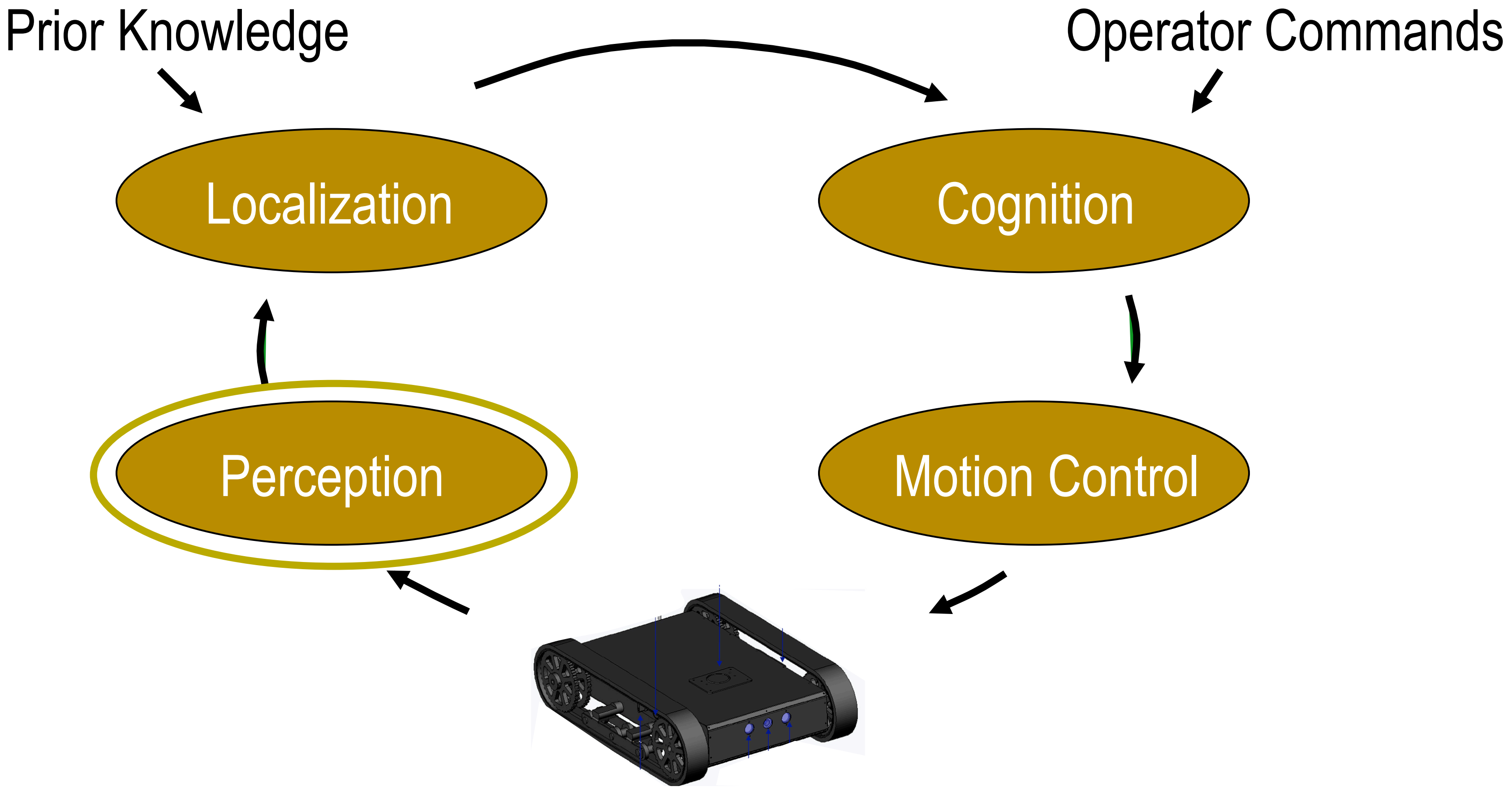
E160 – Lecture 7
Autonomous Robot Navigation

Instructor: Chris Clark

Semester: Spring 2016

Figures courtesy of Siegwart & Nourbakhsh

Control Structures
Planning Based Control

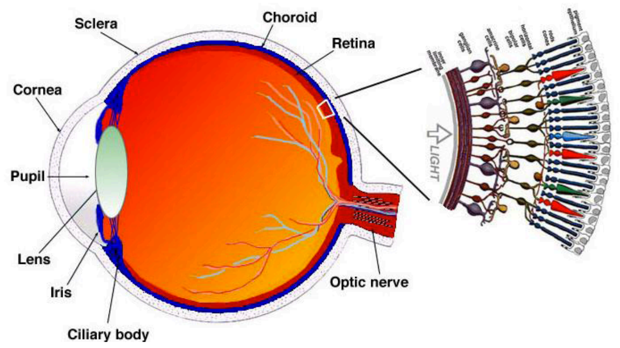


Outline – Vision Systems

1. Introduction
2. Features
3. Stereo Vision
4. Optical Flow
5. Color Tracking

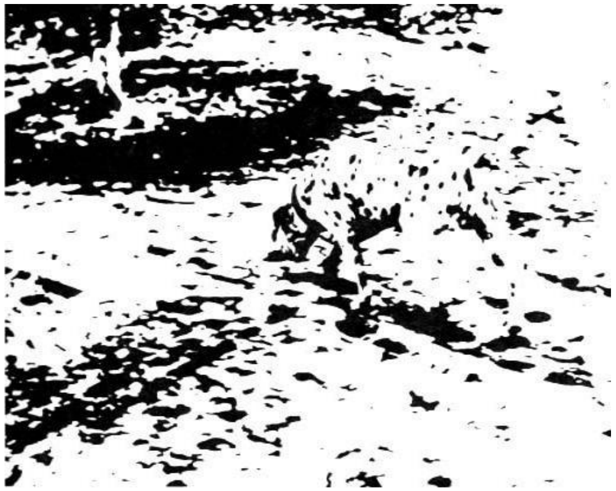
Introduction to Vision Systems

- Vision is our most powerful sense. It provides us with an enormous amount of information about the environment without direct contact.
  - Millions photoreceptors
  - Sample rate of 3 Gbytes/s
  - 60 Billion neurons used to process an image



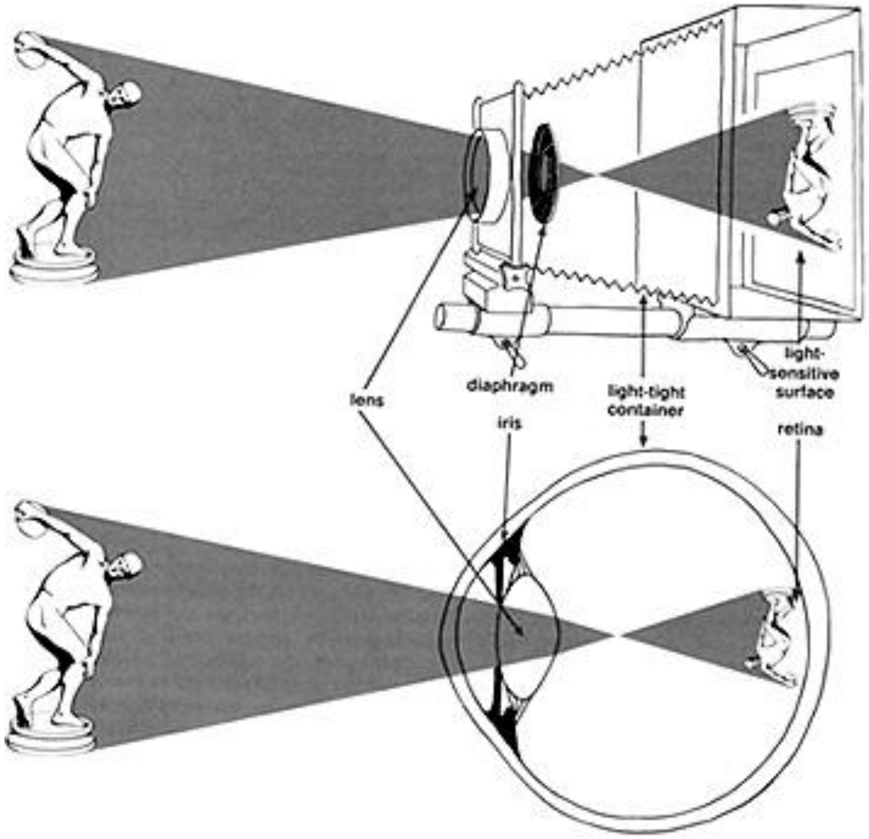
Introduction to Vision Systems

- Our visual system is very sophisticated
  - Humans can interpret images successfully under a wide range of conditions – even in the presence of very limited cues



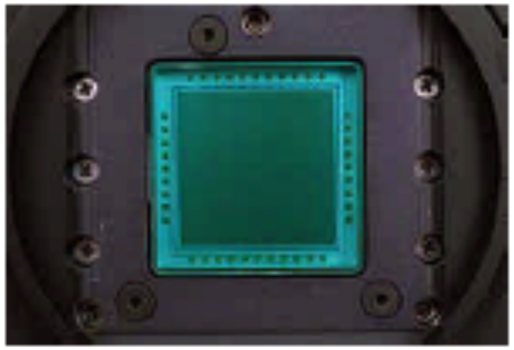
Introduction to Vision Systems

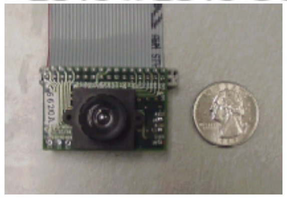
- Not sensible to copy the biology, but learn from it
    - Capture light
    - Convert to digital image
    - Process to get ”salient" information



Introduction to Vision Systems

- There exist a large number of cameras capable of getting these images…


2048 x 2048 CCD array



Orangemicro iBOT Firewire

Sony DFW-X700

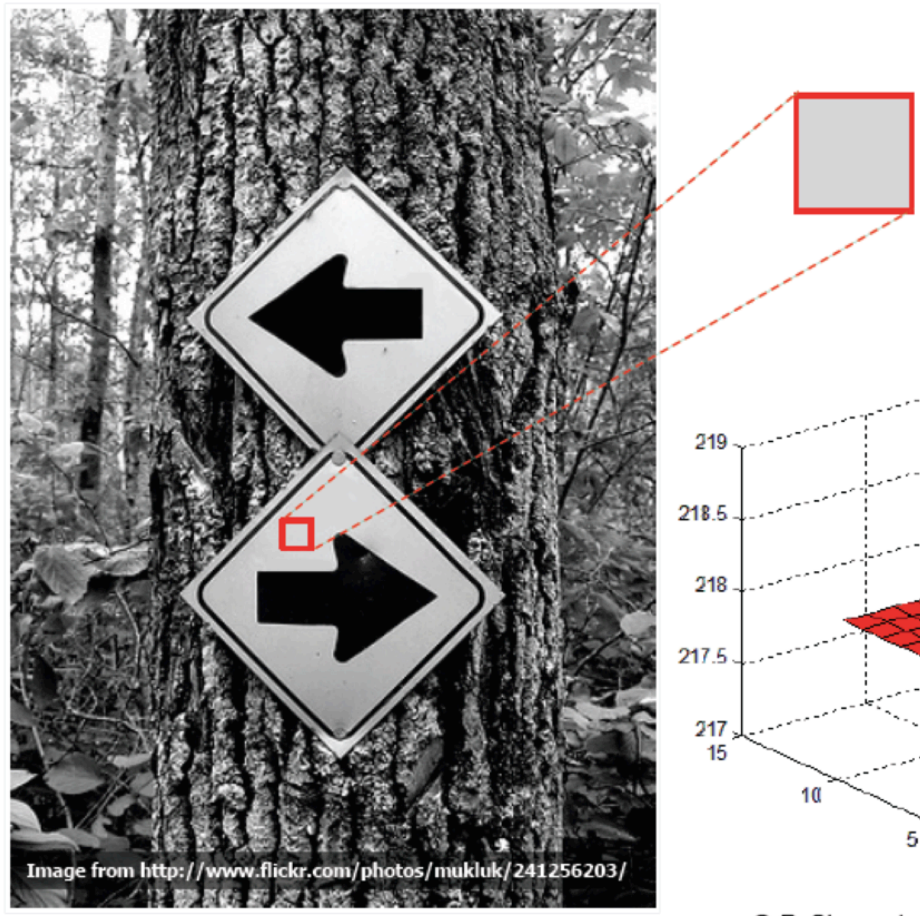
Cannon IXUS 300

Outline – Vision Systems

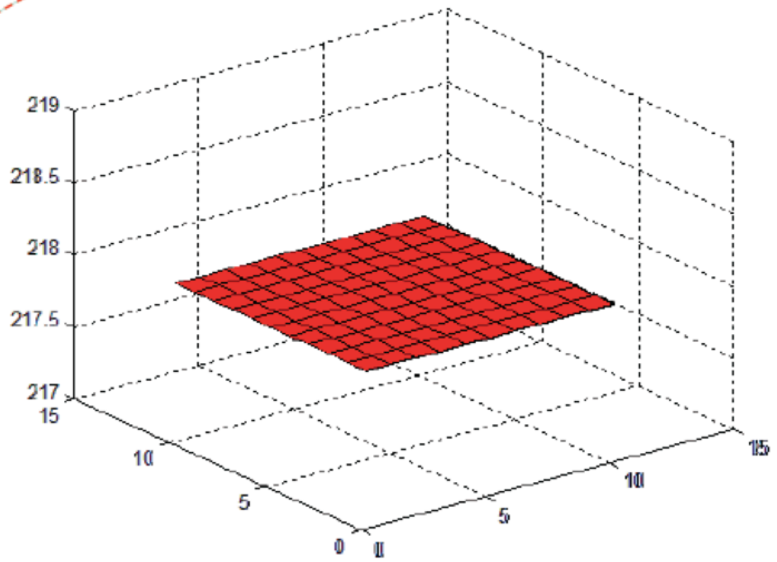
Filtering

- The image is represented as an array of intensity values.
- What useful information can we extract from these intensities?

Filtering

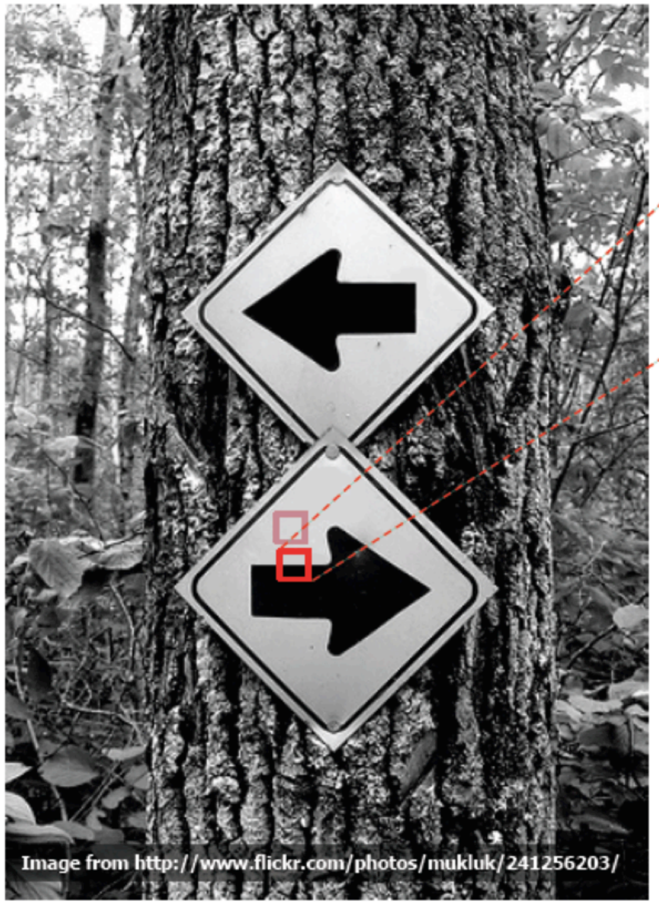


218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218
218	218	218	218	218	218	218	218	218	218	218

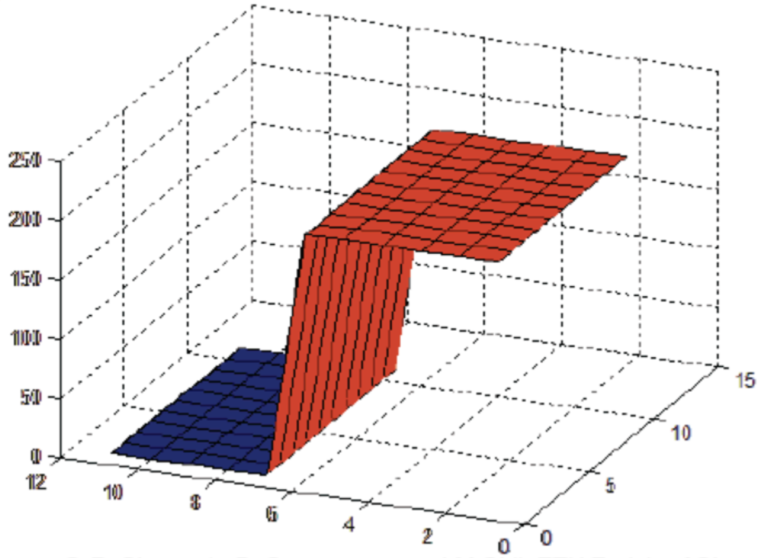


© R. Siegwart , D. Scaramuzza and M.Chli, ETH Zurich - ASL

Filtering

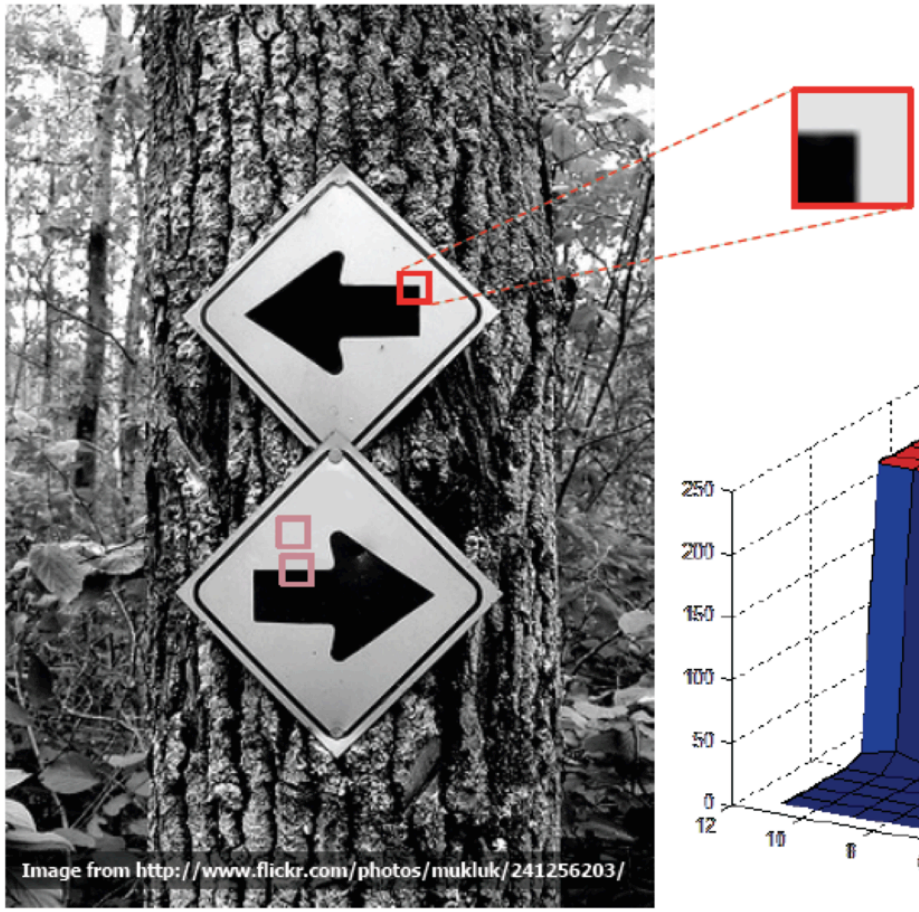


208	208	208	208	208	208	208	208	208	208	208
208	208	208	208	208	208	208	208	208	208	208
208	208	208	208	208	208	208	208	208	208	208
208	208	208	208	208	208	208	208	208	208	208
208	208	208	208	208	208	208	208	208	208	208
208	207	208	208	208	208	208	208	208	208	208
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

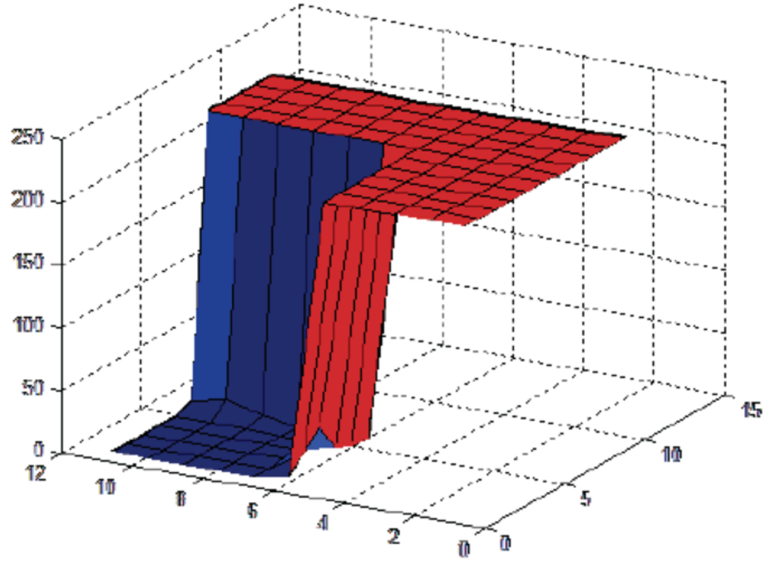


© R. Siegwart , D. Scaramuzza and M.Chli, ETH Zurich - ASL

Filtering



229	229	229	229	229	229	229	229	229	229	229
229	229	229	229	229	229	229	229	229	229	229
229	229	229	229	229	229	229	229	229	229	229
229	229	229	229	229	229	229	229	229	229	229
229	229	229	229	229	230	229	229	229	229	229
5	17	31	7	1	0	229	229	229	229	229
0	0	1	0	0	0	229	229	229	229	229
0	0	0	0	0	0	229	229	229	229	229
0	0	0	0	1	4	229	229	229	229	229
0	0	0	0	0	11	229	229	229	229	229
0	0	0	0	0	5	229	229	229	229	229

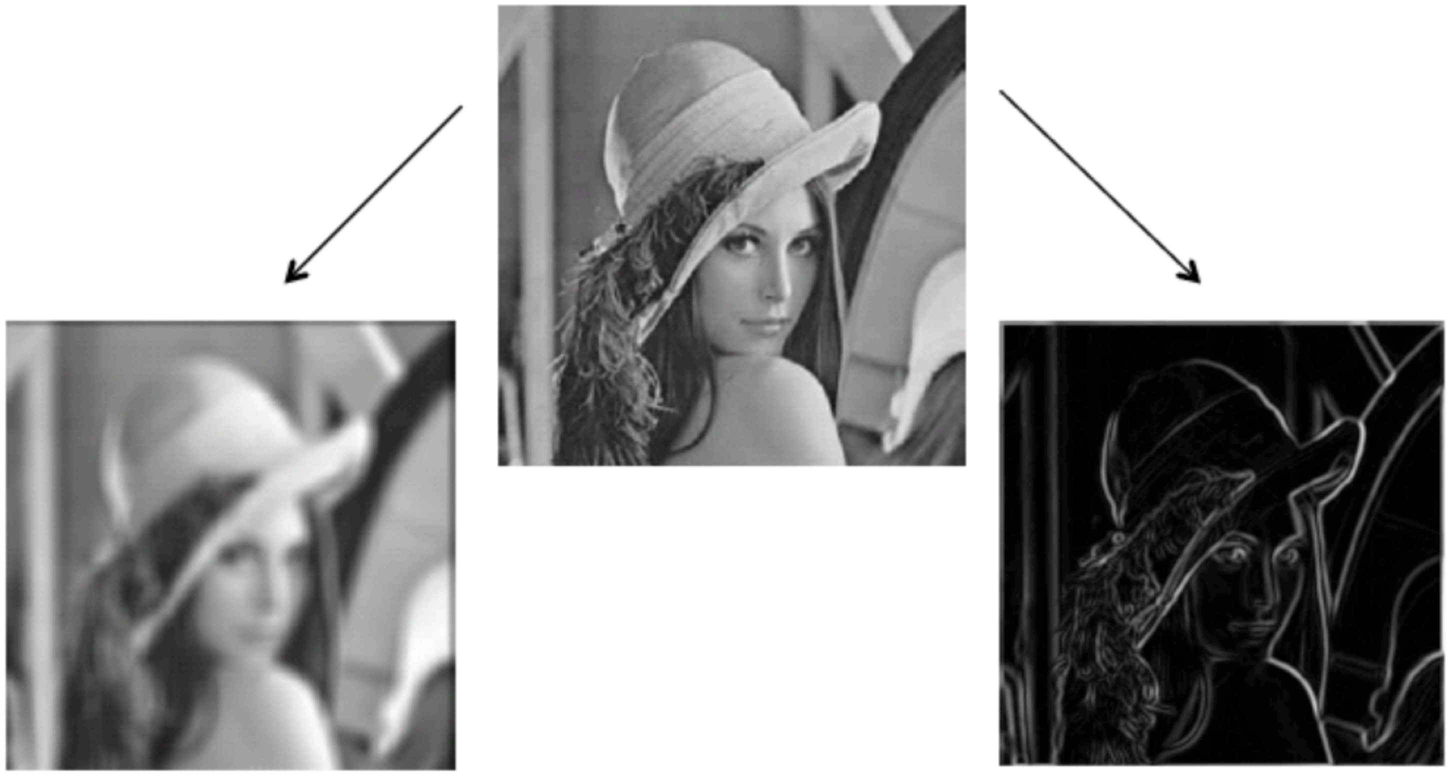


© R. Siegwart , D. Scaramuzza and M.Chli, ETH Zurich - ASL

Filtering

- In other courses, we construct **frequency filters** based on frequency, e.g. lowpass/ highpass filters
- In Image processing, we construct **spatial filters**

Filtering



Lowpass filtered image

Highpass filtered image

Filtering

- Spatial Filters
  - Given an Intensity Image $I(x,y)$
  - Let $S_{x,y}$ be the neighborhood of pixels around the pixel at (x,y)
  - We can create a filtered image $J=F(I(x,y))$
    - Each pixel (x,y) in J is calculated based on a function of S_{xy}

Filtering

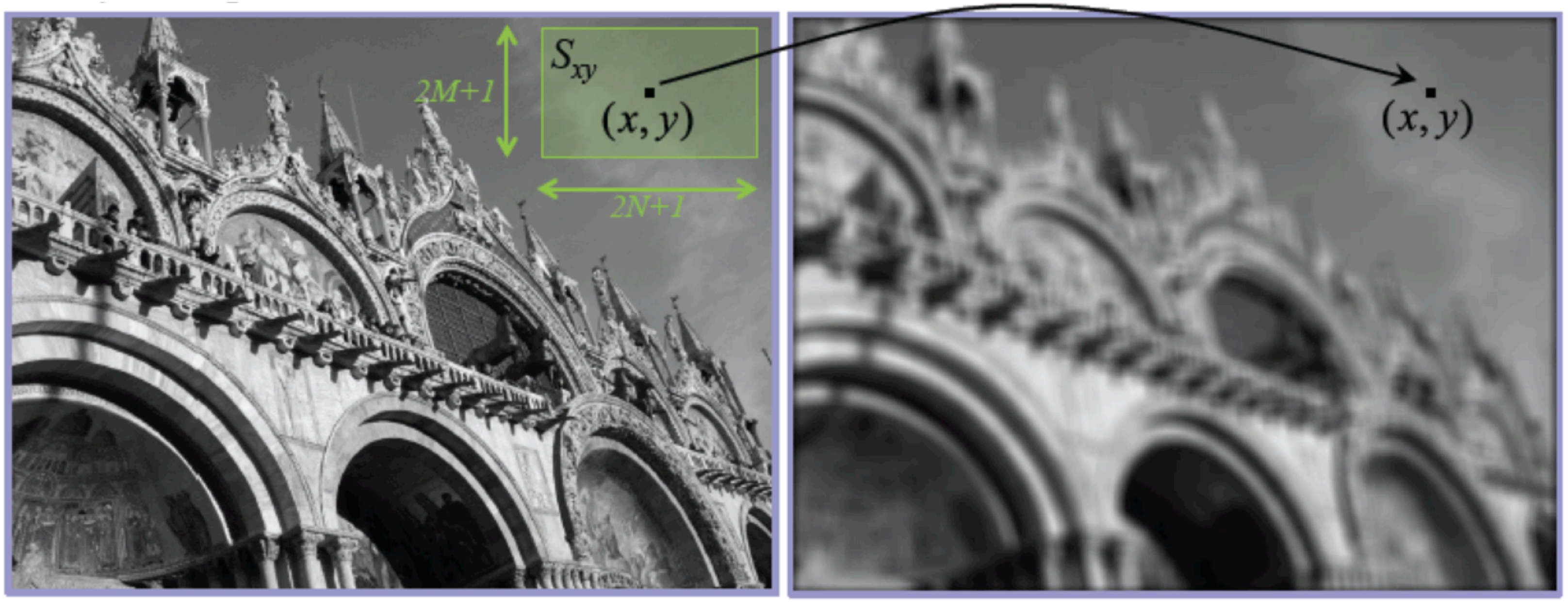


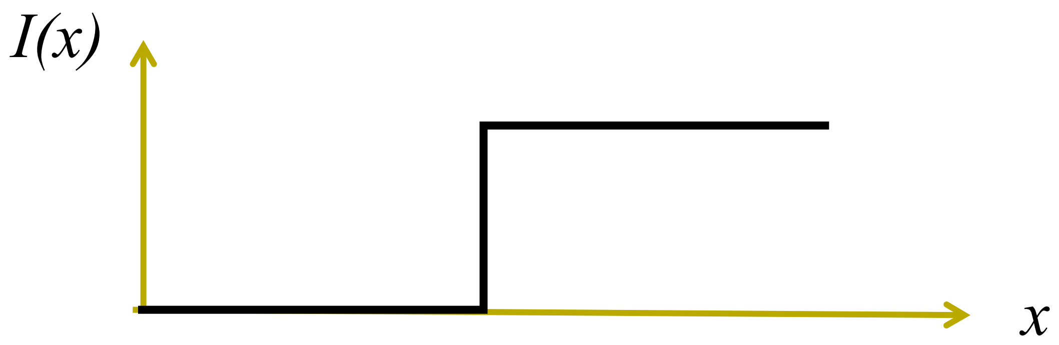
Image I

Filtered Image $J = F(I)$

$$J(x,y) = \frac{\sum_{(r,c)\in S_{xy}} I(r,c)}{(2M+1)(2N+1)}$$

Filtering

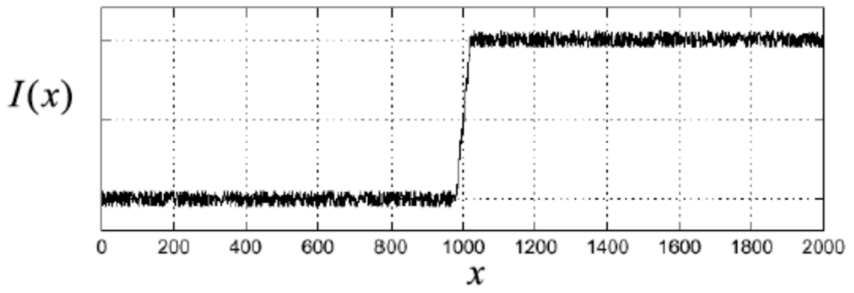
- Edge Detection
  - Edge = intensity discontinuity in one direction



  - First order derivative in 1D should be large
  - 2nd order derivative can also be used…

Filtering

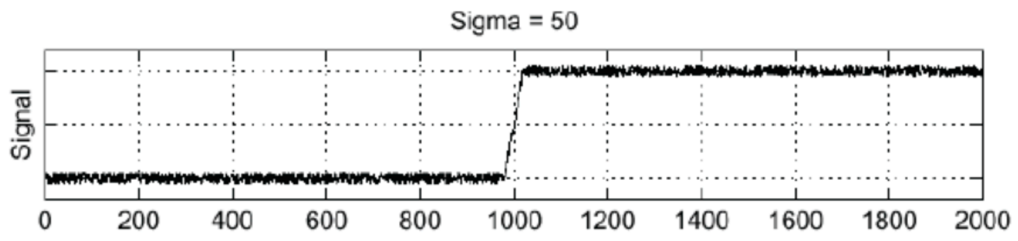
- Edge Detection
  - Actual image intensity might look more like this



Filtering

- Edge Detection

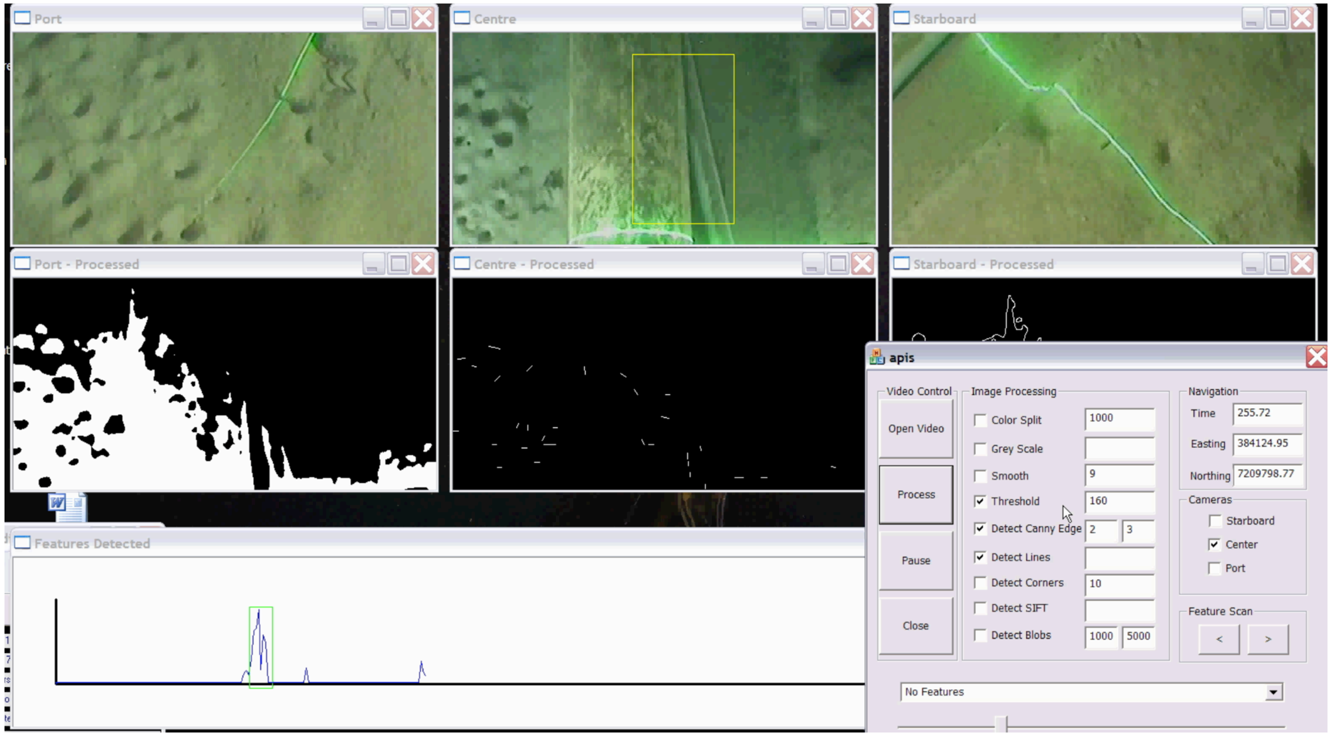
$I(x)$



Peak indicates Edge

Filtering

- Pipe Tracking Example



Video Control: Open Video, Process, Pause, Close

Image Processing

☐ Color Split	1000	
☐ Grey Scale		
☐ Smooth	9	
✓ Threshold	160	
✓ Detect Canny Edge	2	3
✓ Detect Lines		
☐ Detect Corners	10	
☐ Detect SIFT		
☐ Detect Blobs	1000	5000

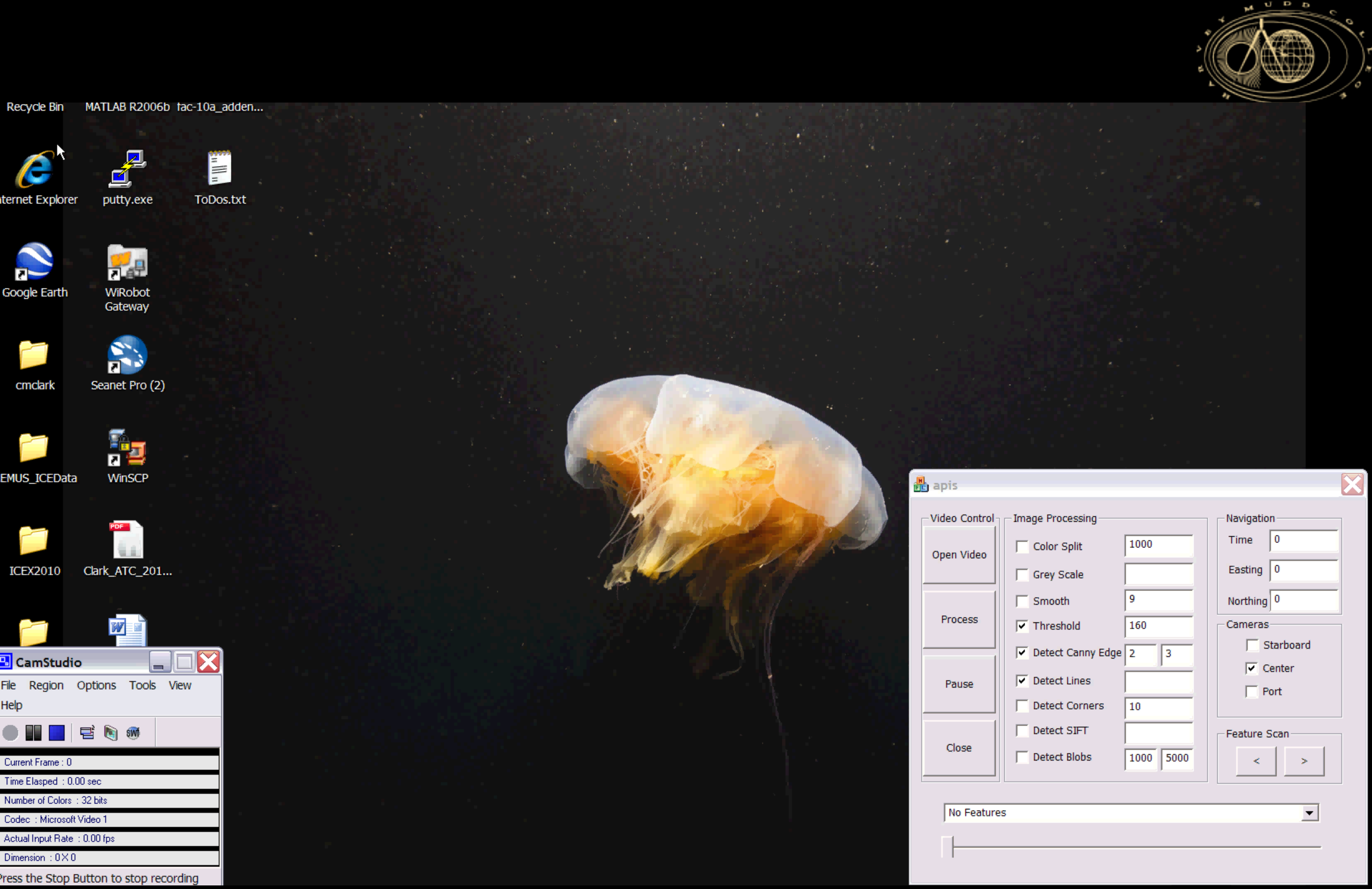
Navigation

Time	255.72
Easting	384124.95
Northing	7209798.77

Cameras: ☐ Starboard, ✓ Center, ☐ Port

Feature Scan: < >

No Features



CamStudio

Current Frame : 0

Time Elasped : 0.00 sec

Number of Colors : 32 bits

Codec : Microsoft Video 1

Actual Input Rate : 0.00 fps

Dimension : 0 X 0

ress the Stop Button to stop recording

apis

Video Control

Image Processing

☐ Color Split	1000
☐ Grey Scale	
☐ Smooth	9
✓ Threshold	160
✓ Detect Canny Edge	2 / 3
✓ Detect Lines	
☐ Detect Corners	10
☐ Detect SIFT	
☐ Detect Blobs	1000 / 5000

Navigation

Time	0
Easting	0
Northing	0

Cameras

☐ Starboard

✓ Center

☐ Port

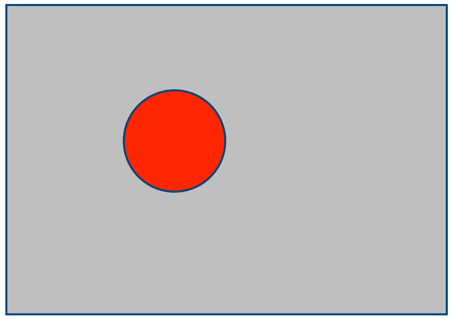
Feature Scan

No Features

Outline – Vision Systems

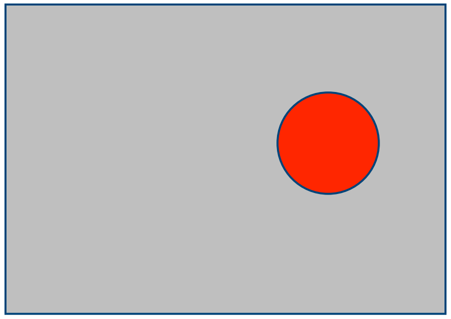
Sensors: Vision Systems

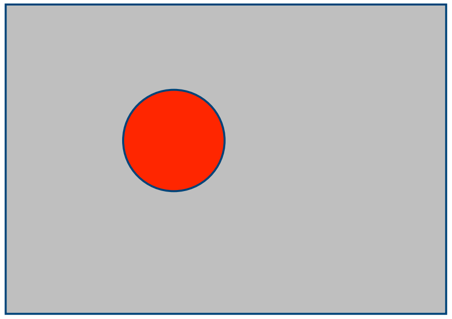
- Monocular Vision:
  - Problem with monocular vision is that you can't tell how far something is from the robot. No Range information!
  - Consider the following image of a red ball:

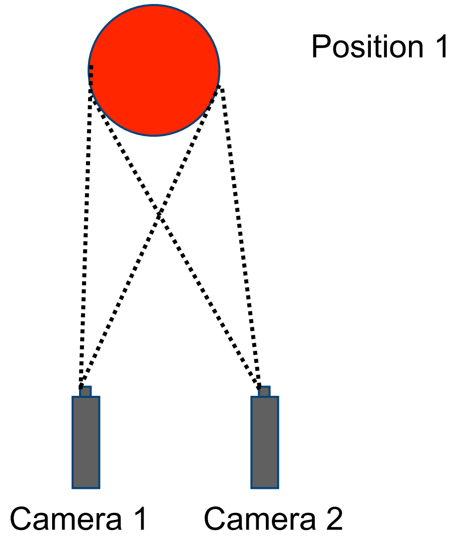


Sensors: Vision Systems

- Stereo Vision:
  - Using two cameras provides enough information to give us the range to the ball
  - The intersection of the two cones must be where the ball lies.

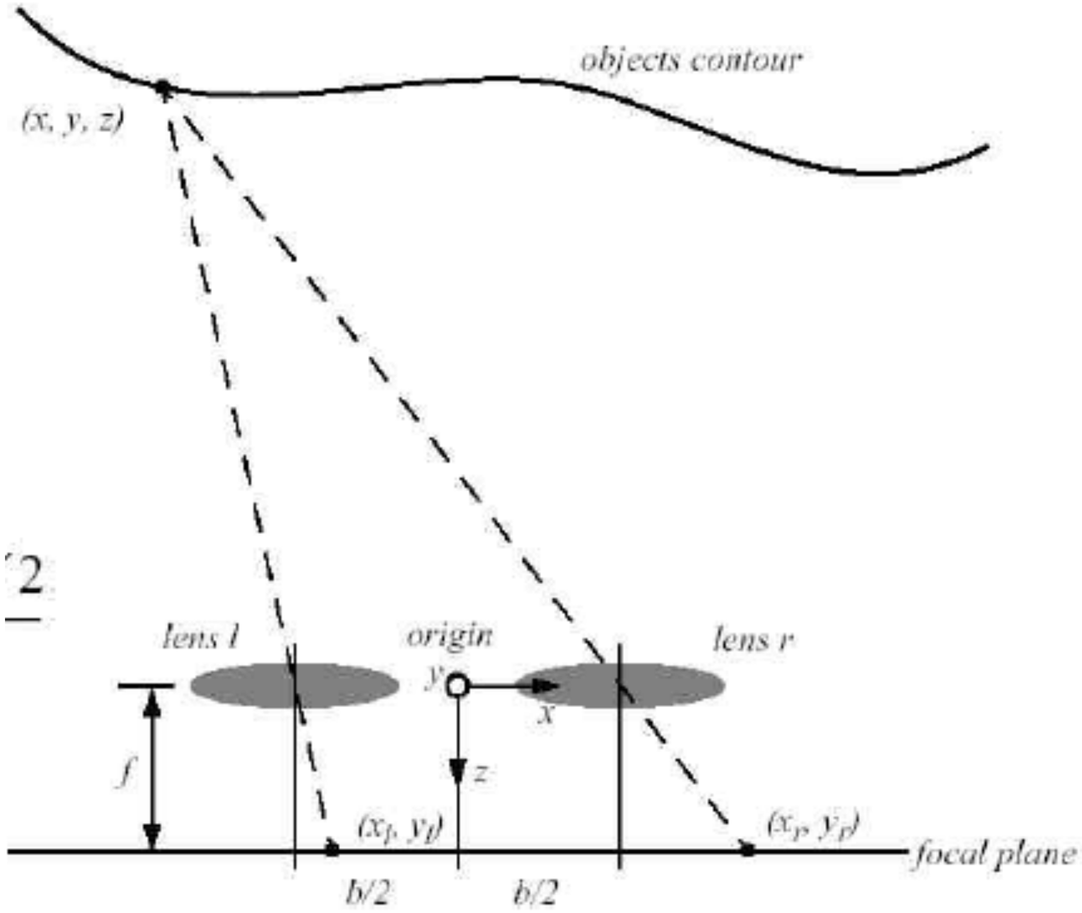

Camera 1 Image


Camera 2 Image



Sensors: Stereo Vision Systems

- Consider idealized camera geometry
- Compare projection of a target on 2 image planes.



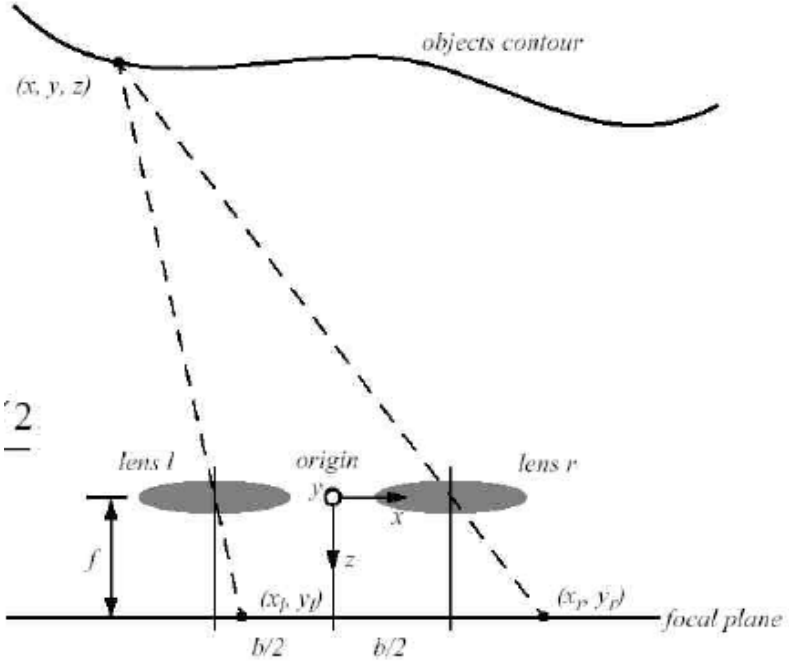
Sensors: Stereo Vision Systems

$$\frac{x_l}{f} = \frac{x+b/2}{z} \text{ and } \frac{x_r}{f} = \frac{x-b/2}{z}$$

$$\frac{x_l - x_r}{f} = \frac{b}{z}$$

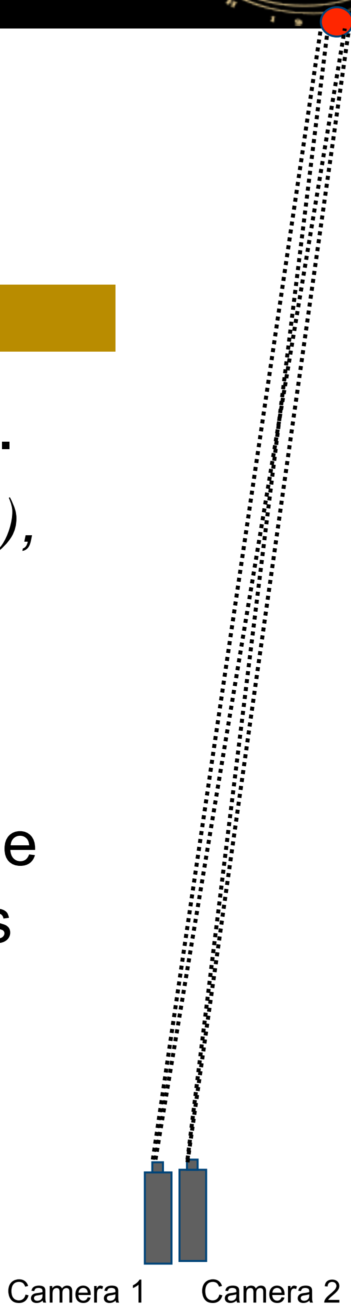
$$x = b\frac{(x_l+x_r)/2}{x_l-x_r} \; ; \; y = b\frac{(y_l+y_r)/2}{x_l-x_r}$$

$$z = b\frac{f}{x_l-x_r}$$



Sensors: Stereo Vision Systems

- There is poor accuracy on far objects.
  - Farther objects have less disparity $(x_l - x_r)$, making it difficult to get accurate depth measurements.
  - This is similar to having cameras colocated so that the object appears to be at the same position in both images. This ends up working like monocular vision



Sensors: Stereo Vision Systems

- Can we tell the difference between something *100* meters away and *101* meters away given our $f = 0.1m$ and $b = 0.5m$?
  - The disparity for $z = 100$ is $(0.1)(0.5)/100 = 0.0005$
  - The disparity for $z = 101$ is $(0.1)(0.5)/101 = 0.000495$
- How about the difference between something 10 meters away and 11 meters away?
  - The disparity for $z = 10$ is $(0.1)(0.5)/10 = 0.005$
  - The disparity for $z = 11$ is $(0.1)(0.5)/11 = 0.0045$
- We can see there is greater difference between the disparities when the object is close, so…
- Easier to get accurate range when the object is closer!

Sensors: Stereo Vision Systems

- Disparity is proportional to b

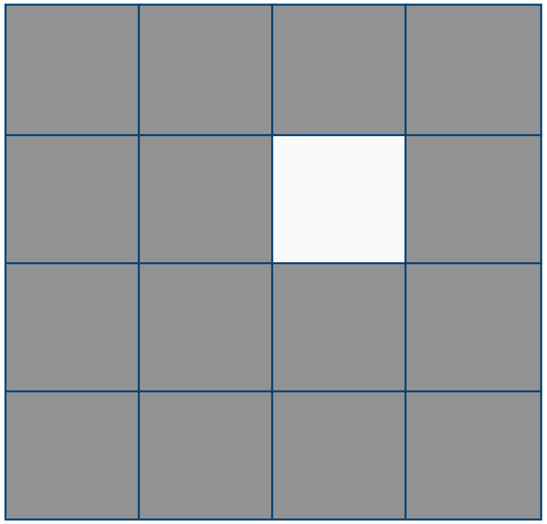
$$\frac{x_l - x_r}{f} = \frac{b}{z}$$

Sensors: Stereo Vision Systems

- We don’t always have red balls on gray backgrounds…
  - “Zero crossing of Laplacian of Gaussian (ZLoG) is widely used method of identifying feature from 2 images”

Sensors: Stereo Vision Systems

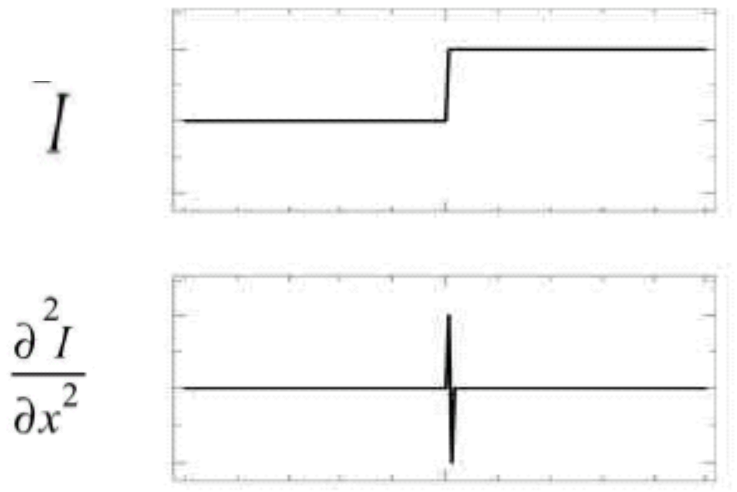
- Consider the 4x4 image:
  - $$I(i,j) = \begin{cases} 0.8 & \text{if } i=3, j=3 \\ 0.1 & \text{else} \end{cases}$$
  - Let's see how we can identify the bright spot



Sensors: Stereo Vision Systems

- ZLOG method finds features with high intensity contrast as measured by Laplacian.

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$



- The zero-crossing indicates a feature!

Sensors: Stereo Vision Systems

- To implement the Laplacian, an approximate function is used.
  - The convolution with P:

$$L = P \otimes I \qquad P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

  - That is, each pixel of L is made by summing over adjacent pixels:

$$L(i,j) = I(i-1, j) + I(i+1, j) + I(i, j-1) + I(i, j+1) - 4I(i, j)$$

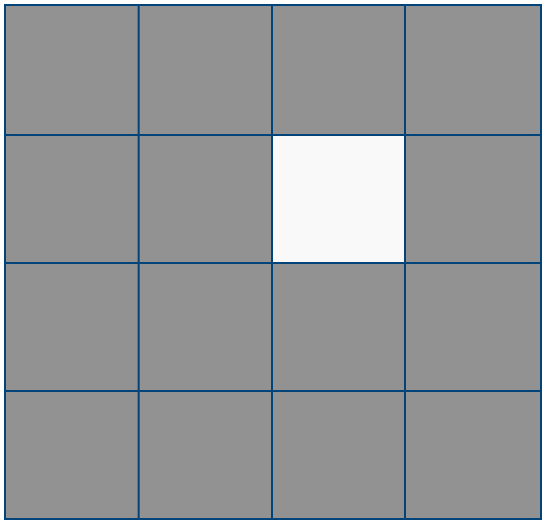
Sensors: Stereo Vision Systems

- Apply the Laplacian to our 4x4 image:

$$L(3,2) \quad = \quad 0.1 + 0.1 + 0.1 + 0.8 - 4\ (0.1) = +0.7$$

$$L(3,3) \quad = \quad 0.1 + 0.1 + 0.1 + 0.1 - 4\ (0.8) = -\ 2.8$$

$$\vdots$$



Sensors: Stereo Vision Systems

- Apply the Laplacian to our 4x4 image:

$$L(3,2) \quad = \quad 0.1 + 0.1 + 0.1 + 0.8 - 4\,(0.1) = +0.7$$

$$L(3,3) \quad = \quad 0.1 + 0.1 + 0.1 + 0.1 - 4\,(0.8) = -\,2.8$$

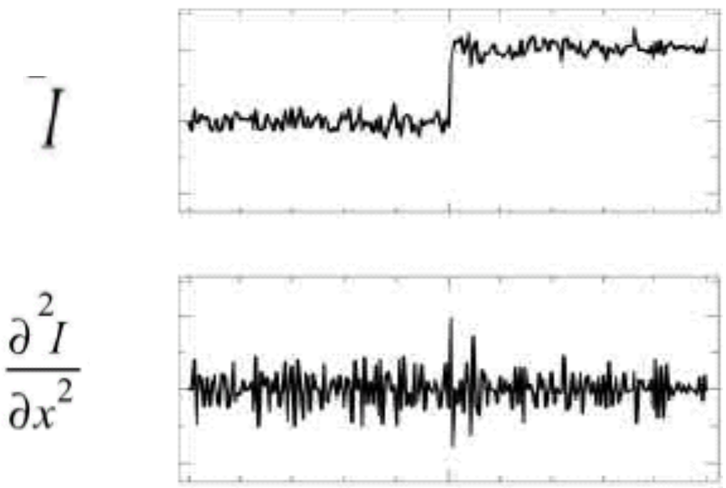
$$\vdots$$

0.0	0.0	0.7	0.0
0.0	0.7	-2.8	0.7
0.0	0.0	0.7	0.0
0.0	0.0	0.0	0.0

Zero crossing

Sensors:
Stereo Vision Systems

- There is a problem that noise makes it difficult to detect zero crossings.



- To deal with this we use a gaussian operator to smooth/blur the image.

Sensors: Stereo Vision Systems

- To deal with noise we use a gaussian operator to smooth/blur the image.

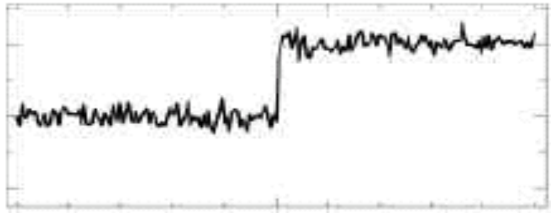
  - *filtering through Gaussian smoothing*

$$G = \begin{bmatrix} \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \\ \frac{2}{16} & \frac{4}{16} & \frac{2}{16} \\ \frac{1}{16} & \frac{2}{16} & \frac{1}{16} \end{bmatrix}$$

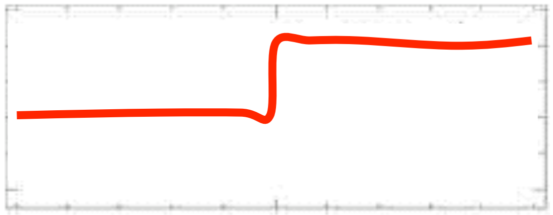
Sensors: Stereo Vision Systems

- The Gaussian operator looks like an operator that averages the pixel values
- This has the effect of smoothing the curve:

$\bar{I}$



$G \otimes I$



Sensors:
Stereo Vision Systems

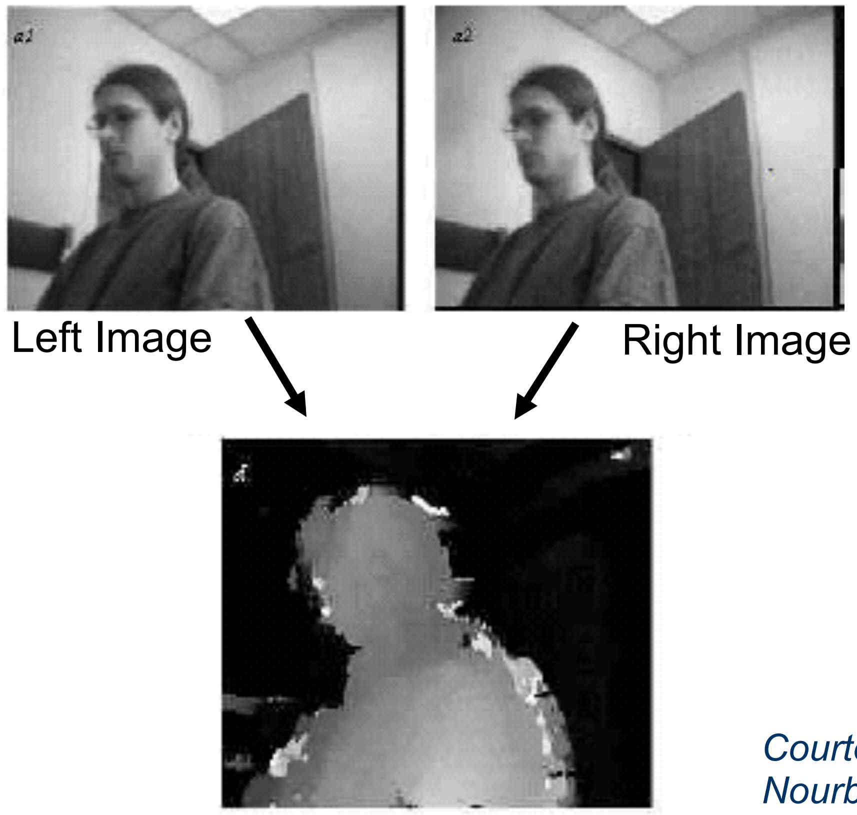
- Apply the Gaussian operator to the image will blur/smooth it so that zero-crossings will not occur simply due to noise.

Sensors: Stereo Vision Systems

- ZLog Method:
  1. Gaussian Filter
  2. Laplacian Filter
  3. Mark features as zero crossing
  4. Match features between images
  5. Use geometry to recover depth map

Sensors: Stereo Vision Systems

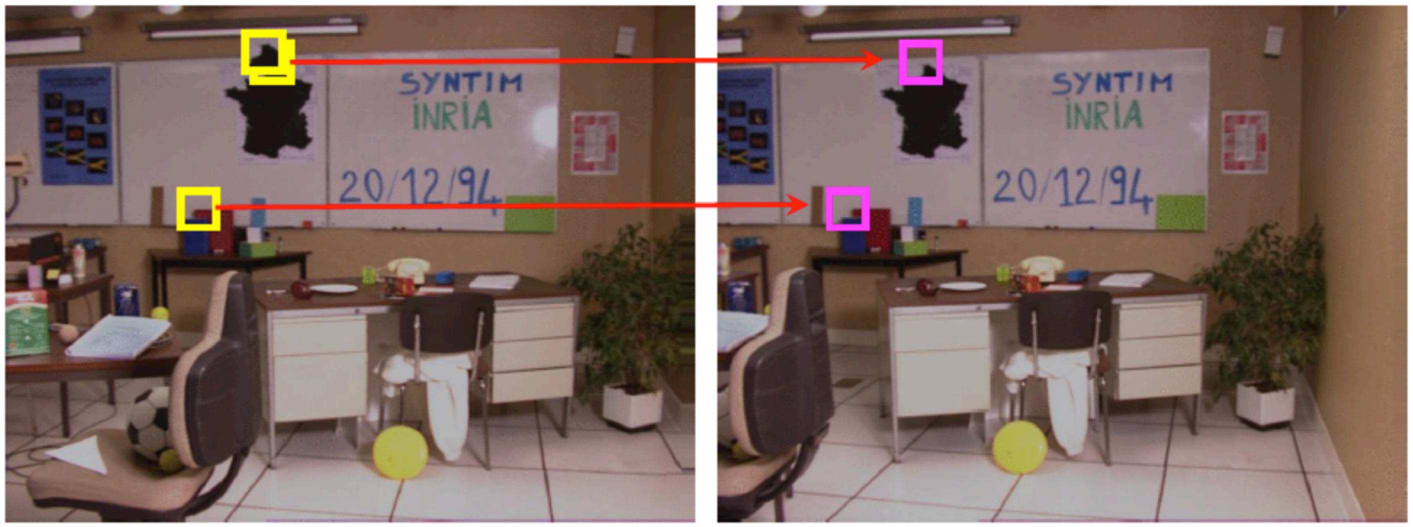
- ZLog Method Example:



Courtesy of Siegwart & Nourbakhsh

Sensors: Stereo Vision Systems

- Key issue is the correspondence problem
  - Identifying same feature from two cameras



© R. Siegwart , M.Chli and D. Scaramuzza, ETH Zurich - ASL

Outline – Vision Systems

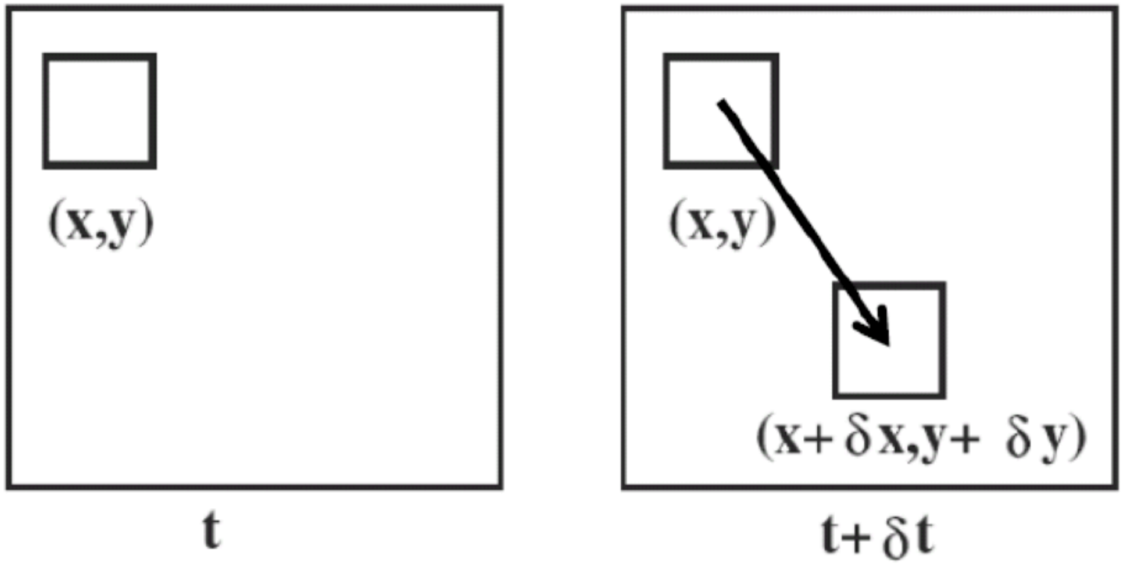
1. Introduction
2. Features
3. Stereo Vision
4. Optical Flow
5. Color Tracking

Sensors: Optical Flow

- Motion Field
  - A velocity vector is assigned to every point in an image.
  - Given velocity of point in image, determine velocity of point in the environment.
- Optical Flow
  - Motion of brightness patterns in image.
  - Can be same motion as object motion.

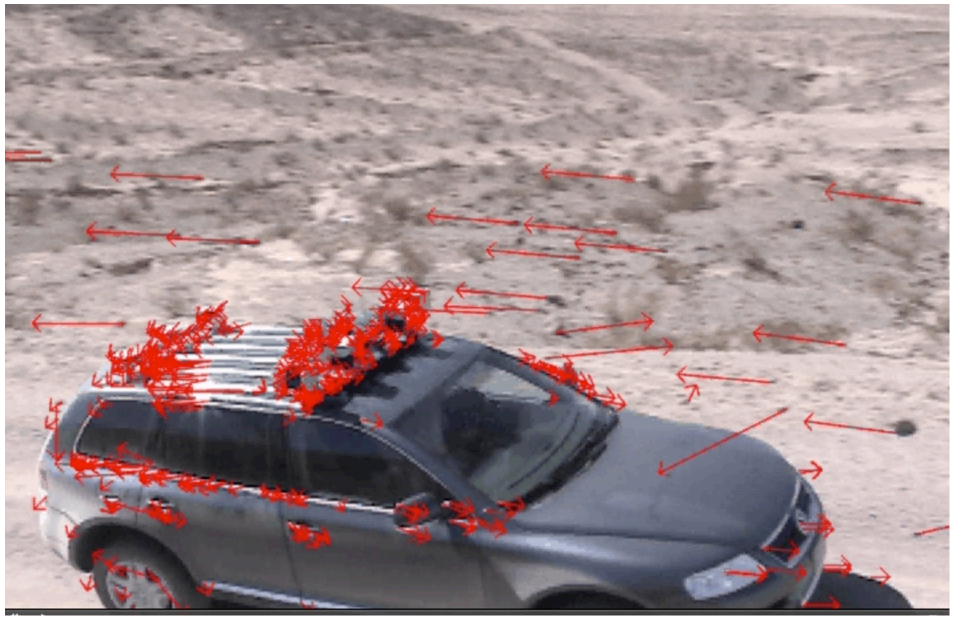
Sensors: Optical Flow

- Optical Flow
  - Computes the motion of all pixels in an image (or subsets of pixels)



Sensors: Optical Flow

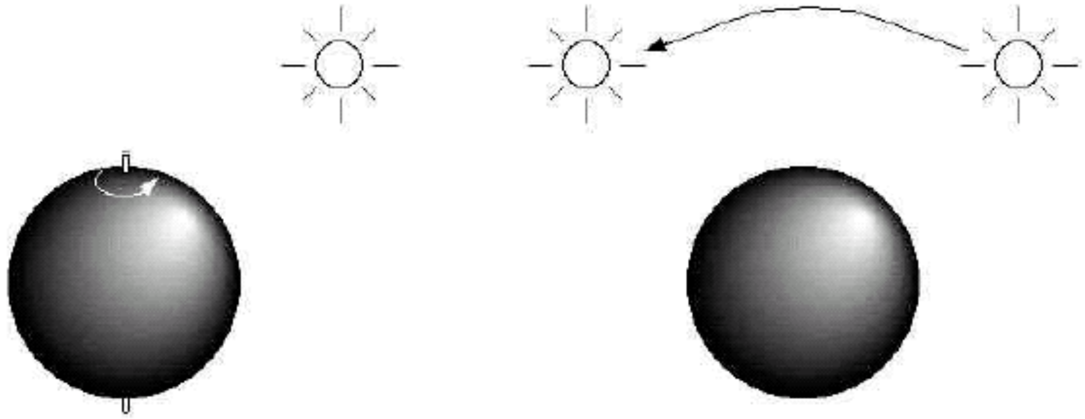
- Optical Flow



https://www.youtube.com/watch?v=ysGM3CfBVpU

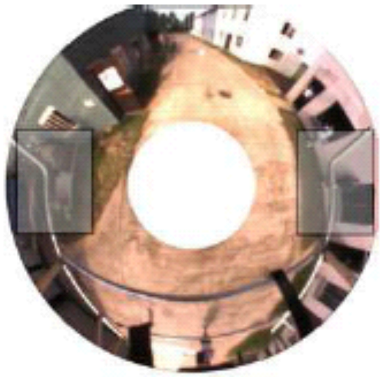
Sensors: Optical Flow

- Problem: with Optical Flow is not always same as motion field.
  - Occlusions lead to discontinuities
  - Moving Light sources
  - Symmetrical Shapes

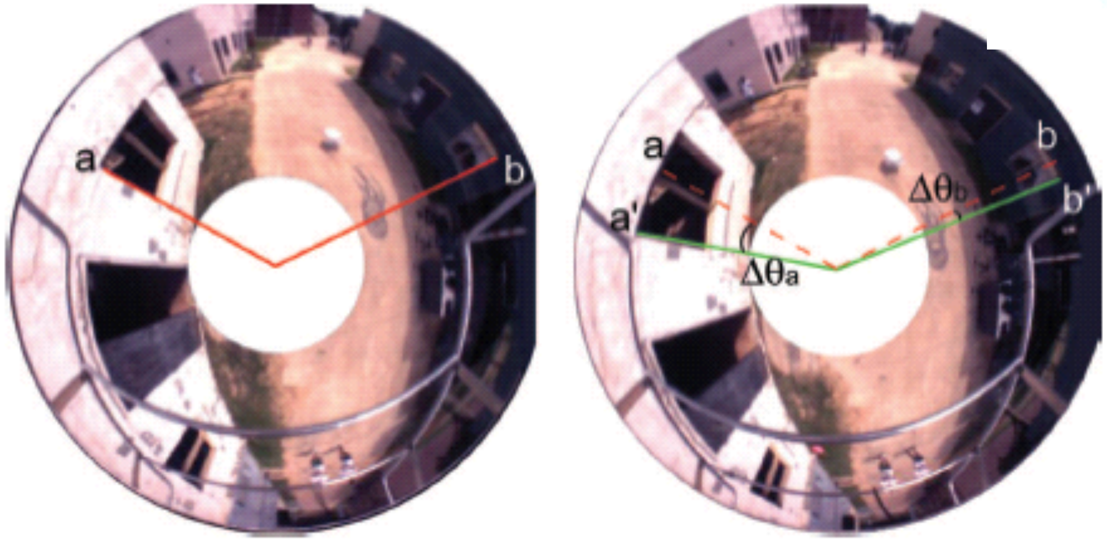


Sensors: Optical Flow for Ravine Navigation

Helicopter equipped with an OmniDirectional Camera



Regions used for optical flow calc's.



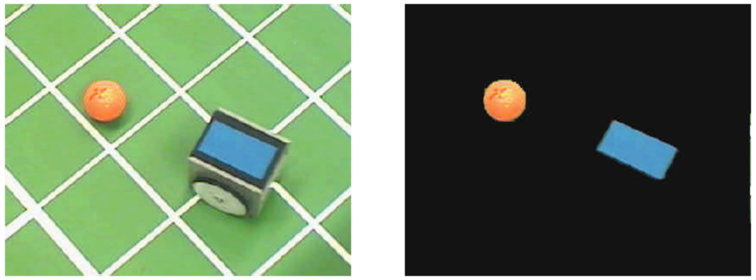
Courtesy of S. Hrabar and G. S. Sukhatme, USC

Outline – Vision Systems

1. Introduction
2. Features
3. Stereo Vision
4. Optical Flow
5. Color Tracking

Sensors: Color Tracking

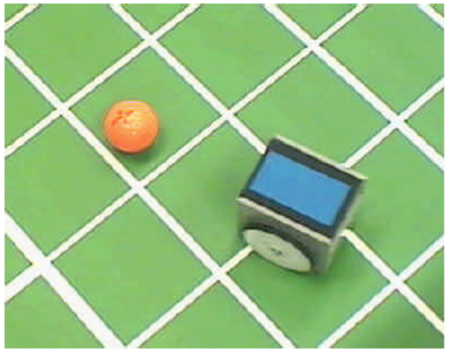
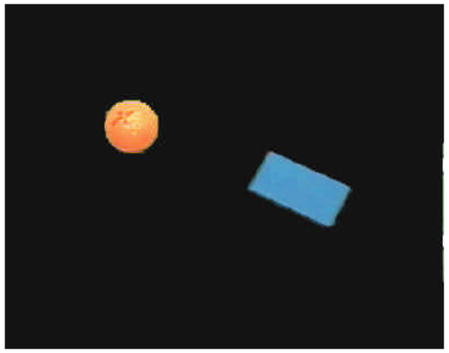
- Goal:
  - Given a color image, extract the pixels that have some specific color of interest.
  - Given the coordinates of these pixels, calculate the coloured object’s position in the robot frame.
  - E.g. How do we extract the ball and robot positions from the image below?



Courtesy of EPFL STeam

Sensors: Color Tracking

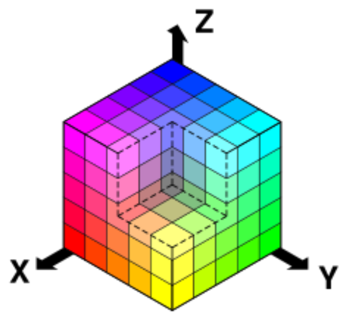
- We are given an R, G, and B value for every pixel in an image…
- Can we match these with some desired RGB values that correspond to the color of the ball and robot?

Courtesy of EPFL STeam

Sensors: Color Tracking

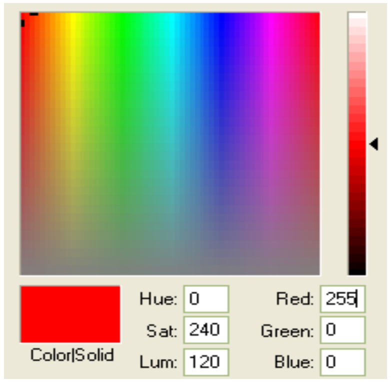
- RGB color representations assign a value from 0 to 255 to each of R, G, and B.
- These colors are additive to form white.
- Examples:
  - [0,0,255] is pure blue
  - [0,255,0] is pure green
  - [0,0,0] is black
  - [255,255,255] is white



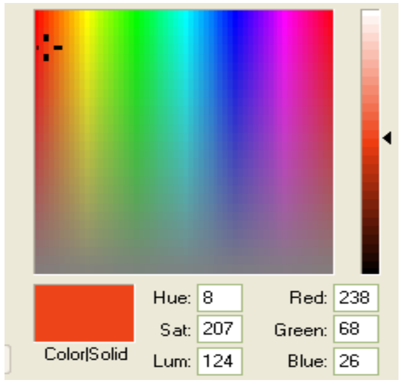
Courtesy of EPFL STeam

Sensors: Color Tracking

- Unfortunately, the RGB representation is not intuitive for measuring the closeness to a desired color.
  - E.g. what if we wanted to track a ball of color pure red - [255,0,0]?
  - We want to find all pixels that have values close to [255,0,0]
  - It is difficult to make good thresholds for any general color from which to accept as being close to our desired color.



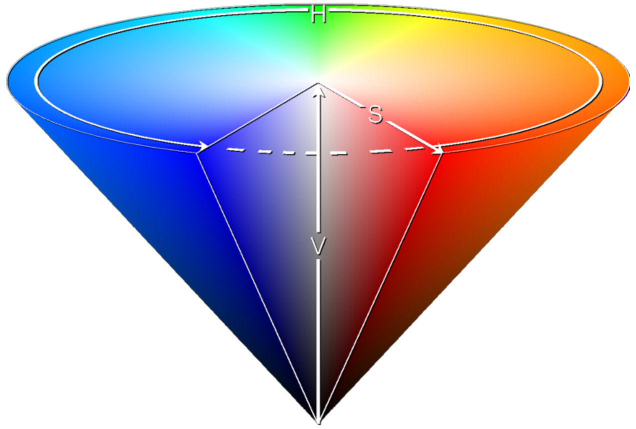
Hue: 0 Sat: 240 Lum: 120 Red: 255 Green: 0 Blue: 0



Hue: 8 Sat: 207 Lum: 124 Red: 238 Green: 68 Blue: 26

Sensors: Color Tracking

- HSV color representations are more intuitive for measuring closeness:
  - Hue
    - The “color type” (such as red, blue, or yellow):
    - Ranges from 0-360 (but normalized to 0-100% in some applications)
  - Saturation
    - The "vibrancy" or “purity”
    - Ranges from 0-100%
  - Value
    - The “brightness”
    - Ranges from 0-100%



Sensors: Color Tracking

- Method, for each pixel:
  - Convert to HSV color space
  - Determine if it is close to color being tracked by passing threshold for each parameter.
- Example:
  - To track pure red, H must belong to range [350, 10], S must belong to range [80,100], and V must belong to range [80,100].
  - Once all pixels that don’t meet threshold are thrown out, the locations of the remaining pixels can used to determine the relative position of the object being tracked.

Sensors:
Color Tracking

- MRS Example:
  - Convoy Search and Track mission

Overhead Vision Systems