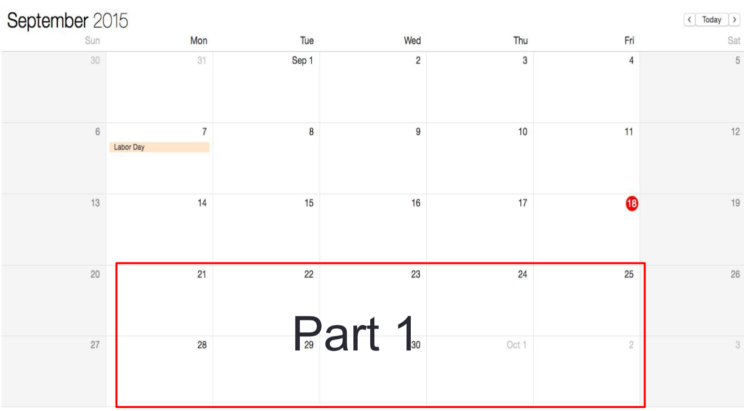
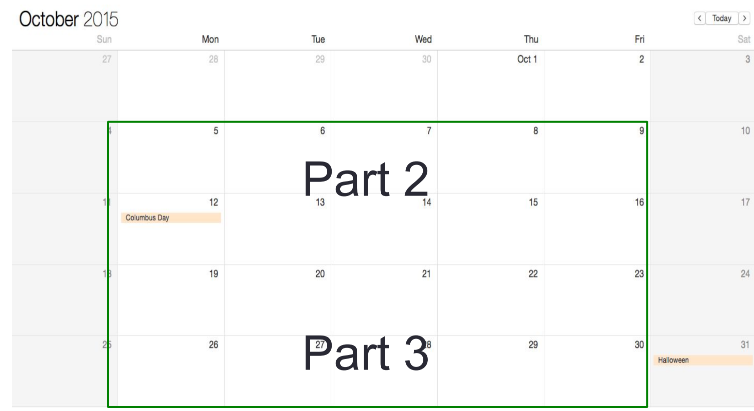
CS123 - Miscellaneous

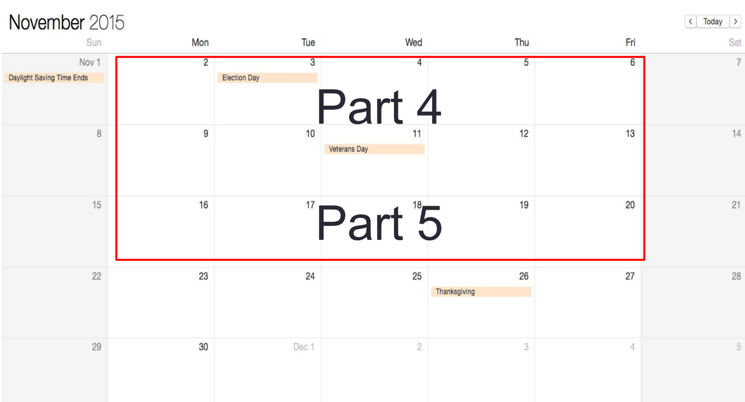
Programming Your Personal Robot

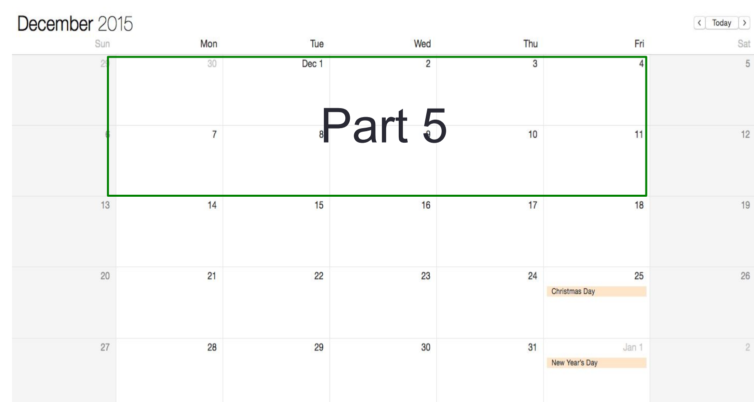
Kyong-Sok "KC" Chang, David Zhu
Fall 2015-16

Calendar









KC
Teaching

David
Teaching

Syllabus

- Part 1 - Communicating with robot (2 weeks)
  - BLE communication and robot API
- Part 2 - Event Driven Behavior (2 weeks)
  - Finite State Machine (Behavior Tree)
- Part 3 - Reasoning with Uncertainty (2 weeks)
  - Dealing with noisy data, uncertainty in sensing and control
- Part 4 - Extending the robot (1 weeks)
  - I/O extensions: digital, analog, servo, pwm, etc
- Part 5 – Putting it together (including UI/UX) (3 weeks)
  - Design and implement of final (group) project
  - Encourage you to go “above and beyond”

Logistics

- Getting new PSD Scanner
  - Update Hamster firmware
    - Over-The-Air Device Firmware Update (DFU)
    - nRF Toolbox App
  - Install the hardware
  - Sign-up sheet
- TA sessions (office hours): this week
  - Location: Gates B21 (Th: Huang basement)
  - Time: M:2~4pm, Tu:2~4pm, W:12:30-2:30pm, Th:2~4pm
- Lab reserved for CS123: this week
  - MTuW: 12~6pm @ Gates B21
- My office hours (KC)
  - Tues & Thurs: 1-2pm @ Gates B21(Tu), Huang Basement(Th)

Robotics Company: Toyota?

- Toyota Research Institute
  - Announced Nov. 6, 2015
  - new company in Silicon Valley
  - $1 billion: 5 Year
    - $50 million: Sept. 2015, committed to AI research in C
      labs in Stanford and MIT
  - 200 new employees
  - Start with the creation of laboratories near Stanford and MIT
  - CEO: Dr. Gill Pratt
    - a former MIT professor
    - a program manager (PM) at DARPA for 5 years
  - Focus on AI, robotics and self-driving technology
  - 2020 Olympic and Paralympic Games (in Tokyo)

Outline

- Logistics
- Future robots: Toyota AI?
- Recap Part 4: I/O extensions
  - ADC, PWM
  - Scanning: PSD Sensor, Servo motor
  - Low-pass filter: Accelerometer, Signal strength
- Part 5-1: Miscellaneous
  - Feedback control
    - Line-tracing: floor sensors (grey scale: 0~100)
  - Least Squares Method: line fitting from Scanning
  - Vision: OpenCV, Tracking, Demo by Kornel Niedziela (TA)
- Assignment #4
- Final project

Analog to Digital Converter (ADC)

- ADC
  - converts an analog voltage on a pin to a digital number
  - converting from the analog world to the digital world
  - to use electronics to interface to the analog world
- Relating ADC Value to Voltage
  - The ADC reports a *ratiometric value*

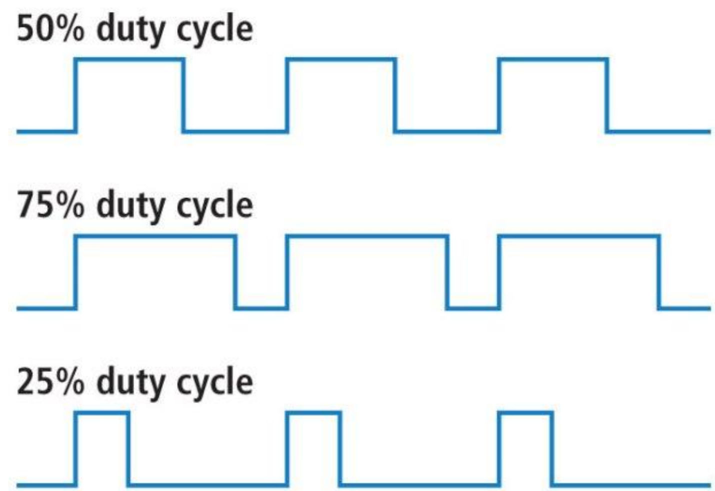
$$\frac{Resolution\ of\ the\ ADC}{System\ Voltage} = \frac{ADC\ Reading}{Analog\ Voltage\ Measured}$$

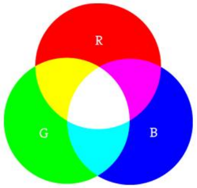
- Hamster
  - System Voltage = 3.7 V
  - Resolution of the ADC = 8 bit = 255 (= 0xFF)
  - Input (Analog): Voltage measured
  - Output (Digital): ADC Reading = Input * 255 / 3.7

Ref. learn.sparkfun.com

Pulse Width Modulation (PWM)

- Duty Cycle
  - on-time: when signal is high
  - duty cycle: amount of on-time
  - measured in % over a period
  - Ex) 5V
    - 50% duty cycle: 2.5V
- Examples
  - RGB LED
    - all equal duty cycle: white
  - Servo motors
    - frequency: 50 Hz waveform
    - duty cycle: 5~10%
    - 1.0 ms pulse: 0 deg
    - 1.5 ms pulse: 90 deg
    - 2.0 ms pulse: 180 deg





Ref. learn.sparkfun.com

I/O mode: Hamster

Sensors Service Packet format definition

	Details	Value from Robot	User converted value
0	**Version / Topology**	0 ~ 255	0 ~ 255
1	**Network ID**	0 ~ 255	0 ~ 255
2	**Command / Security**	0 ~ 255	0 ~ 255
3	**Signal Strength**	-128 ~ 0	-128 ~ 0 dBm
4	**Left Proximity**	0 ~ 255	0 ~ 255
5	**Right Proximity**	0 ~ 255	0 ~ 255
6	**Left Floor**	0 ~ 255	0 ~ 255
7	**Right Floor**	0 ~ 255	0 ~ 255
8	**Acc X High**	-32768 ~ 32767	-32768 ~ 32767
9	**Acc X Low**		
10	**Acc Y High**	-32768 ~ 32767	-32768 ~ 32767
11	**Acc Y Low**		
12	**Acc Z High**	-32768 ~ 32767	-32768 ~ 32767
13	**Acc Z Low**		
14	**Flag**		
15	**Light High or Temperature**	0 ~ 65535 -128 ~ 127	0 ~ 65535 Lux -40 ~ 88 ºC
16	**Light Low or Battery**	0 ~ 255	0 ~ 100 %
17	**Input A**	0~255	0 ~ 255 (0 ~ 3.3 V)
18	**Input B**		
19	**Line Tracer State**	0 ~ 255	0 ~ 255

Effector Service Packet format Definition

	Data	Value to Robot	User input value
0	**Version / Topology**	0 ~ 255	0 ~ 255
1	**Network ID**	0 ~ 255	0 ~ 255
2	**Command / Security**	0 ~ 255	0 ~ 255
3	**Left Wheel**	-100 ~ +100 (+fwd, -bwd)	-100 ~ 100 %
4	**Right Wheel**		
5	**Left LED**	0 (off) ~ 7	0 (off) ~ 7
6	**Right LED**		
7	**Buzzer High**	0(off) 1 ~ 16777215	0(off) 1.00 Hz ~ 167.77215 KHz,
8	**Buzzer Middle**		
9	**Buzzer Low**		
10	**Musical Note**	1~88(piano key) 0(off)	1~88 0(off)
11	**Line Tracer Mode/Speed**	0x11 ~ 0x6A 0x0?(off)	0x11 ~ 0x6A 0x0?(off)
12	**Proximity IR Current**	0 ~ 7 (default 2)	0 ~ 7 (default 2)
13	**G-Range, Bandwidth**	0 ~ 3 (default 0), 0 ~ 8 (default 3)	0 ~ 3 (default 0), 0 ~ 8 (default 3)
14	**IO Mode(A, B)**	0 ~ 127	0 ~ 127
15	**Output A**	0 ~ 255	0 ~ 255
16	**Output B**		
17	**Wheel Balance**	-128 ~ 127	-128 ~ 127
18	**Input Pull**	0~16	
19			

Ref. Kre8 Technology, Inc.

ADC: Hamster

Sensors Service Packet format definition

	Details	Value from Robot	User converted value
0	**Version / Topology**	0 ~ 255	0 ~ 255
1	**Network ID**	0 ~ 255	0 ~ 255
2	**Command / Security**	0 ~ 255	0 ~ 255
3	**Signal Strength**	-128 ~ 0	-128 ~ 0 dBm
4	**Left Proximity**	0 ~ 255	0 ~ 255
5	**Right Proximity**	0 ~ 255	0 ~ 255
6	**Left Floor**	0 ~ 255	0 ~ 255
7	**Right Floor**	0 ~ 255	0 ~ 255
8	**Acc X High**	-32768 ~ 32767	-32768 ~ 32767
9	**Acc X Low**		
10	**Acc Y High**	-32768 ~ 32767	-32768 ~ 32767
11	**Acc Y Low**		
12	**Acc Z High**	-32768 ~ 32767	-32768 ~ 32767
13	**Acc Z Low**		
14	**Flag**		
15	**Light High or Temperature**	0 ~ 65535 -128 ~ 127	0 ~ 65535 Lux -40 ~ 88 ºC
16	**Light Low or Battery**	0 ~ 255	0 ~ 100 %
17	**Input A**	0~255	0 ~ 255 (0 ~ 3.3 V)
18	**Input B**		
19	**Line Tracer State**	0 ~ 255	0 ~ 255

Sensor packet: 17th and 18th bytes: Input

Ref.12) Input A/B

ADC mode) Analog to Digital Converter mode (Measuring analog voltage)

Active only if the Effectors' IO Mode value == 0

Formula) Volt = 3.3 * ADC level / 255 (volt)

DI mode) Digital Input mode (Measuring digital input)

Active only if the Effectors' IO Mode value == 1

Formula) 1 if input voltage >= 0.5, 0 otherwise

Effector packet: 14th byte: External IO Mode

Port A and Port B are independent of each other.

bit	7	6	5	4	3	2	1	0
	Port A (0~127)				Port B (0~127)			
	ADC mode, 0x0				ADC mode, 0x0			

DC (Analog-to-Digital)

0x01 DI (Digital Input)

0x08 SERVO (Analog Servo Control)

0x09 PWM (Digital-to-Analog)

0x0A DO (Digital Output)

Ref. Kre8 Technology, Inc.

PWM: Hamster

Effector packet: 14th byte: External IO Mode

Port A and Port B are independent of each other.

bit	7	6	5	4	3	2	1	0
	Port A (0~127)				Port B (0~127)			
	ADC mode, 0x0				ADC mode, 0x0			

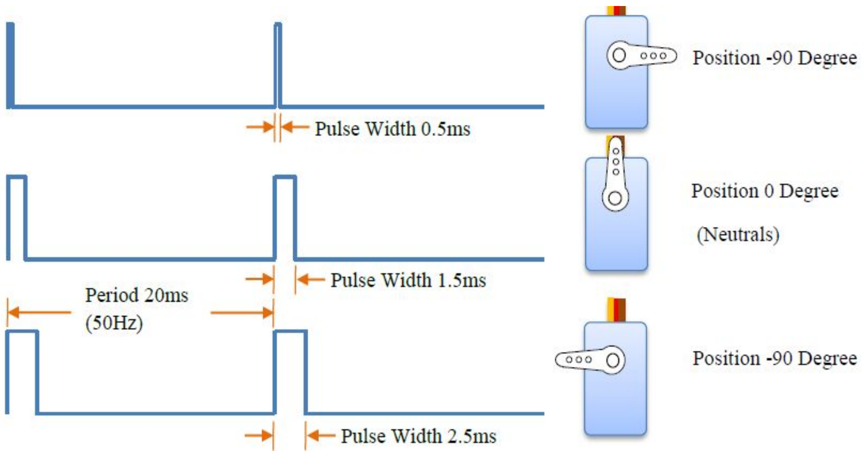
0x00 ADC (Analog-to-Digital)
0x01 DI (Digital Input)
0x08 SERVO (Analog Servo Control)
0x09 PWM (Digital-to-Analog)
0x0A DO (Digital Output)

ADC (Analog-to-Digital) Mode: 0x00
Measures input voltage with 8-bit ADC.
Max input voltage is ~3.7volt → 255(0xFF)

DI (Digital Input) Mode: 0x01
Detect input voltage to either 0 or 1.
1 if input voltage > 3.7/2 (~1.8 volt)
0 otherwise

SERVO (Analog servo) Output Mode: 0x08
Generating PWM signal(mode = 8) for external Servo control
* If value == 0(off) → no pulse
* If value > 180, pulse width limits to 2.5 ms

Port A	1 ~ 180	0(off), 90(center)	1deg=1.0ms, 90deg=1.5ms, 180deg=2.0ms
Port B	1 ~ 180	0(off), 90(center)	1deg=1.0ms, 90deg=1.5ms, 180deg=2.0ms



PWM (Digital-to-Analog) Output Mode: 0x09
Output: PWM signal's Duty value
If value > 100(0x64), output is 1 and PWM pulse period is 20 msec.
Therefore, if Duty value is 50%(50, 0x32), output is 0 for 10 msec, then output is 1 for the next 10msec.

DO (Digital Output) Mode: 0x0A
If value is not 0, output is 'high'.

Ref. Kre8 Technology, Inc.

PSD IR Sensor

Distance Measuring Sensor Unit : cheap ~$8

-- 4~30 cm, 5V

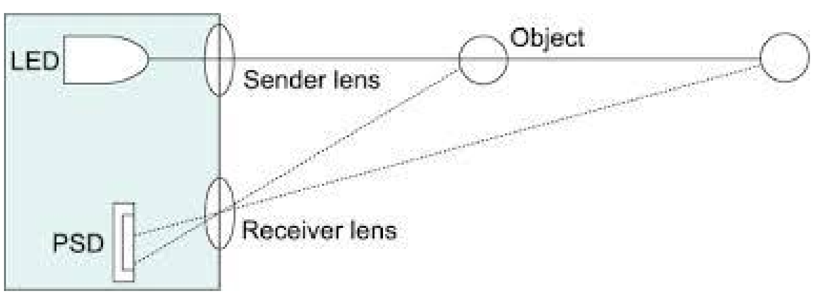
-- Receiver: PSD (Position sensitive detector)

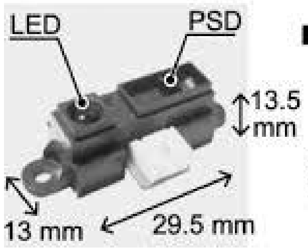
-- Transmitter: IR-LED

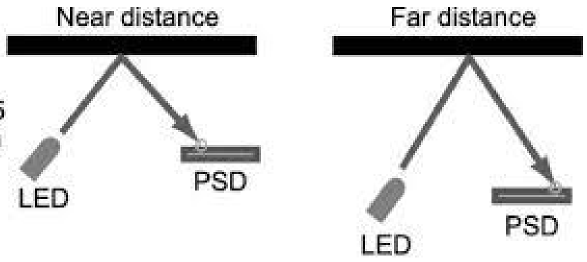
Sharp GP2Y0A41SK0F Analog Distance Sensor

Distance sensors comparison

Technical Specification and Manual







Servo Motor

TowerPro SG90 9G Mini Servo: cheap ~$1

-- 0~180 deg

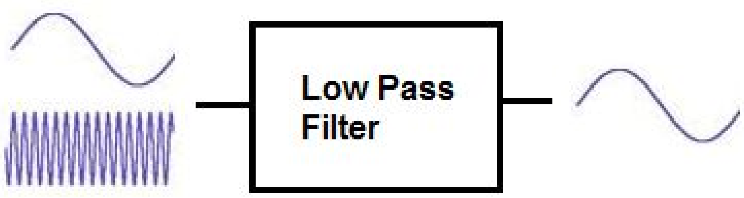
-- Stall Torque: 1.5kg/cm at 4.8V

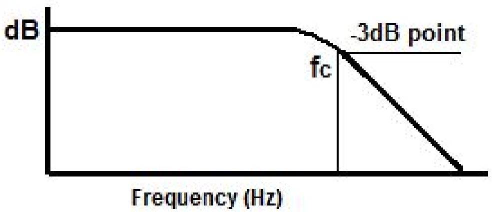
-- Voltage : 3.0 - 7.2V

Low-pass filter

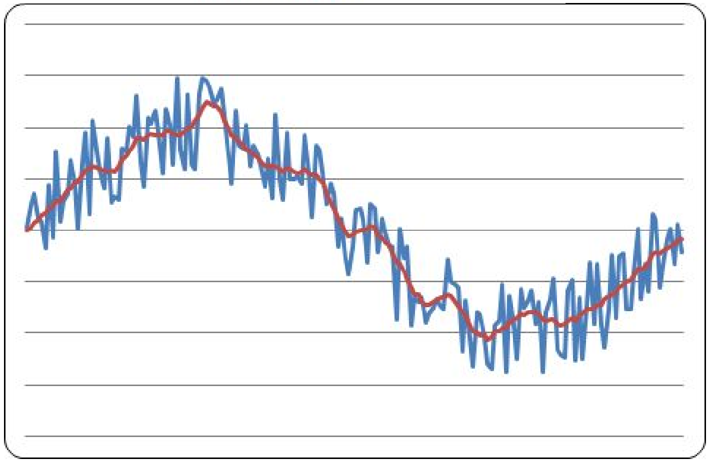
- Low-pass filter
  - passes signals with a frequency lower than a certain cutoff frequency
  - attenuates signals with frequencies higher than the cutoff frequency
  - a smoother form of a signal
  - removing the short-term fluctuations
  - leaving the longer-term trend

Ref. www.learningaboutelectronics.com



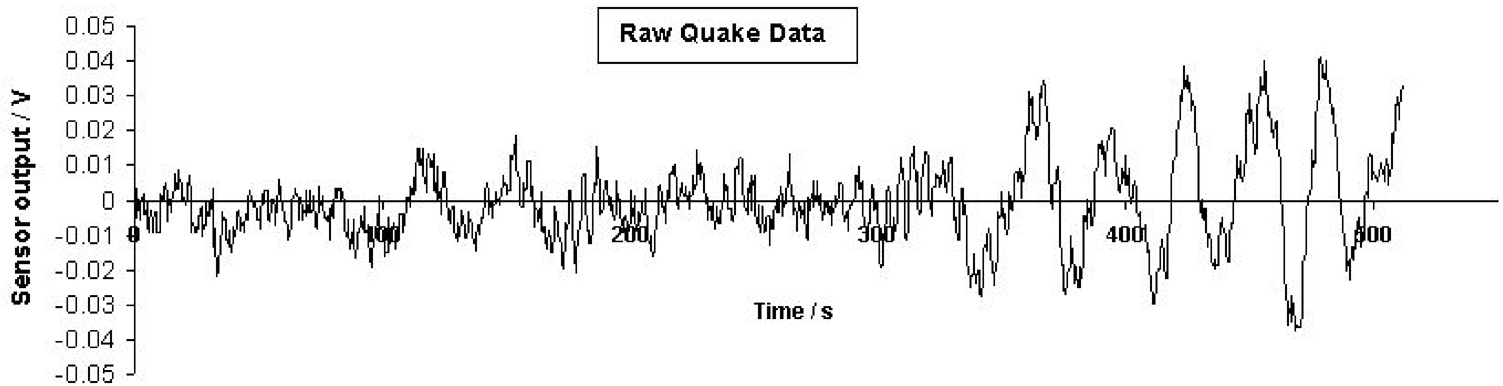


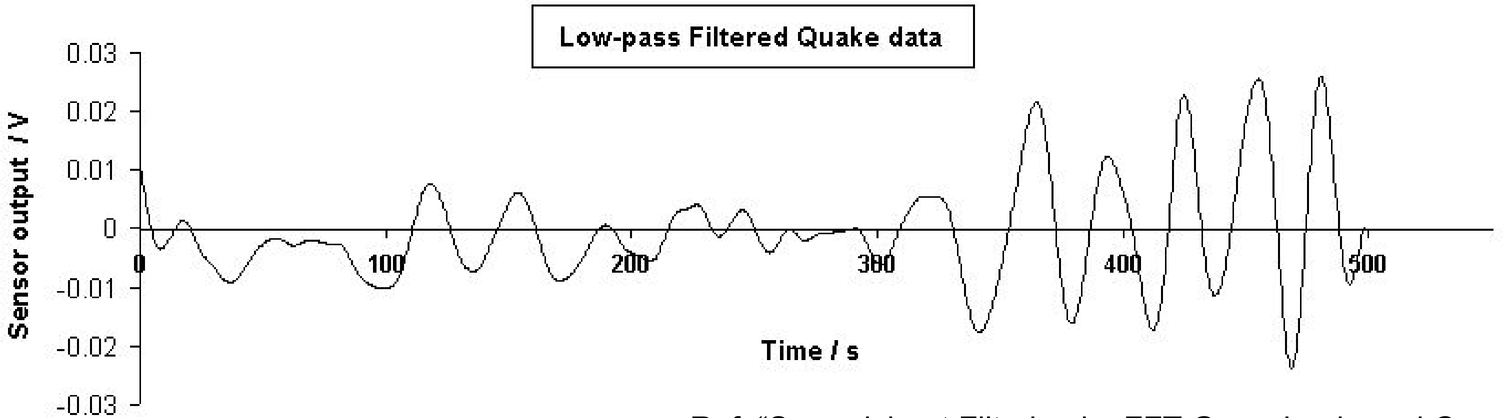
- Example
  - digital filters for smoothing sets of data
  - acoustic barriers
  - blurring of images
  - moving average operation:
    - used in finance field
    - can be analyzed with the same signal processing techniques as are used for other low-pass filters



Ref. Wikipedia

Low-pass filter





Ref. “Spreadsheet Filtering by FFT Gaussian-based Convolution”
by Randall D. Peters, Mercer University

Low-pass filter: example

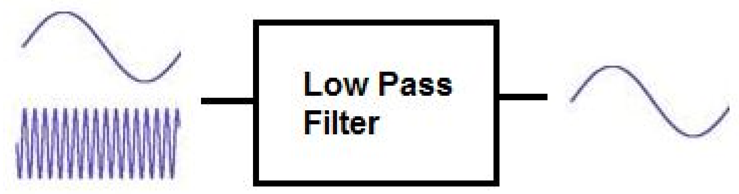
Simple infinite impulse response filter
-- First-order discrete-time realization
-- digital filter

```
// Return: RC low-pass filter output samples y
// Given: input samples x, time interval dt,
//          and time constant RC
 function lowpass(real[0..n] x, real dt, real RC)
   var real[0..n] y
   var real α := dt / (RC + dt)
   y[0] := x[0]
   for i from 1 to n
       y[i] := α * x[i] + (1-α) * y[i-1]
   return y
```

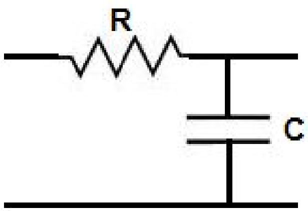
Note:

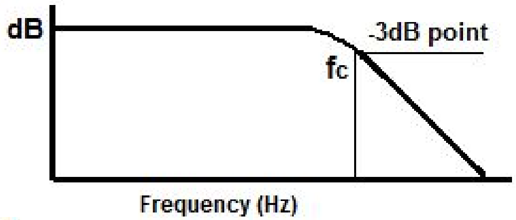
- α: *smoothing factor*
- False-positive vs. False-negative

Low Pass Filter Calculator



RC Low Pass Filter





$$f_c = \frac{1}{2\pi RC}$$

Ref. Wikipedia

Ref. www.learningaboutelectronics.com

Feedback control: Line-tracing

- Threshold Method
  - frequency
  - default_speed
  - more tweak required

```
def threshold(self, left, right):
    diff = 10
    speed_l = self._default_vel
    speed_r = self._default_vel
    if (self._line_loc == 0):
        if (left > self._threshold):
            speed_l = speed_l + diff
        else:
            speed_l = speed_l - diff
        if (right > self._threshold):
            speed_r = speed_r + diff
        else:
            speed_r = speed_r - diff
    return (speed_l, speed_r)
```

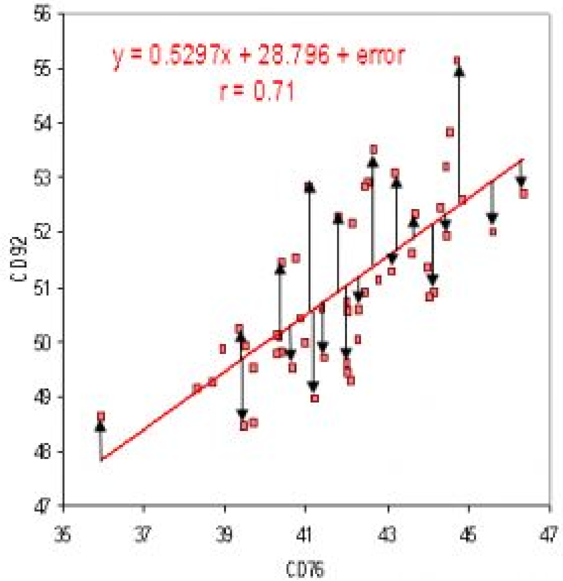
- Floor sensors: IR
  - Grey scale (0~100:white)
- Feedback Control
  - frequency
  - default_speed
  - p_gain: error multiplier
  - more robust
  -

```
def p_control(self, left, right):
    speed_l = self._default_vel
    speed_r = self._default_vel
    if (self._line_loc == 0):
        speed_l = self._default_vel
                + self._kp * (left - right)
        speed_r = self._default_vel
                - self._kp * (left - right)
    return (speed_l, speed_r)
```

- Demo

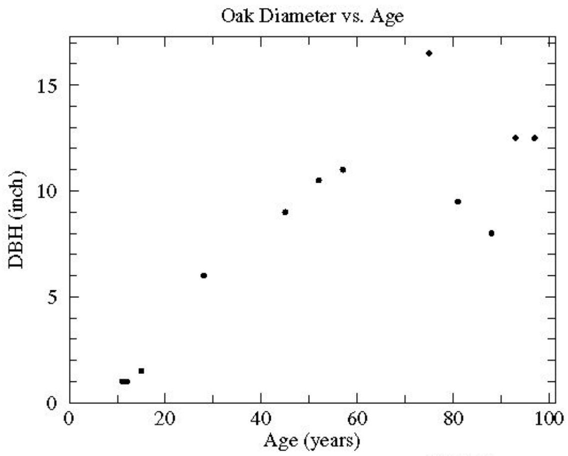
Line fitting: Least Squares Method

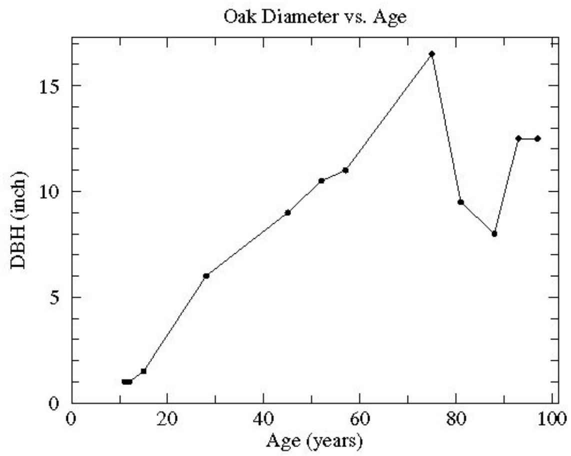
- Simple linear regression
- For a given set of points (x_i, y_i), find the regression line, $y = mx + b$, such that the sum of squared residuals of the data points is minimized.
    - residual: vertical distance of data point to regression line $y = mx + b$
    - each data point has one residual
    - residual > 0 (< 0), if data point is above (below) the regression line
    - residual = 0, if data point is on the regression line

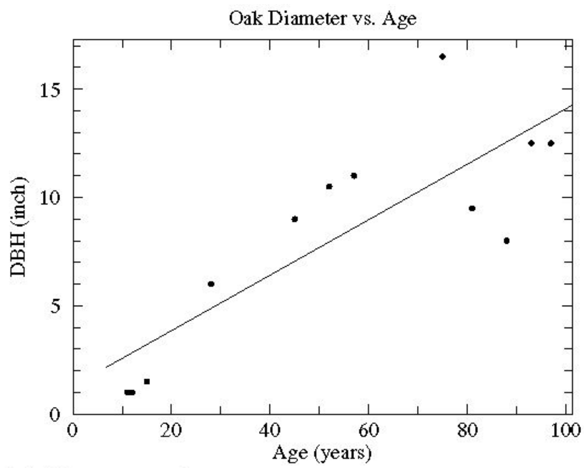


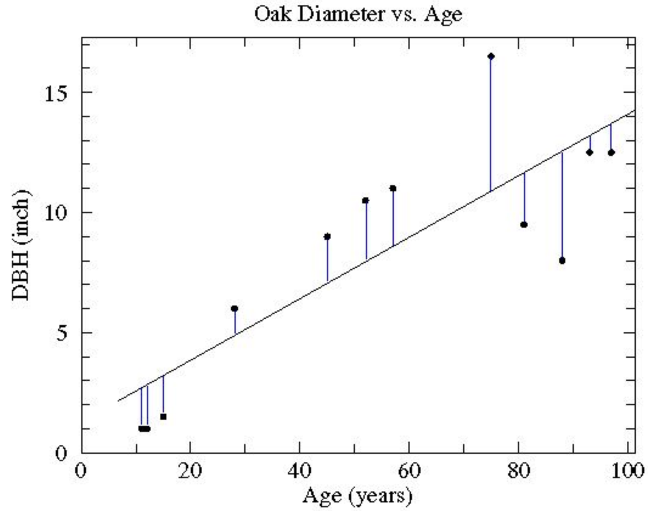
Ref. www.statisticshowto.com

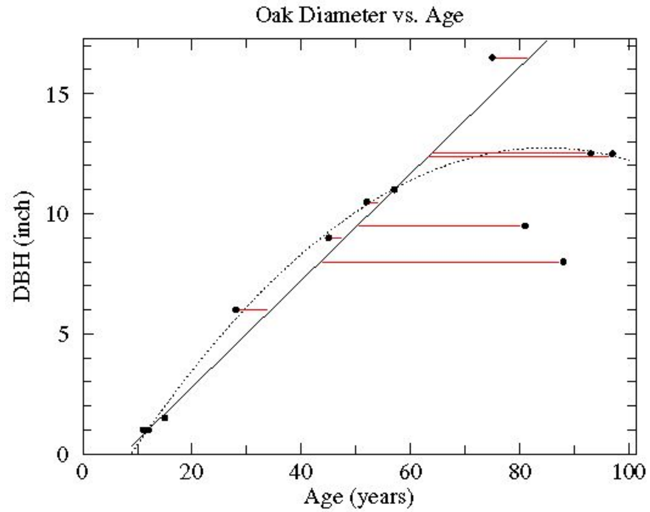
Line fitting: Least Squares Method











Ref. http://www.physics.csbsju.edu/stats/least_squares.html

Line fitting: Least Squares Method

Step 1: Calculate the mean of the *x*-values and the mean of the *y*-values.

$$\overline{X} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad \overline{Y} = \frac{\sum_{i=1}^{n} y_i}{n}$$

Step 2: The following formula gives the slope of the line of best fit:

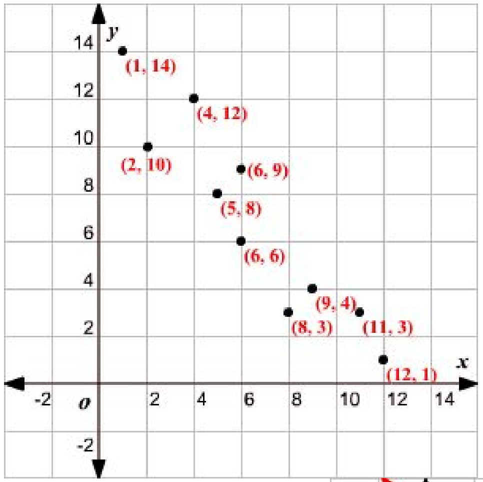
$$m = \frac{\sum_{i=1}^{n}\left(x_i - \overline{X}\right)\left(y_i - \overline{Y}\right)}{\sum_{i=1}^{n}\left(x_i - \overline{X}\right)^2}$$

Step 3: Compute the *y*-intercept of the line by using the formula:

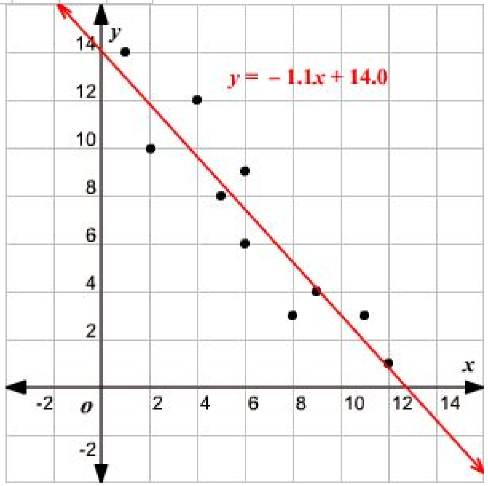
$$b = \overline{Y} - m\overline{X}$$

Step 4: Use the slope *m* and the *y*-intercept *b* to form the equation of the line.

$$y = mx + b$$



$y = -1.1\,x + 14$



Ref. HotMath.com

Line fitting: Least Squares Method

```
import scipy.optimize as optimization
import numpy
….
def get_line(self, x, y):
        n = len(x)
        m, b = self.least_square_fit(x, y, 0, n)
        print m, b

def line_func(self, x, A, B):
        return A*x + B

def least_square_fit(self, x, y, i, f):
         xdata = numpy.array(x[i:f])
         ydata = numpy.array(y[i:f])
         popt, pcov = optimization.curve_fit( self.
   line_func, xdata, ydata)
          return popt
```

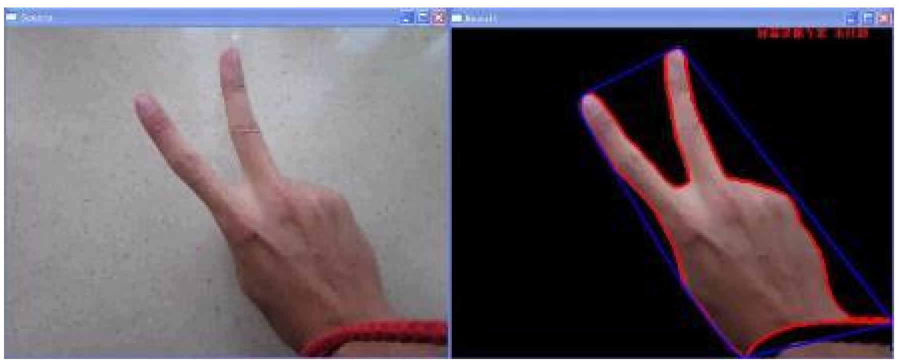
- Python Modules
  - http://www.scipy.org/
  - SciPy library:
    Fundamental library for scientific computing
  - NumPy package:
    Base N-dimensional array package
- Demo

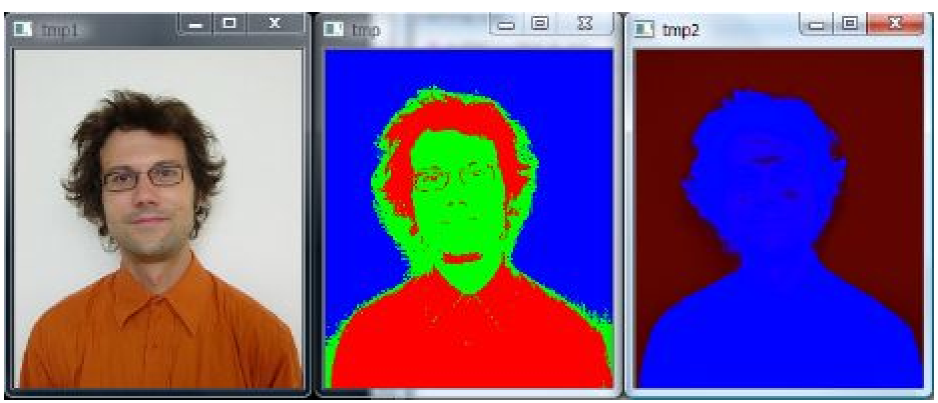
Vision: Tracking

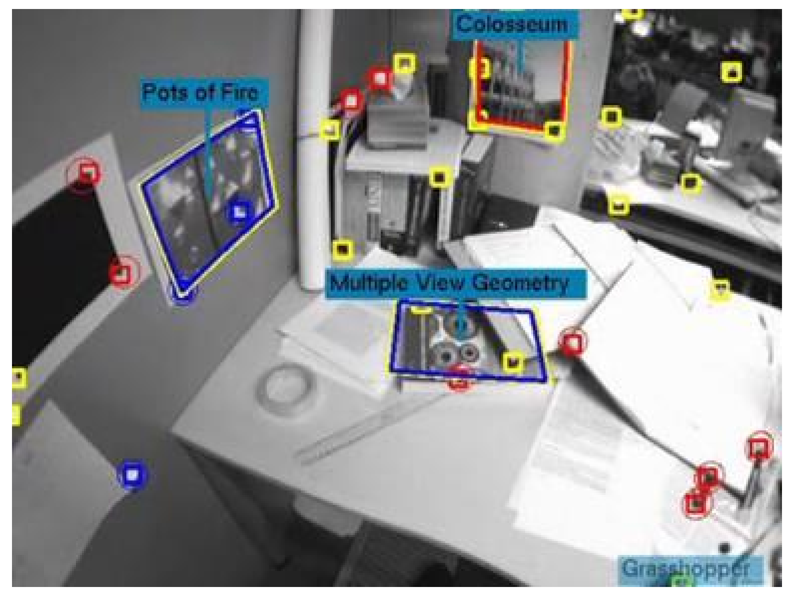
- Prepared and presented by Kornel Niedziela (TA)
- OpenCV
- Tracking
- Demo

OpenCV

- Open source computer vision library
- Provides easy image processing functionality
- Compiled in C++, linkable in python





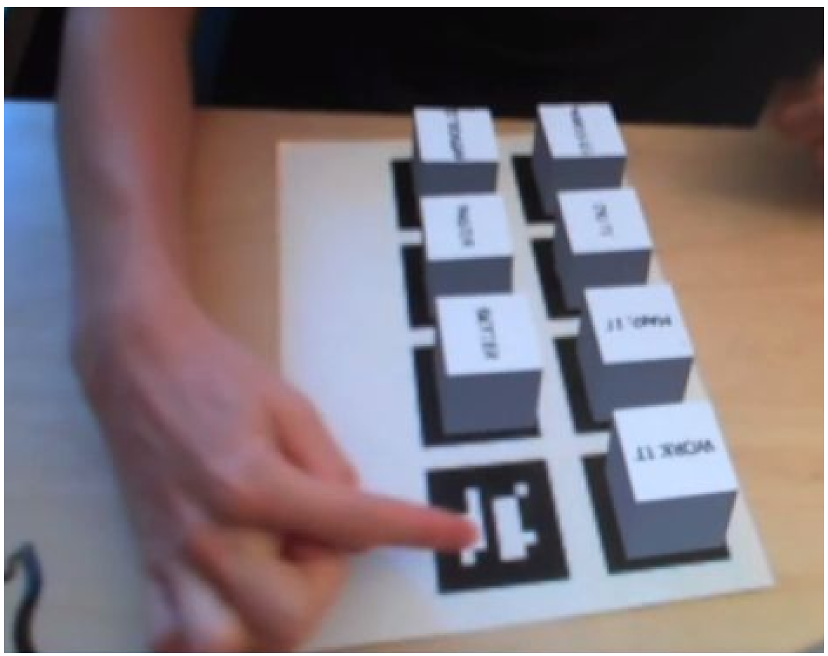


OpenCV

- Core functionality
- Image processing
- Image file reading and writing
- Media I/O
- High-level GUI
- Video Analysis
- Camera Calibration and 3D Reconstruction
- 2D Features Framework
- Object Detection
- Machine Learning
- Clustering and Search in Multi-Dimensional Spaces
- Computational Photography
- Images stitching
- Hardware Acceleration Layer
- Shape Distance and Matching
- Super Resolution
- Video Stabilization
- 3D Visualizer
- ArUco Marker Detection
- Improved Background-Foreground Segmentation Methods
- Biologically inspired vision models and derivated tools
- Custom Calibration Pattern for 3D reconstruction
- GUI for Interactive Visual Debugging of Computer Vision
- Framework for working with different datasets
- Deep Neural Network module
- Deformable Part-based Models
- Face Recognition
- Binary descriptors for lines extracted from an image
- MATLAB Bridge
- Optical Flow Algorithms
- Image Registration
- RGB-Depth Processing
- Saliency API
- Stereo Correspondance Algorithms
- Structured Light API
- Surface Matching
- Scene Text Detection and Recognition
- Tracking API
- Extra 2D Features Framework
- Extended Image Processing
- Extended object detection
- Additional photo processing algorithms
- Core_va_intel

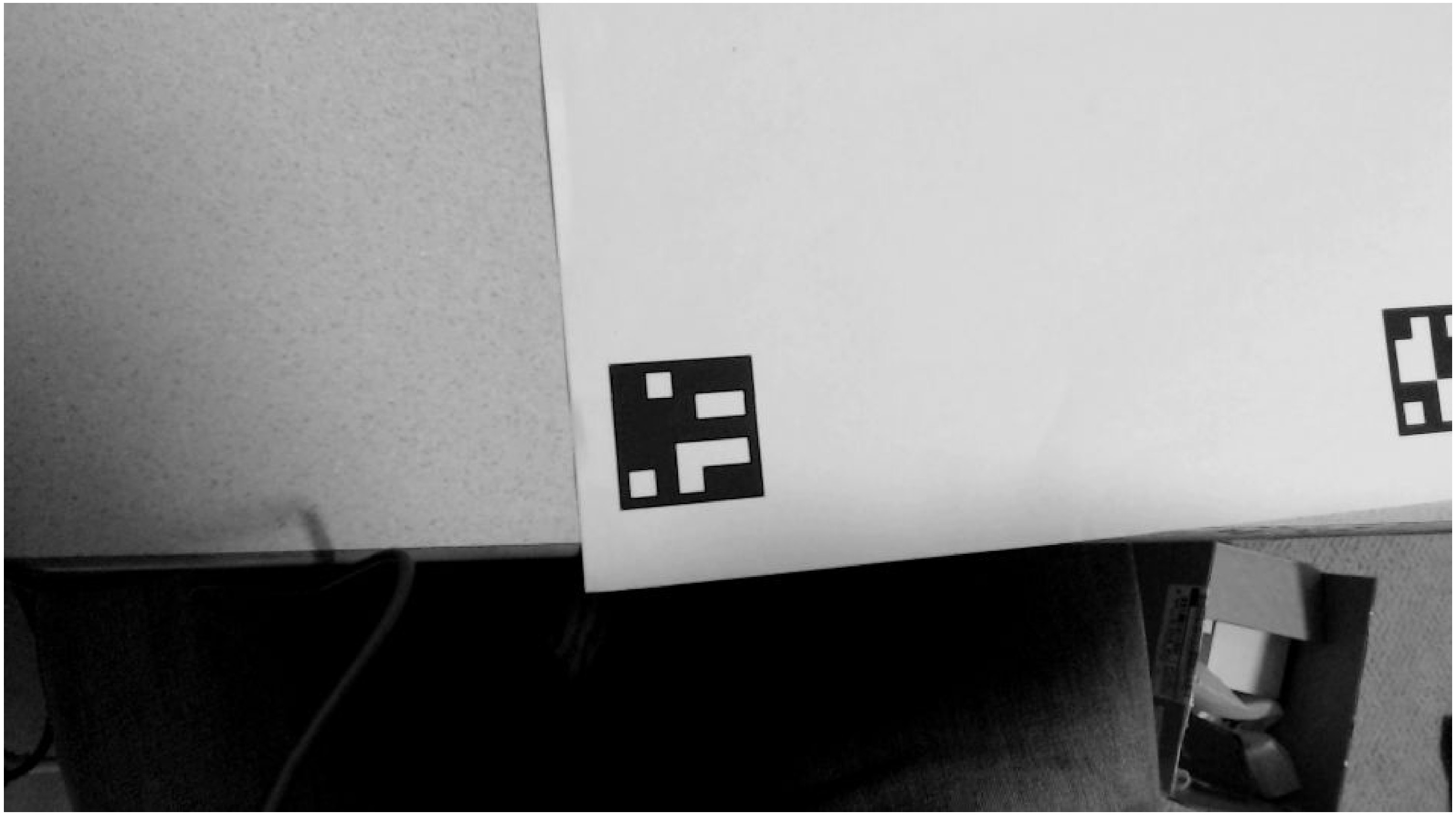
python-ar-markers

- 3rd party library to detect Augmented Reality markers
- Used for AR, but useful to track anything globally



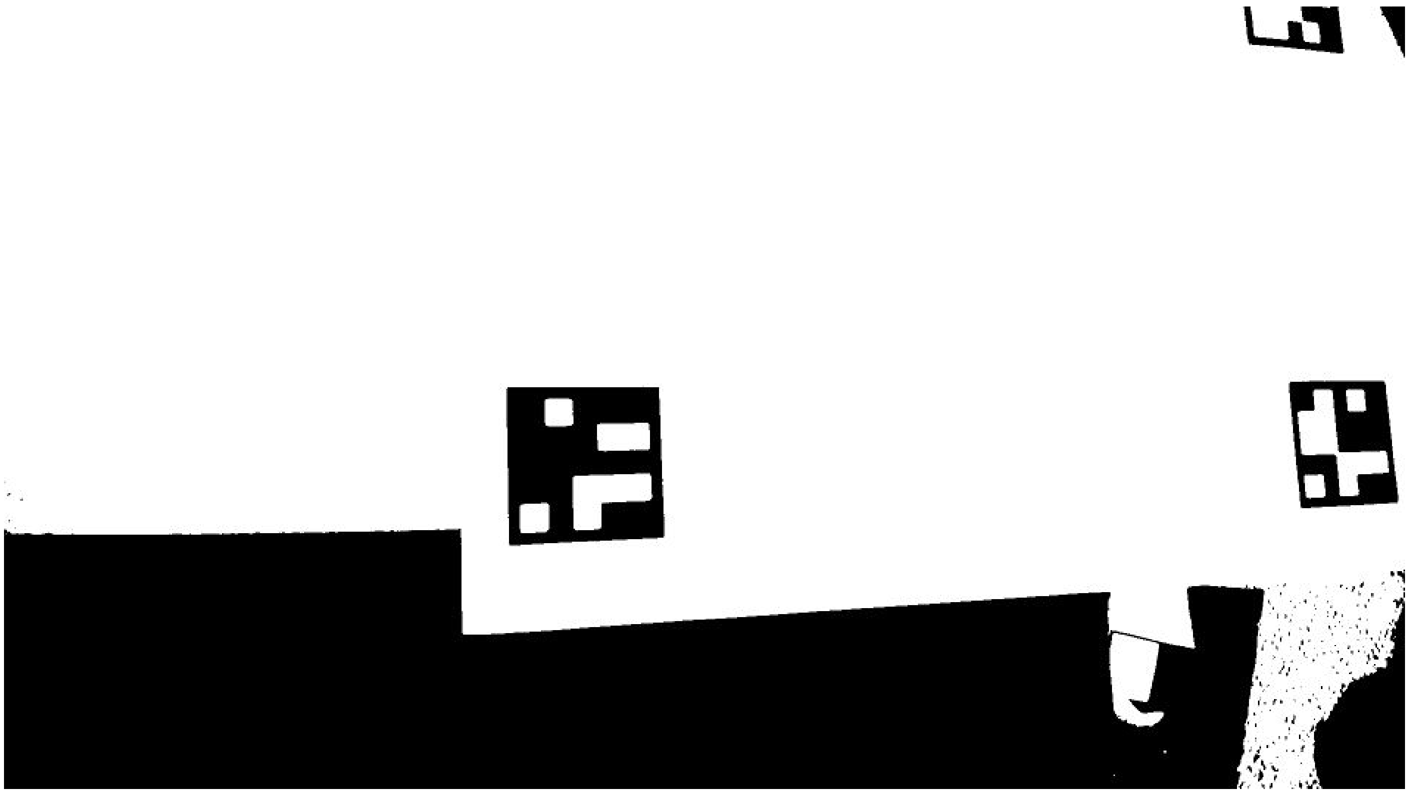
python-ar-markers

- Start with grayscale image



python-ar-markers

- Apply thresholding

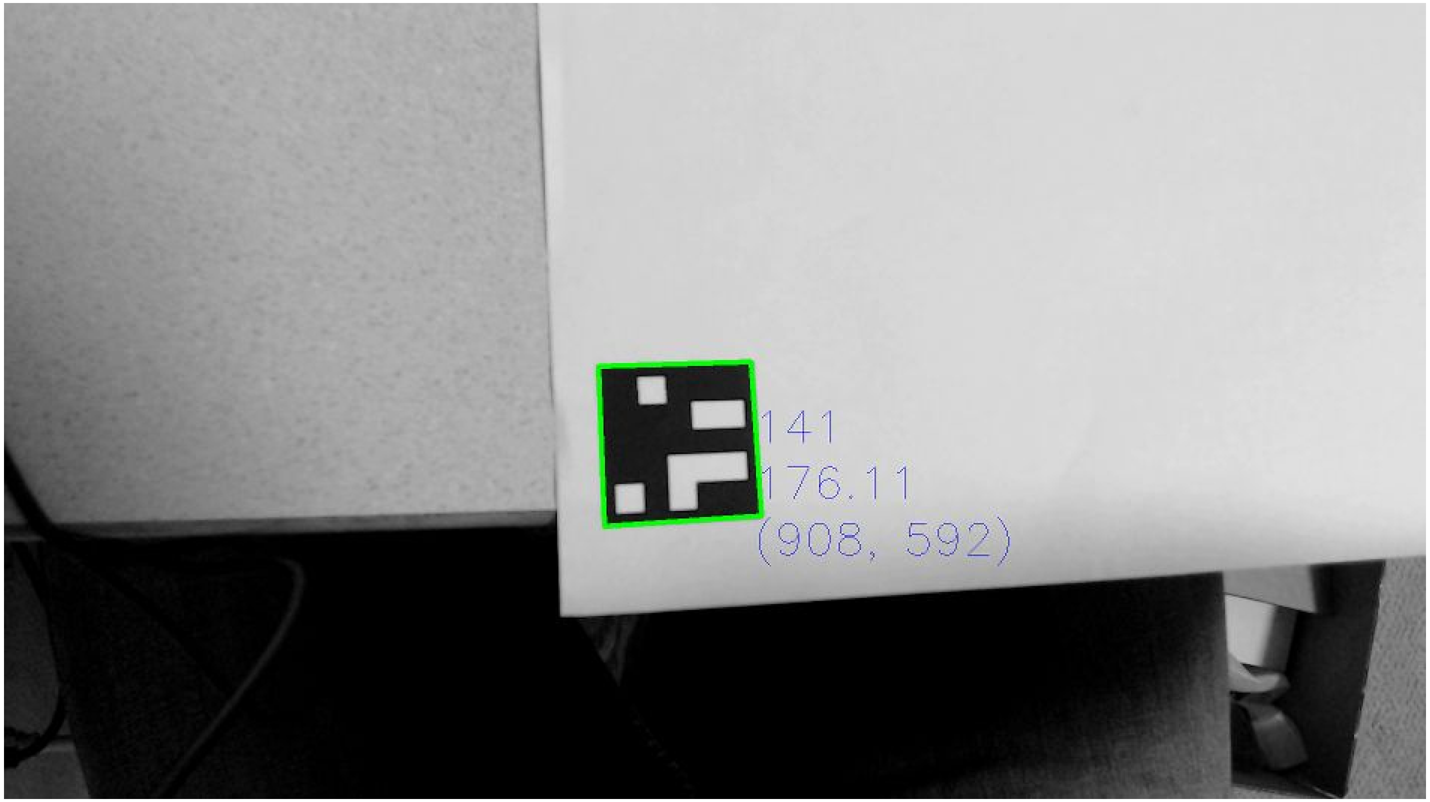


python-ar-markers

- Use OpenCV findContours

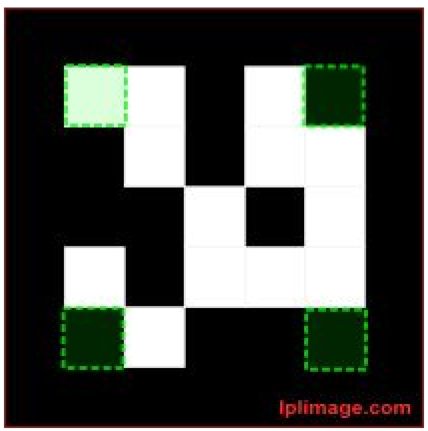
python-ar-markers

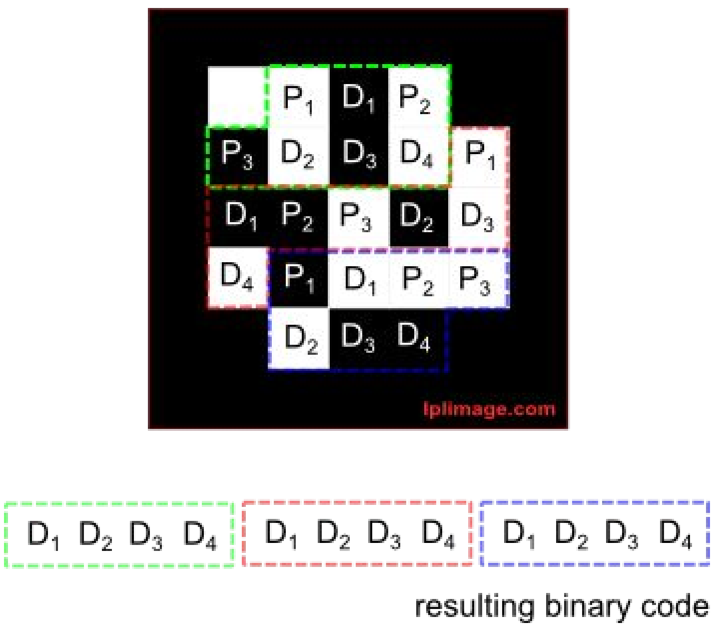
- Fit polygon to contours (approxPoly2D)
  - Keep ones with 4 sides only
  - Calculate center of square for x/y, and angle of sides for rotation



python-ar-markers

- Read id value from marker
  - Read 4 corners to determine orientation, rotate if needed
  - Apply hamming code to determine marker id





http://iplimage.com/blog/approach-encodedecode-black-white-marker/

Limitations

- Sensitive to lighting conditions
  - Must be uniform lighting and well lit
- Needs clear surface
- Precision is limited by camera resolution
- Not practical for all situations

Assignment#4

1. "Drop-off" problem: global localization with landmarks

   -- known map (known set of obstacles) + unknown position

   Solution) Make the scanning sensor work and model the sensor values.

   -- PSD IR Sensor + Servo motor

2. Collision detection problem

   -- false-positive vs. false-negative

   Solution) Apply low-pass filter and compare to the raw data.

   -- Accelerometer

3. UI/UX problem

   Solution) Model the control input and the sensor data graphically.

Assignment#4: Global Localization and Collision Detection

Final Project

- Robot programming
  - Mobile robot
  - Navigation
    - modeling
    - localization
    - planning
    - execution
    - UI/UX
  - Team of 2 people
    - Multiple robots

- CS 123 Final Project Proposal Guidelines

Reference and Reading

- "Line of Best Fit (Least Square Method)" by HotMath.com
- "Residual Values in Regression Analysis" by statisticshowto.com
- "Simple linear regression" by Wikipedia