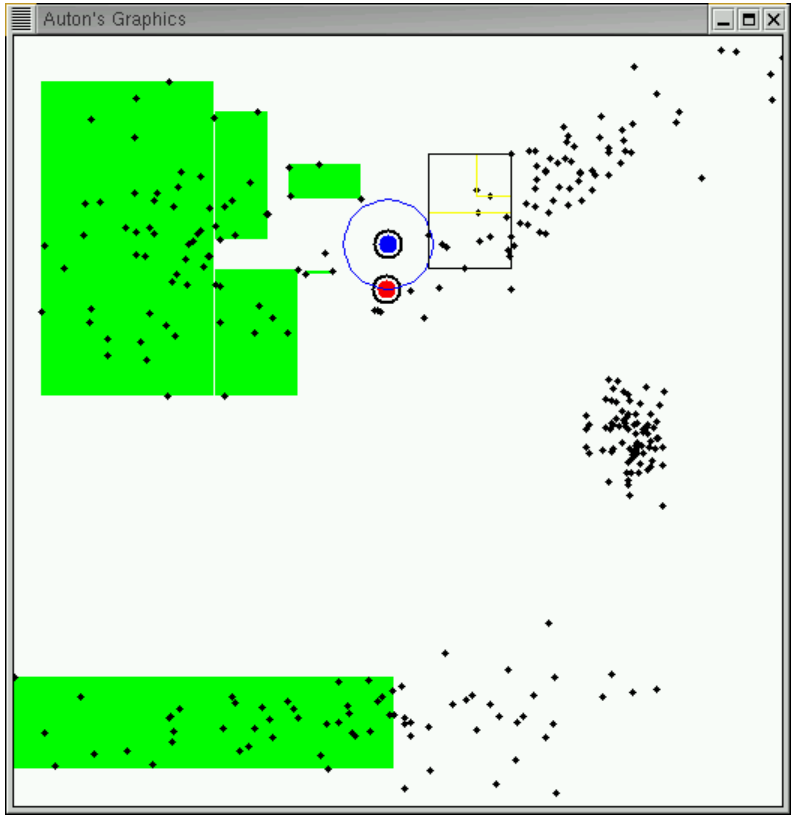
Example with many points in 2D



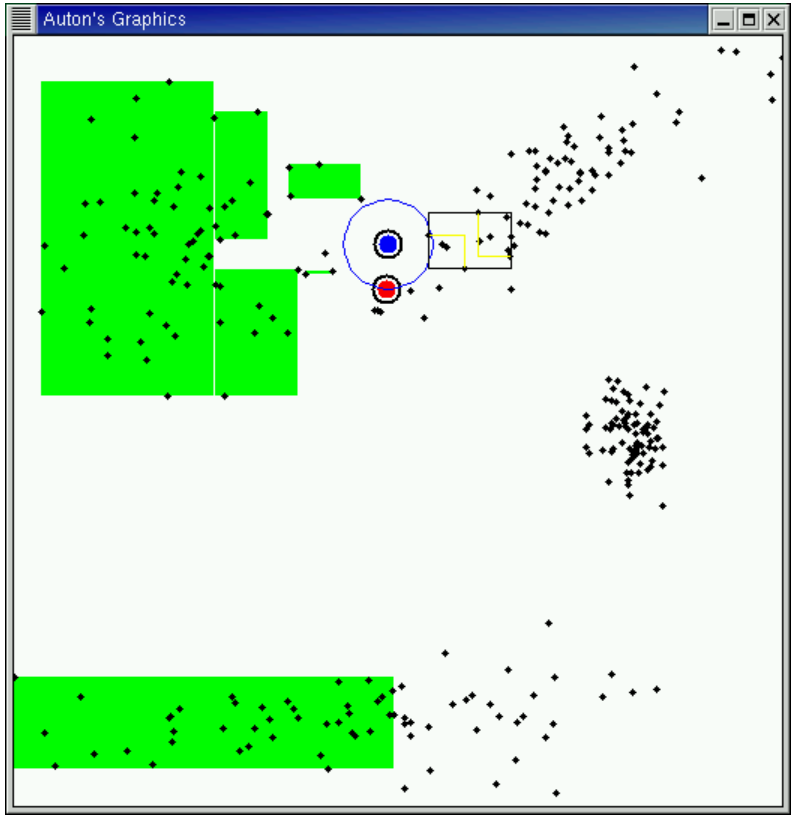
Example with many points in 2D



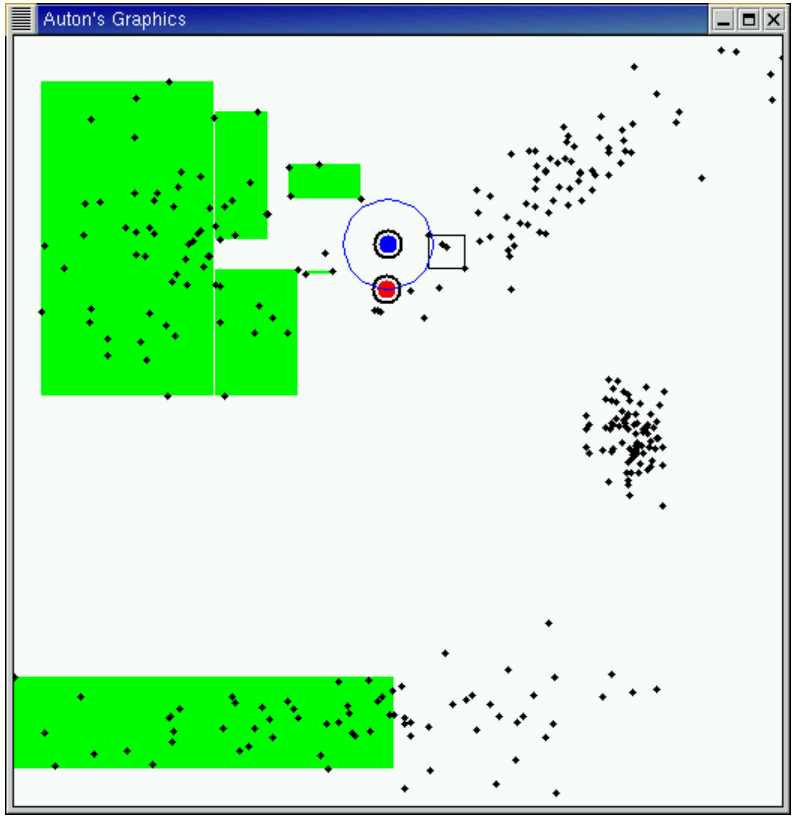
Example with many points in 2D



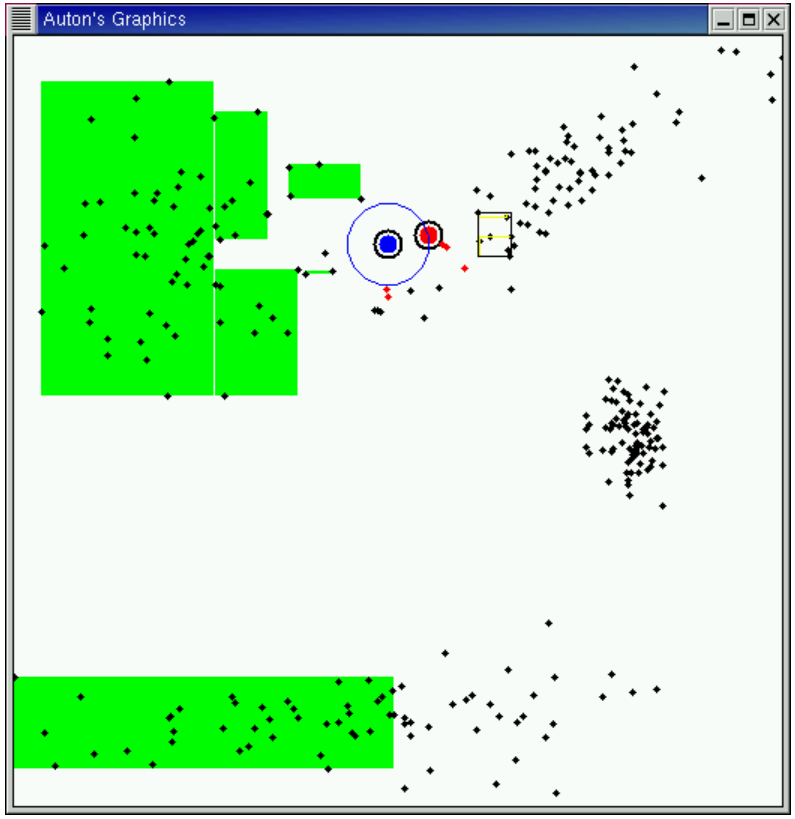
Example with many points in 2D



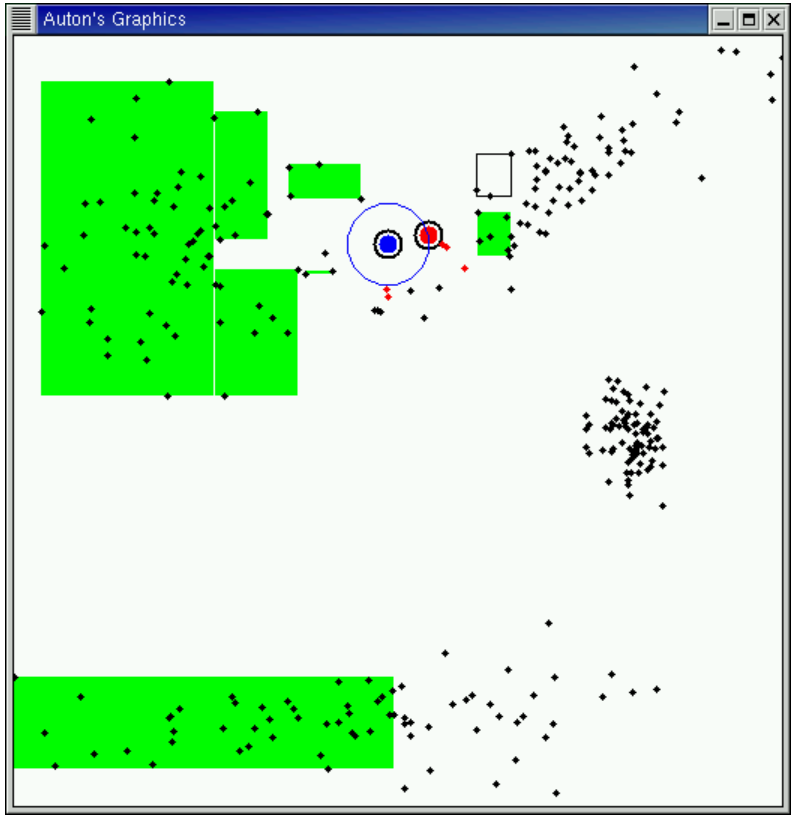
Example with many points in 2D



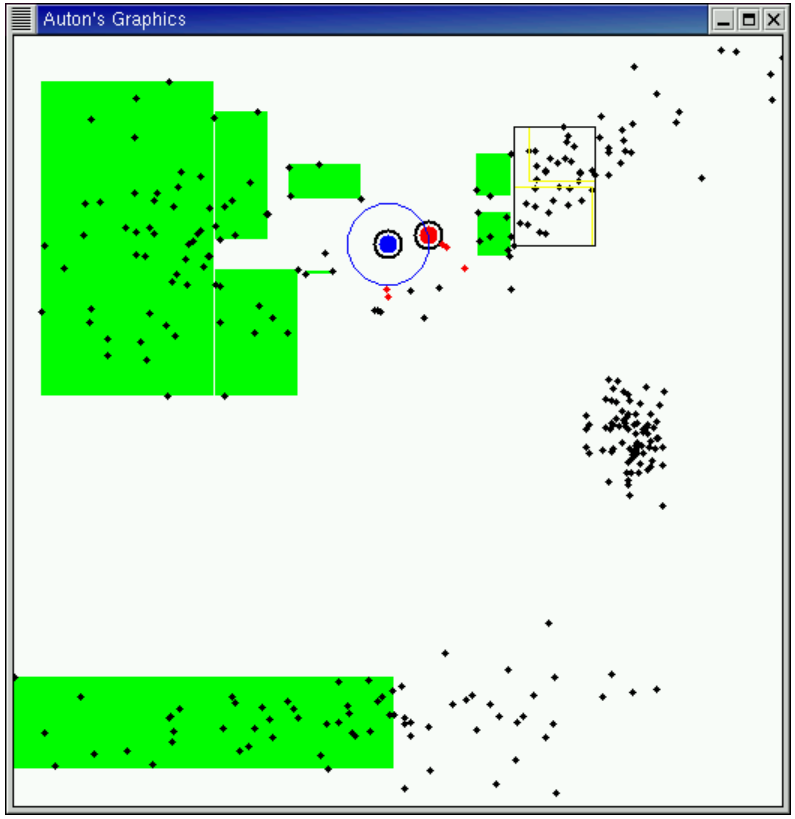
Example with many points in 2D



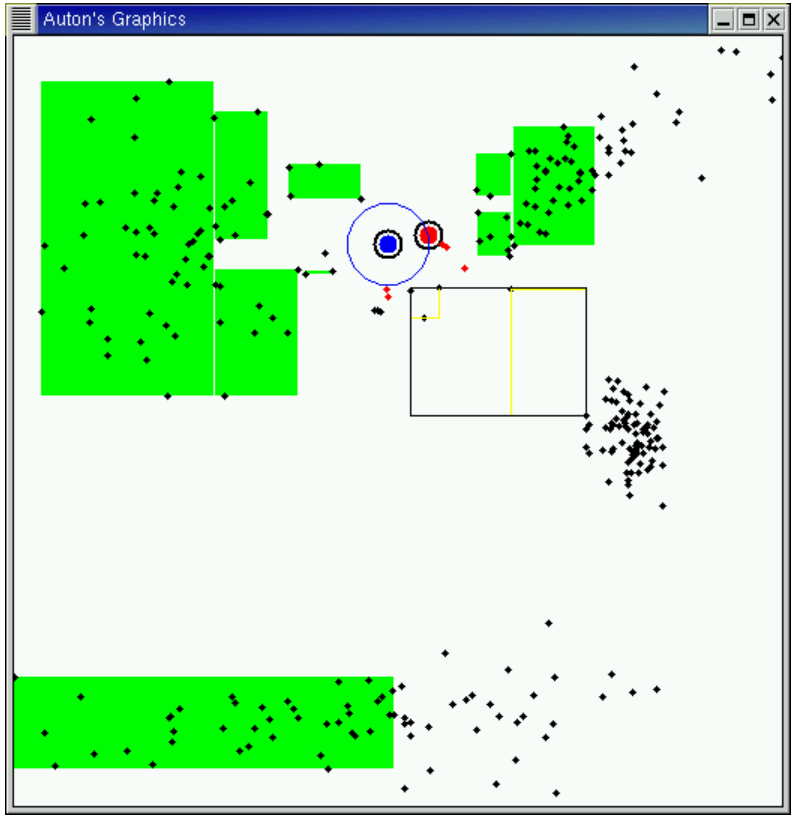
Example with many points in 2D



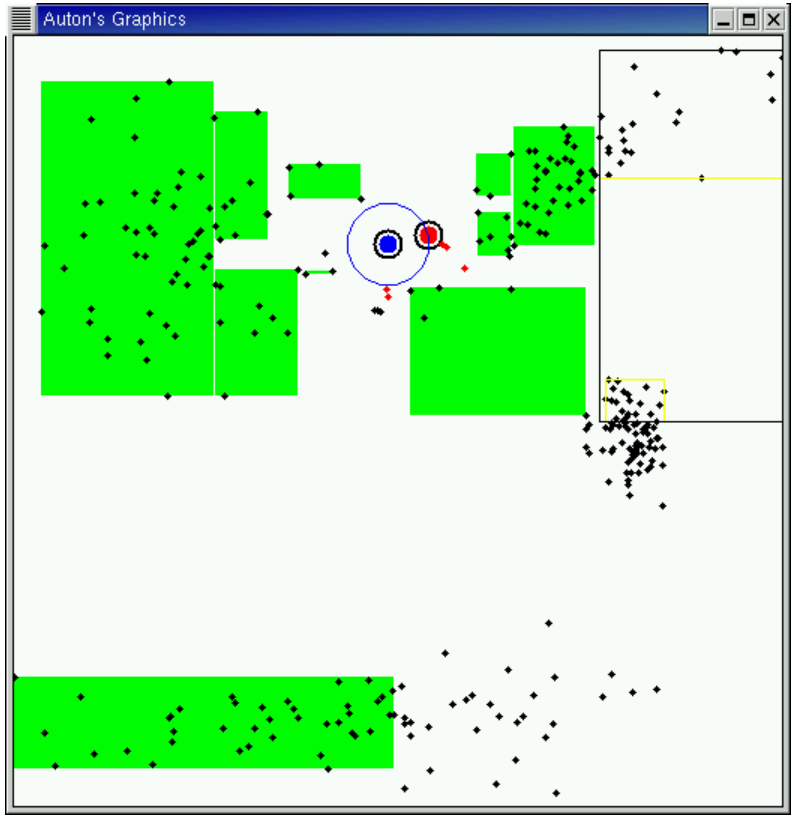
Example with many points in 2D



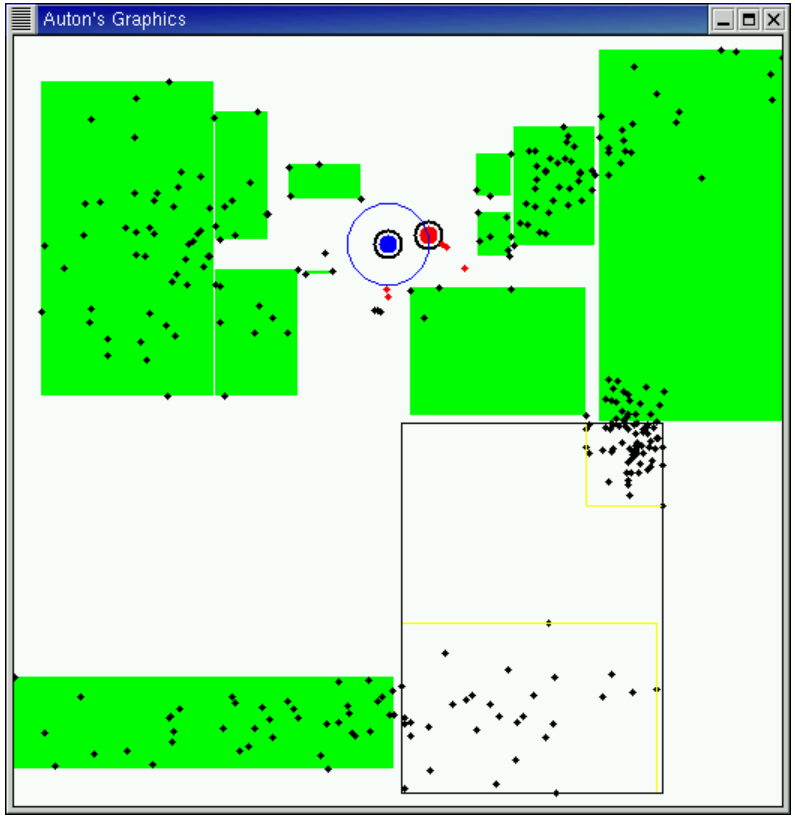
Example with many points in 2D



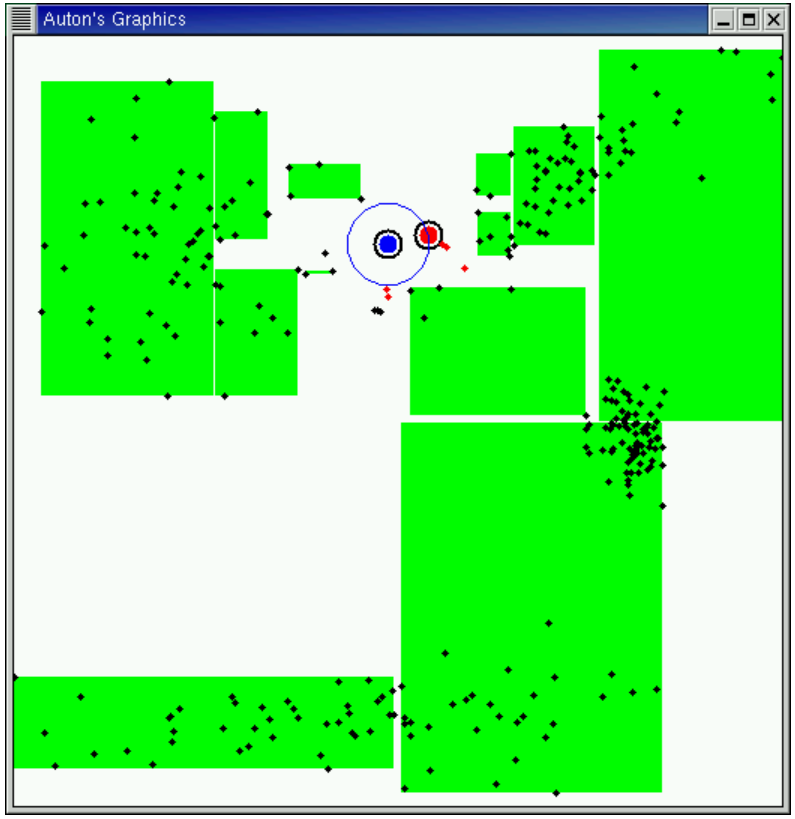
Example with many points in 2D



Example with many points in 2D

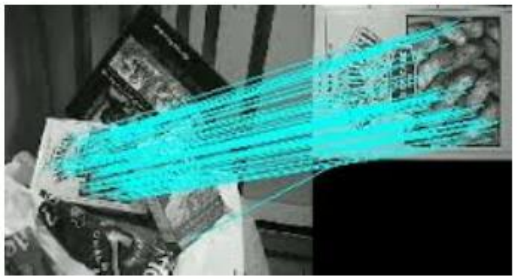


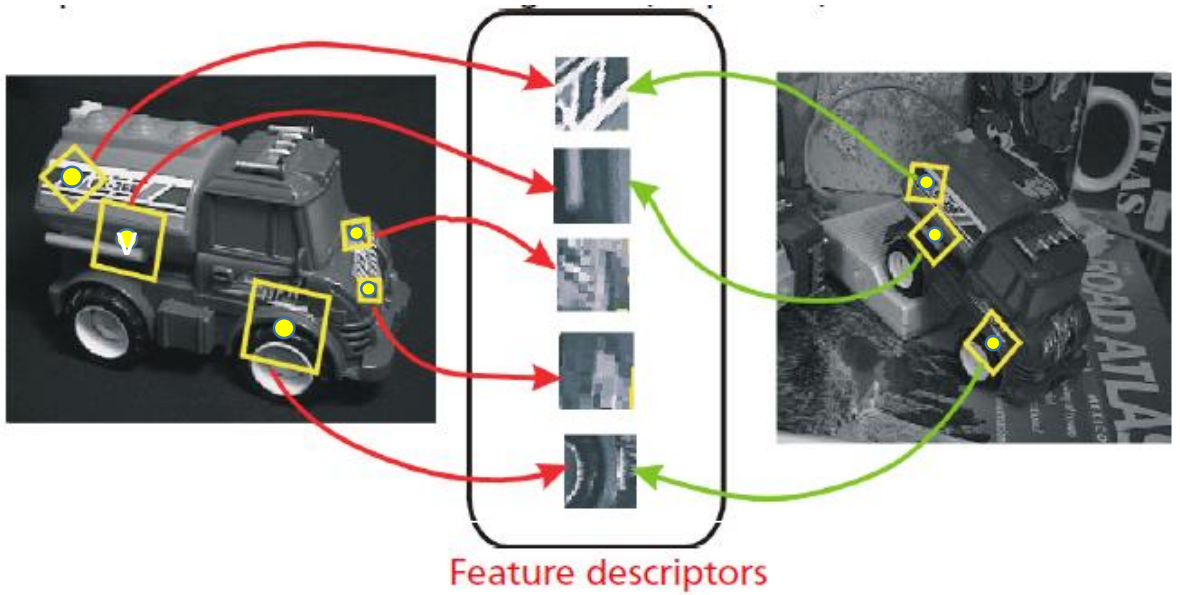
Example with many points in 2D



Roadmap: matching 2 Images (appearance & geometry)

- Find interest points (including different scales)
- Find orientated patches around interest points to capture appearance
- Encode patch in a descriptor
- Find matching patches according to appearance (similar descriptors)
- Verify matching patches according to geometry





Reading for next class

This lecture:

- Photometric image formation (sec 2.2)
- Camera Types and Hardware (sec 2.3)
- Appearance matching: (sec. 4.1.2-4.1.3)

Next lecture:

- Two-view Geometry (Hartley Zissermann)