

Introduction to Robotics

Jeff McGough

Computer Science and Engineering
SDSMT

November 5, 2021

Motion

Locomotion

The word locomotion means "The act of moving from place to place." from Latin combining location and motion.

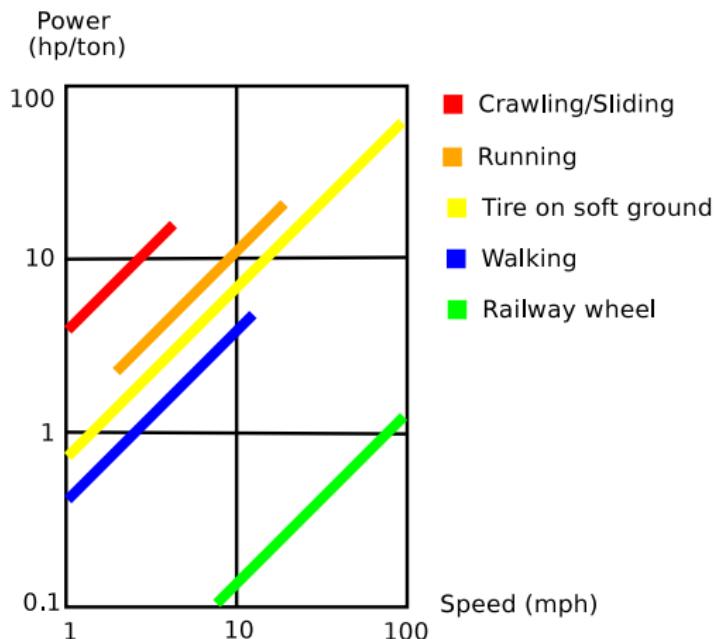
Biology has explored a variety of very interesting ways to move around.

Animals can be carried on currents, swim, crawl, slide, walk, run, jump, and fly.



Locomotion

The relations between energy, speed and motion type.



Locomotion

- ▶ Motion is found in nature - can be difficult to imitate all the types of motion in nature
- ▶ Human systems use wheels or tracks
- ▶ Rolling is very efficient
Nature does not use wheels
- ▶ Human movement can be modeled as a rolling polygon



Walk or Roll

- ▶ Number of actuators
- ▶ Structural complexity
- ▶ Control issues
- ▶ Energy expense
- ▶ Internal mass movement



Mobility Issues

Stability

- ▶ Number of contact points
- ▶ Center of gravity
- ▶ Static/Dynamic stabilization
- ▶ Terrain

Contact

- ▶ Contact area
- ▶ Angle of contact
- ▶ Friction

Environment

- ▶ Structure
- ▶ Medium



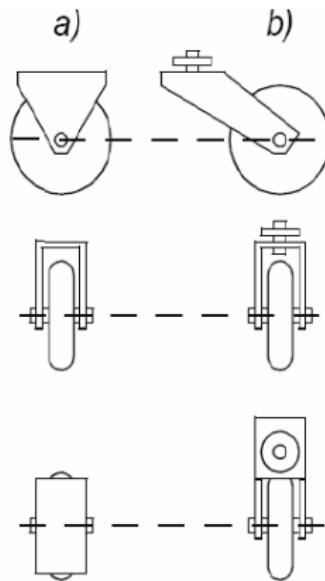
Wheels

- ▶ Wheels work well in many applications
- ▶ Three wheels give stability
- ▶ More than three require suspension
- ▶ Wheel type depends on application
- ▶ Steering may be required



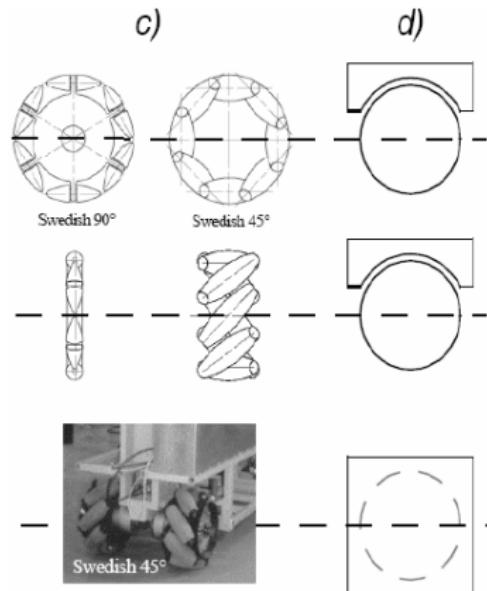
Wheel Types

- ▶ Standard Wheel (a): Two degrees of freedom. 1. Rotation about the axle. 2. Rotation about the contact point.
- ▶ Castor Wheel (b): Three degrees of freedom. 1. Rotation about the axle. 2. Rotation about the contact point. 3. Rotation about the castor axle.



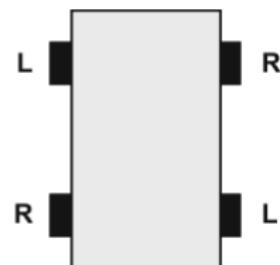
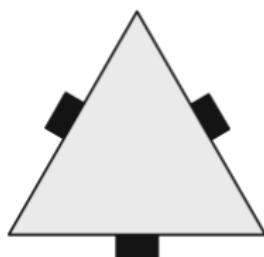
Wheel Types

- ▶ Swedish Wheel (c): Three degrees of freedom. 1. Rotation about the axle. 2. Rotation about the rollers. 3. Rotation about the contact point.
- ▶ Ball or Spherical Wheel (d): Suspension technically not solved. If implemented with Swedish wheels, one may obtain three degrees of freedom.



Omnidirectionality

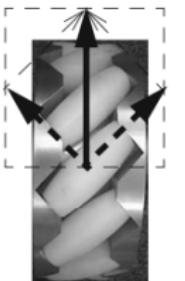
Recall the Swedish Wheel:



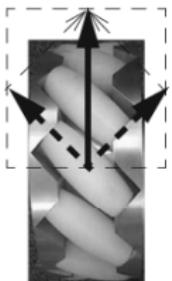
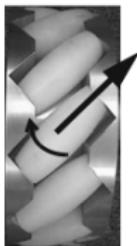
Omnidirectionality



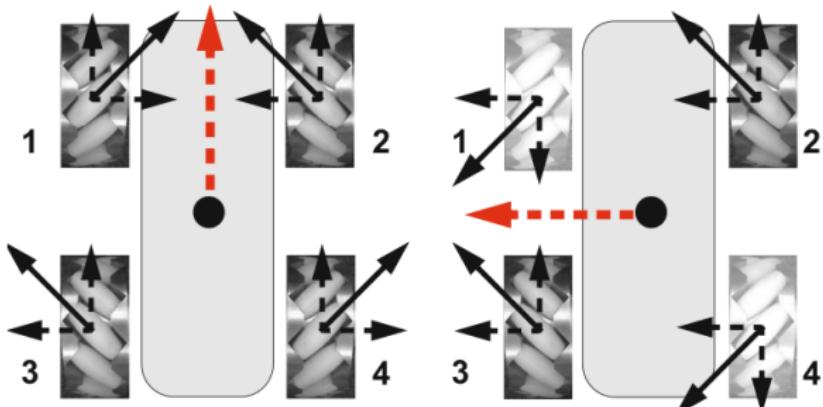
Swedish Wheel Driving



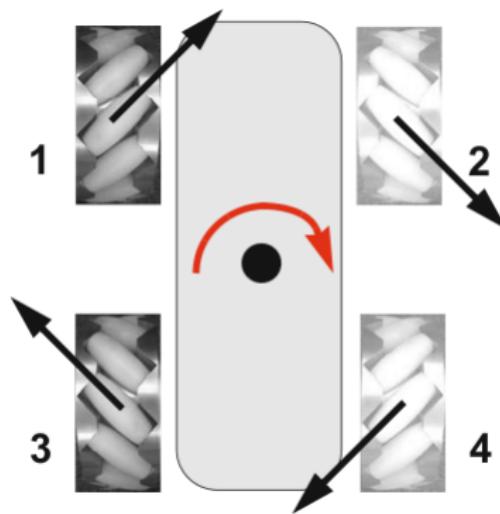
left-hand wheel
seen from below



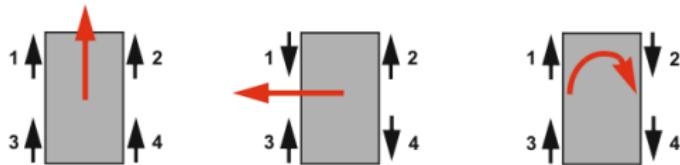
right-hand wheel
seen from below



Swedish Wheel Turning

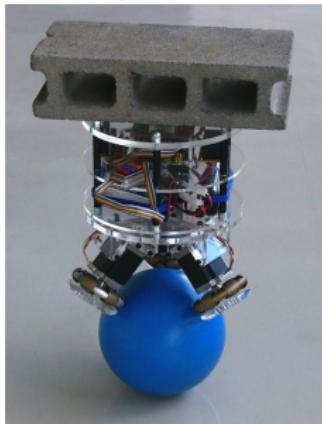
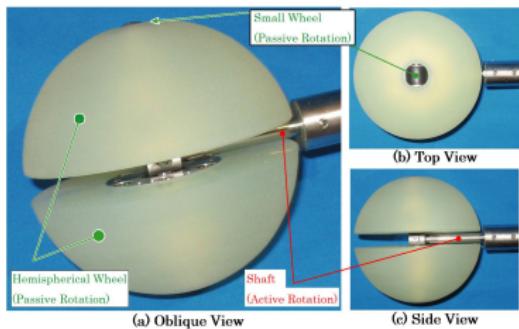


Swedish Wheel Summary



- ▶ Driving forward: all four wheels forward
- ▶ Driving backward: all four wheels backward
- ▶ Driving left: 1,4 backwards; 2,3 forward
- ▶ Driving right: 1,4 forward; 2,3 backward
- ▶ Turning clockwise: 1,3 forward; 2,4 backward
- ▶ Turning counterclockwise: 1,3 backward; 2,4 forward

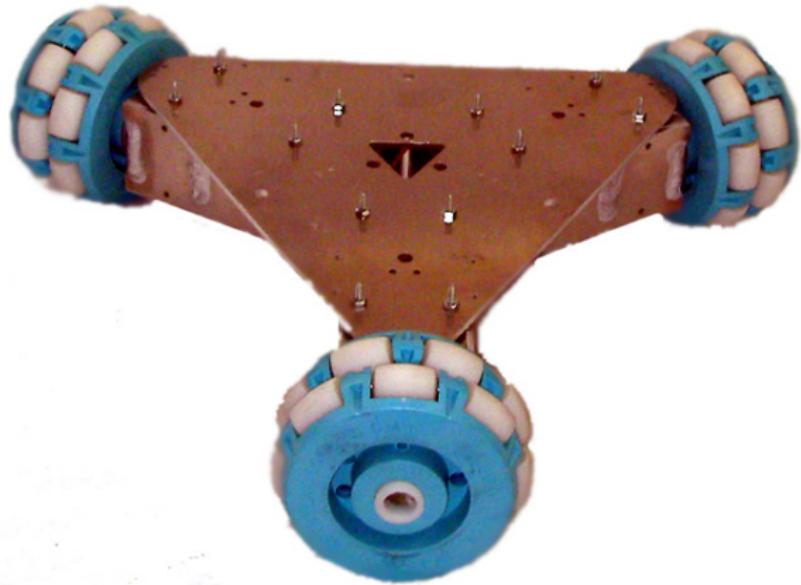
Other Types



Other Types



Example



<http://www.youtube.com/watch?v=CjcyHicm3NA>
<http://www.youtube.com/watch?v=5vJCucpVdX0>

Characteristics

- ▶ Stability of a vehicle is guaranteed with three wheels: center of gravity is within the triangle formed by the ground contact points of the wheels.
- ▶ Stability may be improved by 4 or more wheels: however, this requires a suspension system. More complicated but more robust.
- ▶ Bigger wheels: allow movement over larger obstacles (lower angle of incidence) but require

higher torque or higher gear ratios.



- ▶ Combining drive and steering on one wheel increases complexity and reduces the accuracy of odometry.

Variations on Treads

<http://www.youtube.com/watch?v=L3NGCL-efRM>

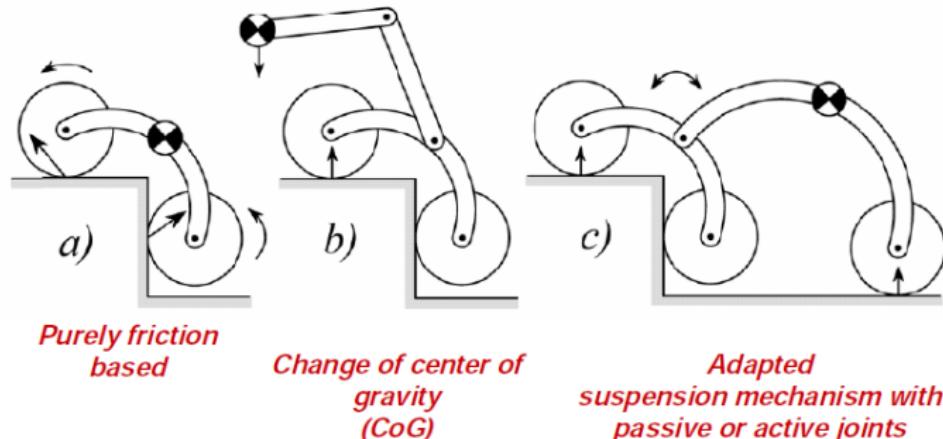
<https://www.youtube.com/watch?v=BTp2UAaihaI>

Walking Wheels

- ▶ Passive locomotion concept
- ▶ Six wheels
 - ▶ two non-steered wheels on each side
 - ▶ rear steered wheel (single)
 - ▶ front steered wheel with suspension (single)
- ▶ length 60 cm
- ▶ height 20 cm
- ▶ Characteristics
 - ▶ very stable on rough terrain
 - ▶ can overcome obstacles 2x wheel diameter



Adaptive Suspension



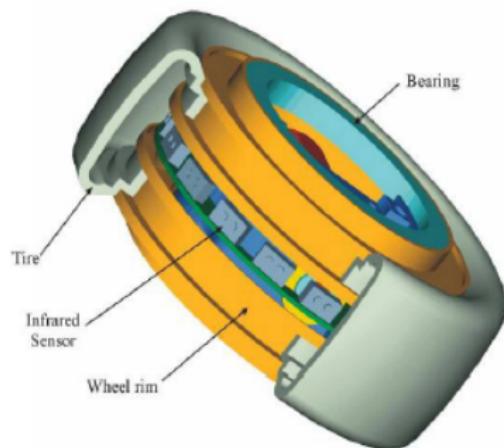
Purely friction
based

Change of center of
gravity (CoG)

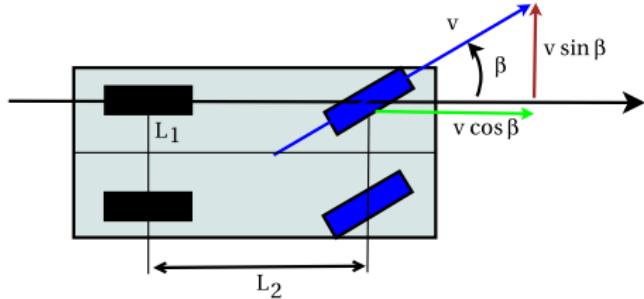
Adapted
suspension mechanism with
passive or active joints

Flexible Wheel

- ▶ Better grip in loose ground
- ▶ Measure the contact force and points
- ▶ Active locomotion concept
 - ▶ Eight motorized wheels
 - ▶ Integrate sensor data with control
- ▶ Sensors
 - ▶ Inclinometer
 - ▶ Joint sensors
 - ▶ Tactile wheels

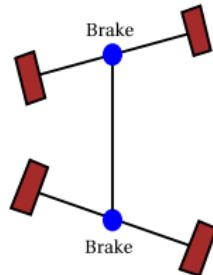
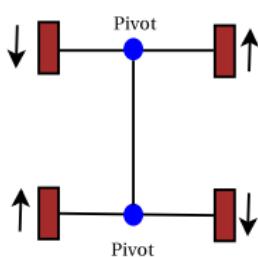


Drive Systems

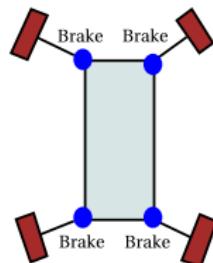
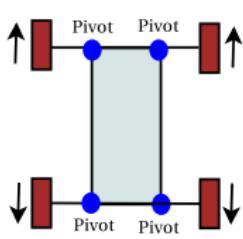


Front Wheel Steered System.

Drive Systems

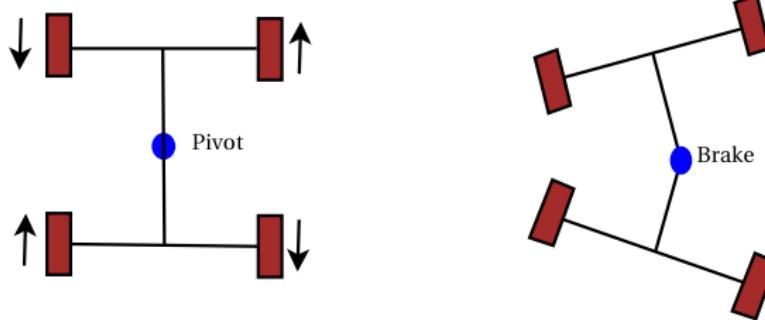


Dual Differential Drive (DDD).

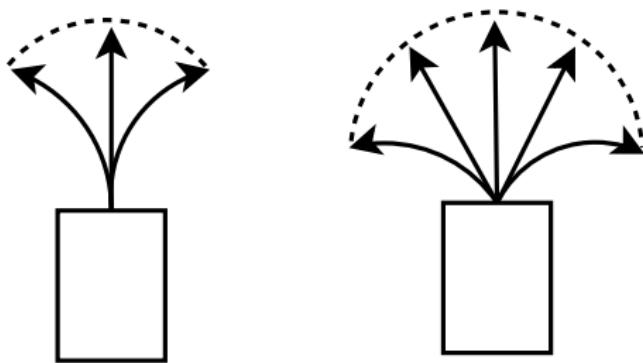


Four Wheel Steer (FWS).

Articulated Systems



Reach



Legs

We move over to a more biological approach. What appears trivial in the natural world is not so easy for robotics. We will examine a few systems based on articulated arms.



Recall Mobility Issues - Holds for Walking

Stability

- ▶ Number of contact points
- ▶ Center of gravity
- ▶ Static/Dynamic stabilization
- ▶ Terrain

Contact

- ▶ Contact area
- ▶ Angle of contact
- ▶ Friction

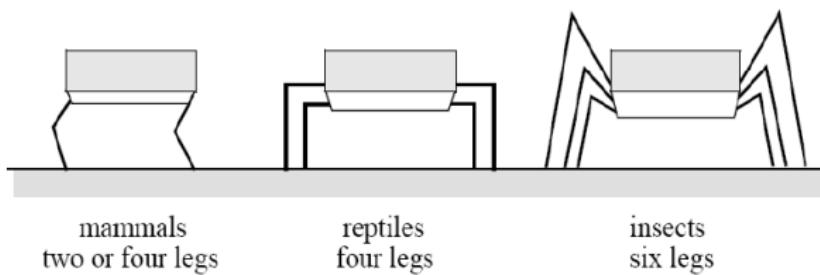
Environment

- ▶ Structure
- ▶ Medium



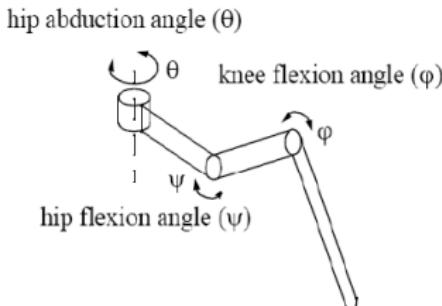
Legs - More details

- ▶ Number of legs
 - ▶ Three legs for static stability
 - ▶ More legs - more coordination
 - ▶ Fewer legs - more complicated motion
- ▶ Legs must be lifted
 - ▶ Possible loss of stability
 - ▶ Shifting center of gravity
 - ▶ Additional energy cost
 - ▶ Complexity of positioning system
 - ▶ Articulator path planning
- ▶ Static stability and walking requires 6 legs.



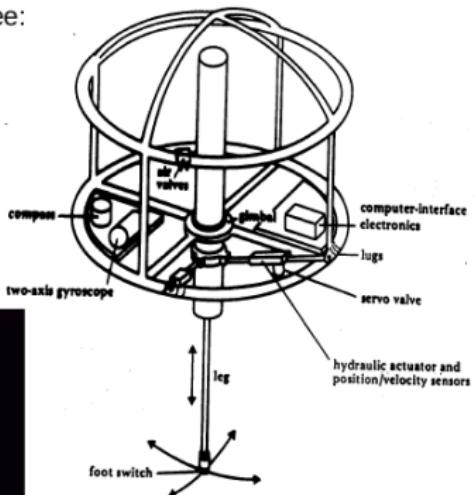
Leg Joints

- ▶ Two DOF is required:
lift and swing
- ▶ Three DOF is needed in most cases:
lift, swing and position
- ▶ Fourth DOF is needed for stability:
ankle joint - improves balance and walking



Examples

- No industrial applications up to date, but a popular research field
- For an excellent overview please see:
<http://www.uwe.ac.uk/clawar/>



The Hopping Machine at MIT

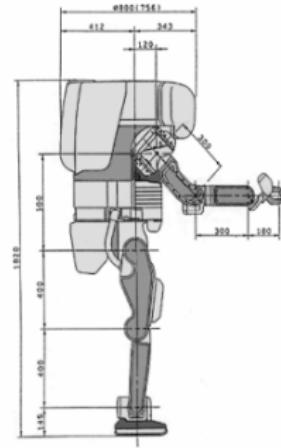
Early Example - Honda

- P2 from Honda, Japan

- Maximum Speed: 2 km/h
- Autonomy: 15 min
- Weight: 210 kg
- Height: 1.82 m
- Leg DOF: 2x6
- Arm DOF: 2x7



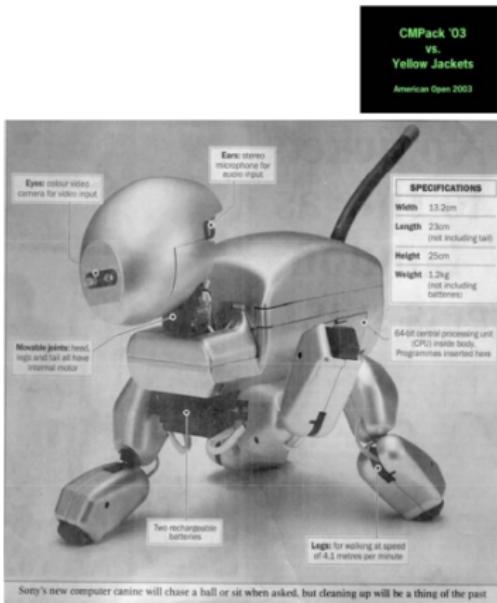
Honda P2



© R. Siegwart, ETH Zurich - ASL

Example - Sony

- Artificial Dog Aibo from Sony, Japan



Sony's new computer canine will chase a ball or sit when asked, but cleaning up will be a thing of the past

© R. Siegwart, ETH Zurich - ASL

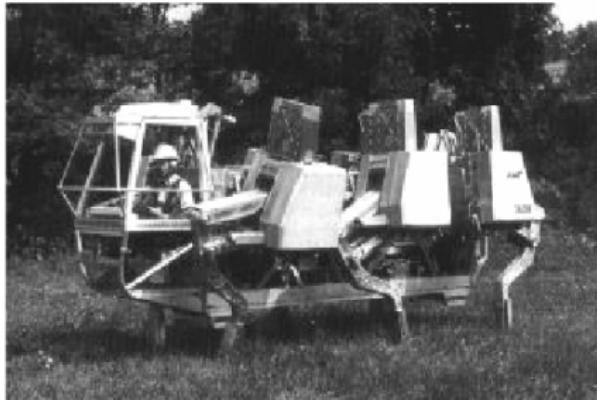
More Current Example can be found with

Boston Dynamics



Examples

- Most popular because static stable walking possible
- The human guided hexapod of Ohio State University
 - Maximum Speed: 2.3 m/s
 - Weight: 3.2 t
 - Height: 3 m
 - Length: 5.2 m
 - No. of legs: 6
 - DOF in total: 6*3



Examples



- Lauron II,
University of Karlsruhe
 - Maximum Speed: 0.5 m/s
 - Weight: 6 kg
 - Height: 0.3 m
 - Length: 0.7 m
 - No. of legs: 6
 - DOF in total: 6*3
 - Power Consumption: 10 W

Bipeds

Consider a humanoid robot - what are the issues

- ▶ Need 5 DOF per leg
- ▶ Two legs
- ▶ Need 5 DOF per arm
- ▶ Two arms
- ▶ Camera pan and tilt - 2 DOF
- ▶ Gaits ($k = 2$ or 4 ?)

This adds to 22 DOF to control and control over numerous gaits.

Robot must also balance itself...

- ▶ Static balance
- ▶ Dynamic balance

Dynamic Balance

Balancing robots - inverted pendulum problem.



- ▶ Gyroscopes
- ▶ Accelerometers
- ▶ Compass
- ▶ IMUs
- ▶ Camera

Dynamic Balance

Balancing robots - inverted pendulum problem.

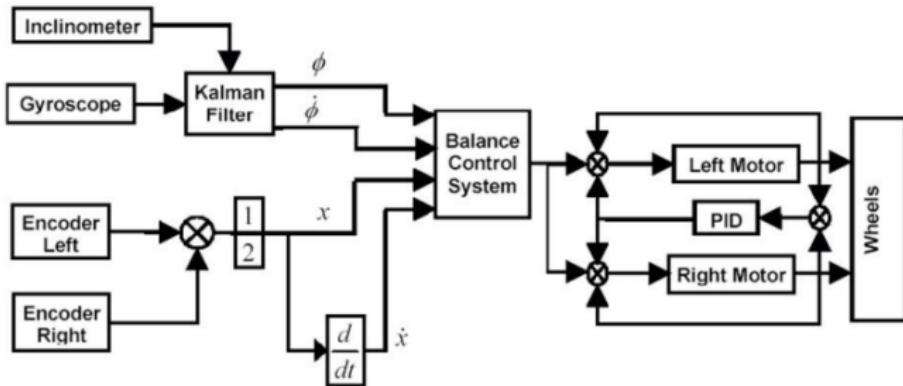
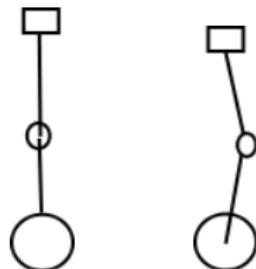


Figure 10.5: Kalman-based control system [Ooi 2003]

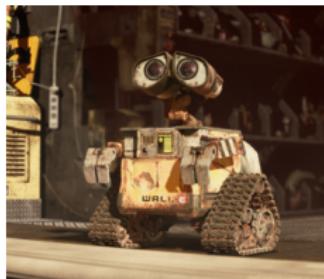
Dynamic Balance for walking

Double inverted pendulum problem.

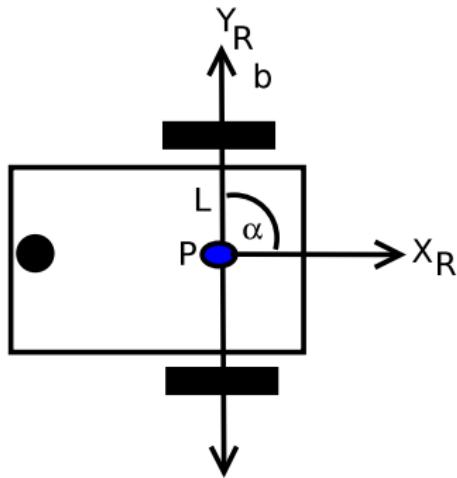


Classic problems in controls. You will see more in the ECE controls courses.

Motion Modeling



Drive Systems



The differential drive robot dimensions and variables.

DD Model

Recall the FK

$$\dot{x} = \frac{r}{2}(\dot{\phi}_1 + \dot{\phi}_2) \cos(\theta)$$

$$\dot{y} = \frac{r}{2}(\dot{\phi}_1 + \dot{\phi}_2) \sin(\theta)$$

$$\dot{\theta} = \frac{r}{2L}(\dot{\phi}_1 - \dot{\phi}_2)$$

DD Model

And the IK

$$\dot{\phi}_1 = \frac{L}{r} \left(\frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2} \right) \pm \frac{\sqrt{\dot{x}^2 + \dot{y}^2}}{r}$$

$$\dot{\phi}_2 = -\frac{L}{r} \left(\frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{\dot{x}^2 + \dot{y}^2} \right) \pm \frac{\sqrt{\dot{x}^2 + \dot{y}^2}}{r}$$

Direction along the path is selected depending on the \pm .

Mecanum Model

- ▶ r - wheel radius.
- ▶ L_1 - distance between left and right wheel pairs, L_2 - distance between front and rear wheel pairs.
- ▶ v_x, v_y, ω - the robot velocity and angular velocity in robot coordinates.
- ▶ $\dot{x}, \dot{y}, \dot{\theta}$ - robot velocity in x, y and robot angular velocity in global coordinates.
- ▶ $\dot{\phi}_{FL}, \dot{\phi}_{FR}, \dot{\phi}_{BL}, \dot{\phi}_{BR}$ - front left, front right, back left, back right, radians per minute.

Mecanum Model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = rR(\theta) \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{2(L_1+L_2)} & \frac{1}{2(L_1+L_2)} & -\frac{1}{2(L_1+L_2)} & \frac{1}{2(L_1+L_2)} \end{bmatrix} \begin{bmatrix} \dot{\phi}_{FL} \\ \dot{\phi}_{FR} \\ \dot{\phi}_{BL} \\ \dot{\phi}_{BR} \end{bmatrix}$$

Mecanum Model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \frac{r}{4} \begin{bmatrix} A \cos(\theta_k) - B \sin(\theta_k) \\ A \sin(\theta_k) + B \cos(\theta_k) \\ \frac{2}{(L_1+L_2)} C \end{bmatrix}$$

where

$$A = (\omega_{FL,k} + \omega_{FR,k} + \omega_{BL,k} + \omega_{BR,k}),$$

$$B = (-\omega_{FL,k} + \omega_{FR,k} + \omega_{BL,k} - \omega_{BR,k}),$$

$$C = (-\omega_{FL,k} + \omega_{FR,k} - \omega_{BL,k} + \omega_{BR,k})$$

Swedish Wheel Kinematics

Inverse kinematics:

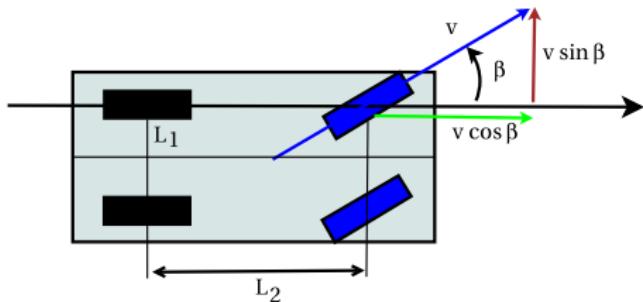
$$\begin{bmatrix} \dot{\phi}_{FL} \\ \dot{\phi}_{FR} \\ \dot{\phi}_{BL} \\ \dot{\phi}_{BR} \end{bmatrix} = \frac{1}{r} R^T(\theta) \begin{bmatrix} 1 & -1 & -(L_1 + L_2)/2 \\ 1 & 1 & (L_1 + L_2)/2 \\ 1 & 1 & -(L_1 + L_2)/2 \\ 1 & -1 & (L_1 + L_2)/2 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}$$

Swedish Wheel Kinematics

Inverse kinematics:

$$= \frac{1}{r} \begin{bmatrix} \cos(\theta)\dot{x} + \sin(\theta)\dot{y} + \sin(\theta)\dot{x} - \cos(\theta)\dot{y} - (L_1 + L_2)\dot{\theta}/2 \\ \cos(\theta)\dot{x} + \sin(\theta)\dot{y} - \sin(\theta)\dot{x} + \cos(\theta)\dot{y} + (L_1 + L_2)\dot{\theta}/2 \\ \cos(\theta)\dot{x} + \sin(\theta)\dot{y} - \sin(\theta)\dot{x} + \cos(\theta)\dot{y} - (L_1 + L_2)\dot{\theta}/2 \\ \cos(\theta)\dot{x} + \sin(\theta)\dot{y} + \sin(\theta)\dot{x} - \cos(\theta)\dot{y} + (L_1 + L_2)\dot{\theta}/2 \end{bmatrix}.$$

Steered Wheel Kinematics

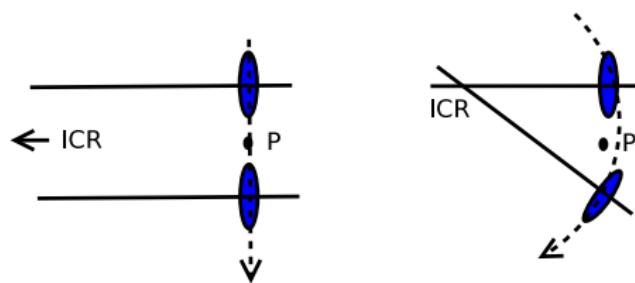


$$\begin{bmatrix} v \\ \dot{\theta} \end{bmatrix} = r\dot{\phi} \begin{bmatrix} 1 \\ \frac{\sin \beta}{L_2} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \dot{\phi} \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{v}{r} \\ \sin^{-1} \frac{L_2 \dot{\theta}}{v} \end{bmatrix}$$

Maneuverability

ICR - Instantaneous Center of Rotation.

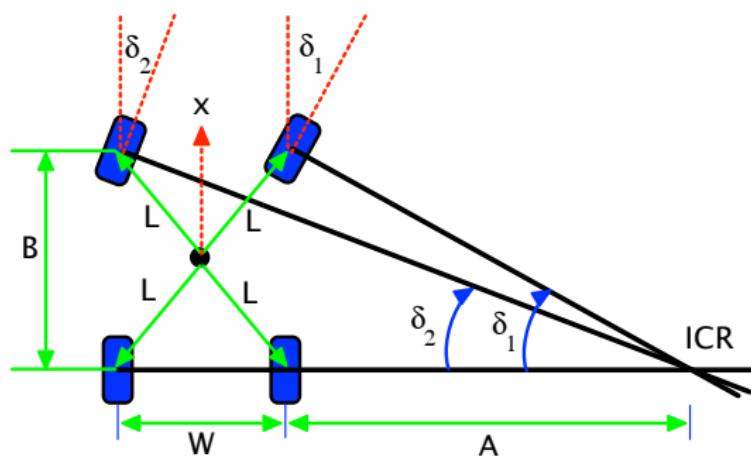
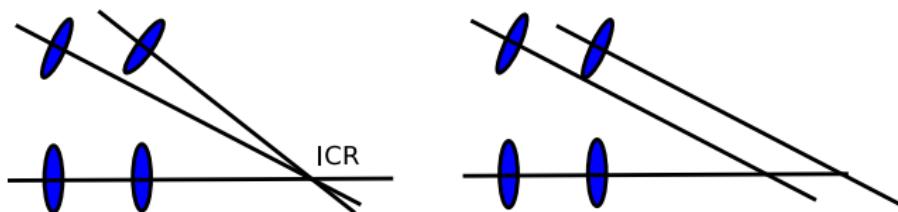
Each sliding constraint generates a zero motion line (orthogonal to the wheel plane). The intersection of the zero motion lines is the ICR.



In other words, each wheel is traveling on a circle whose center must be on the zero motion line.

Steered Wheel Kinematics

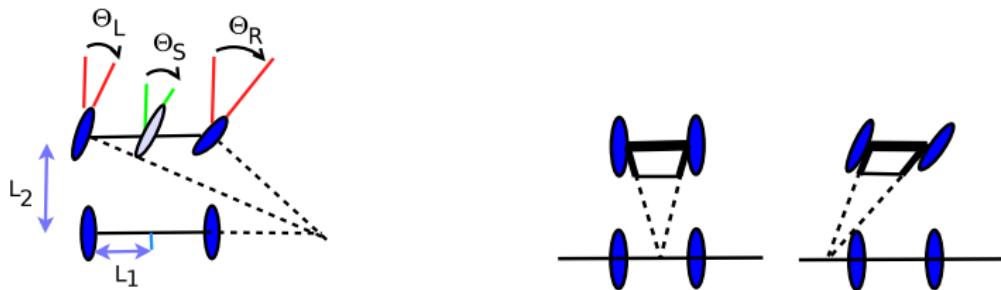
ICR - Instantaneous Center of Rotation.



Ackerman Wheel Kinematics

To satisfy the constraint placed on by the ICR, the steering system must satisfy the Ackerman equation:

$$\cot \theta_R - \cot \theta_L = \frac{2L_1}{L_2}$$



The effective steering angle, θ_S can be found by

$$\cot \theta_S = \frac{L_1}{L_2} + \cot \theta_R \quad \text{or} \quad \cot \theta_S = \cot \theta_L - \frac{L_1}{L_2}$$

Power and Signals

Main aspects of electricity

Voltage is the electrical pressure . . .

think of water pressure.

Measured in Volts.

Current is the flow of electrons . . .

think of water flow in a pipe.

Measured in Amps.

Power = Volts * Amps . . .

think of horsepower.

Measured in Watts

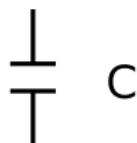
The basic components

Resistor (Ohms):



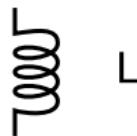
think of a narrowing in a pipe.

Capacitor (Farads):



think of a storage tank.

Inductor (Henrys):



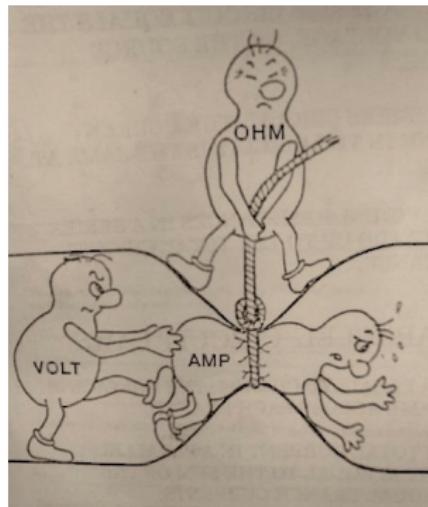
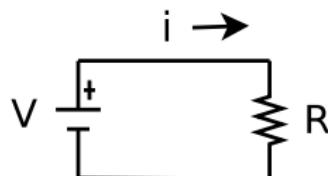
think of a flywheel in the pipe

Ohms Law

The fundamental law in circuits is Ohm's Law:

$$V = iR$$

where V is in volts, i is in amps, R is in Ohms.



Note the direction of current flow is the opposite electron flow.

Batteries

Batteries

Batteries

There are three common chemistries for rechargeable batteries:

- ▶ Lead-Acid
- ▶ Nickel-Metal Hydride (NiMH)
- ▶ Lithium Polymer (LiPo). ¹

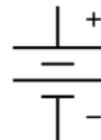


¹Yes, there are plenty more, but so that we can finish this section without getting a degree in chemistry ...

Batteries

Lead-Acid

- ▶ Cell: 2.1V
- ▶ Heavy
- ▶ High current
- ▶ Low energy density



NiMH

- ▶ Cell: 1.2V
- ▶ Light
- ▶ Relatively green
- ▶ High energy density

LiPo

- ▶ Cell: 3.7V
- ▶ Light
- ▶ No memory effect
- ▶ High energy density

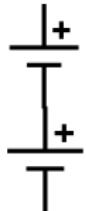


Connections

Series: Components connected in a linear chain:

For batteries this adds voltage, current is the same.

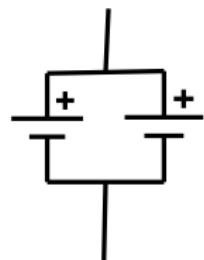
Series splits voltage.



Parallel: Components connected in an adjacent manner:

For batteries, this adds current, voltage stays the same.

Parallel splits current.

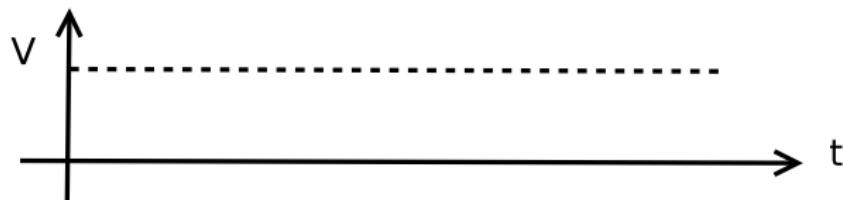


Power Supplies

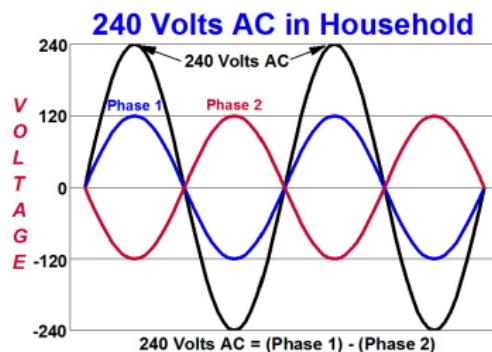
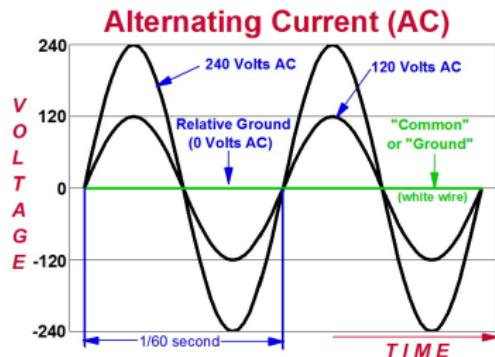
Power supplies

Basic Power Delivery

Direct Current:

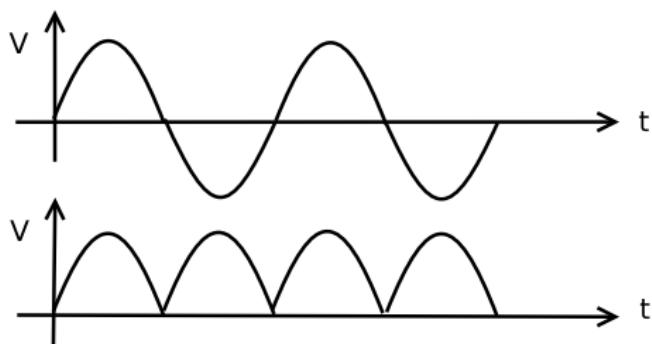


Alternating Current:

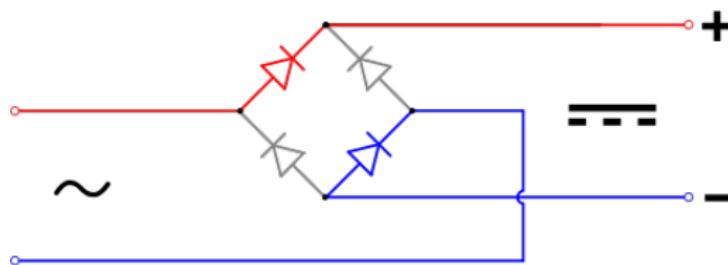


Basic Power Delivery

Bridge conversion to direct current:

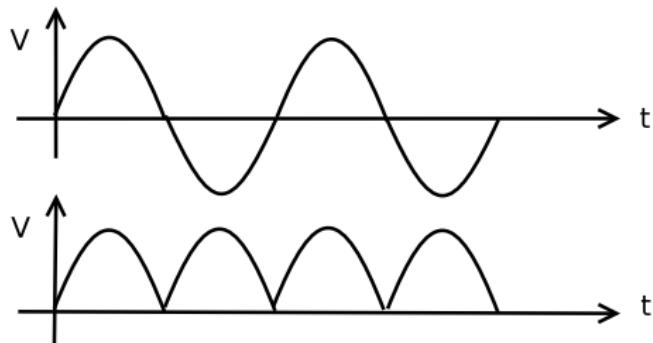


Diode bridge reroutes power

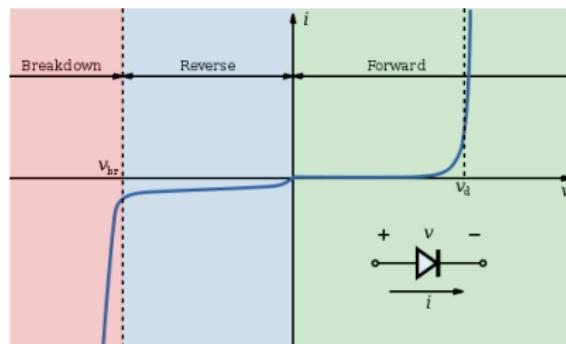


Basic Power Delivery

Bridge conversion to direct current:

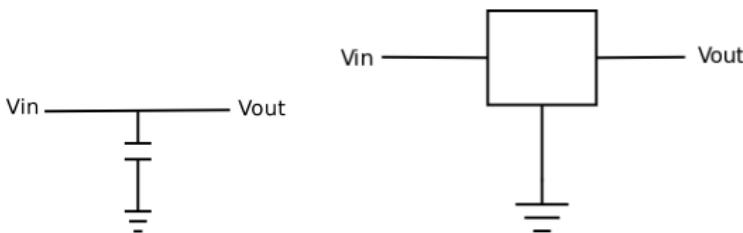


Diode:

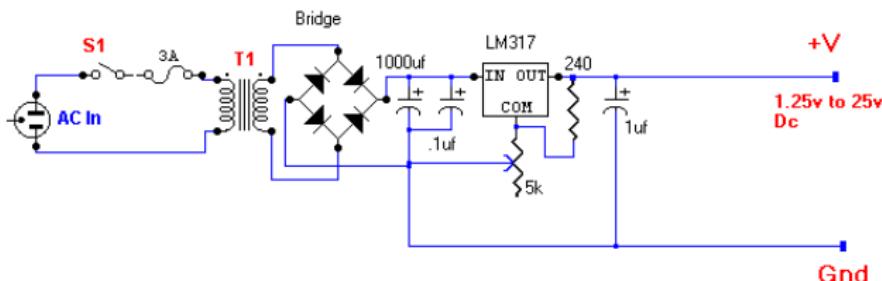


Basic Power Delivery

Smoothing using a capacitor and voltage regulation:



Power supply:

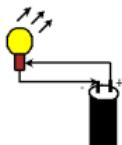


DC Power Supply

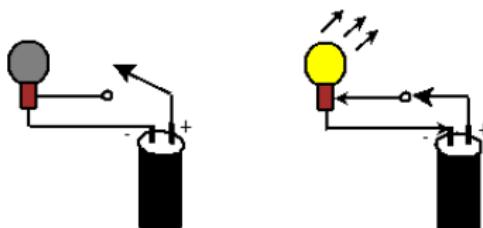
T1 = 115 v primary &
24 v secondary
AC Volts

Power Delivery

Say that you have a basic electric light circuit (like a flashlight):



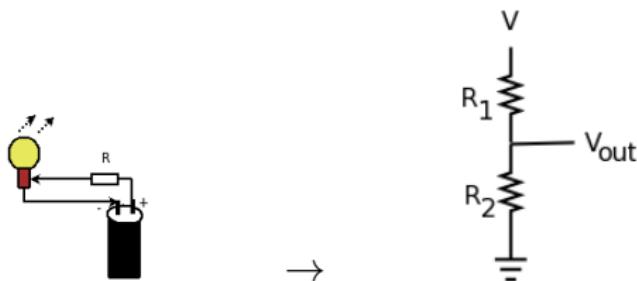
We are able to turn this on and off using a switch:



How can you dim the light?

Power Dissipation

A simple approach is to place a device along the flow of current that will resist the current flow - a resistor:



with $R = R_1$ and the resistance of the bulb is R_2 . The problem with this design is that some of the energy is wasted as heat in the resistor.

For low power circuits, this may not be a problem, but for higher power devices like for electric motors, considerable energy is wasted as heat - shorter battery life and maybe melted circuits!

Waste

Current through the resistors is

$$i = \frac{V}{R_1 + R_2}.$$

Voltage drop across R1 is

$$V_{R_1} = \left(\frac{R_1}{R_1 + R_2} \right) V.$$

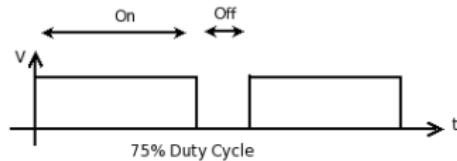
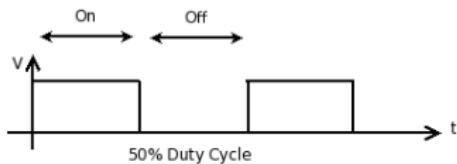
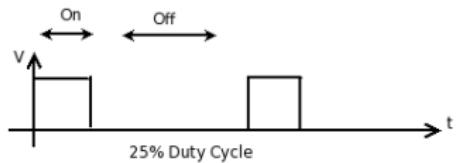
Power is

$$W = i * V_{R_1} = R_1 \left(\frac{V}{R_1 + R_2} \right)^2$$

For low power circuits, this may not be a problem, but for higher power devices like for electric motors, considerable energy is wasted as heat. [As well as shorter battery life and maybe melted circuits!]

Waste

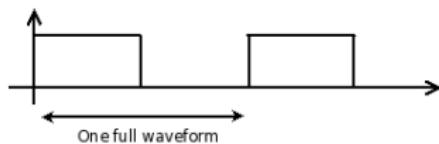
The solution is to switch on and off the power very quickly. To see what we mean, here is a picture of the voltage though time.



The amount of time the pulse is high compared to low is the duty cycle. Often this is a percent of the pulse length which is called the period.

Pulses

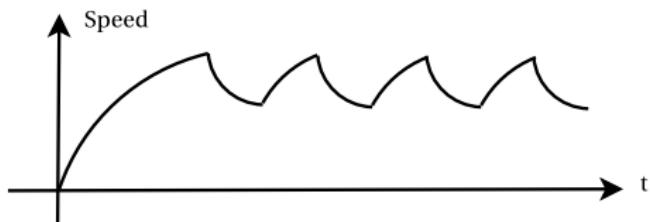
Why does this matter? By this method, we deliver a fraction of the energy which then makes light dimmer. It does not have the energy waste as compared to using a resistor. If we run the on and off fast enough, our eyes will not see the flicker and it will just appear dimmer.



This is also the method by which we control an electric motor. The frequency of this waveform does not change (because the duration of a single waveform is unchanged). The time that the voltage is high compared to the voltage is low does change.

PWM Motor Control

During the high part of the waveform an electric motor will start to increase in speed. During the low part the motor will coast and slow down.



PWM Motor Control

You may ask how we switch the power on and off really fast.

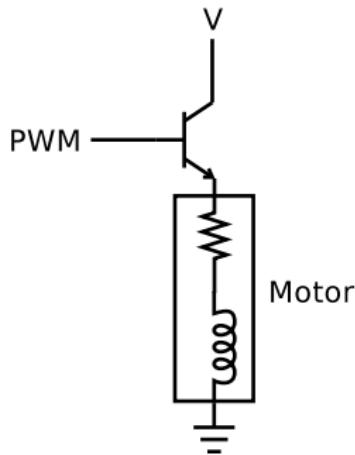
It is not like we have a little light switch and 87 cups of coffee.

Hard for us, trivial for a computer. In fact, this is the basic way computers operate.

They switch lines on and off millions or even billions of times per second. A program can be used to switch on and off an output line at a variety of frequencies and duty cycles.

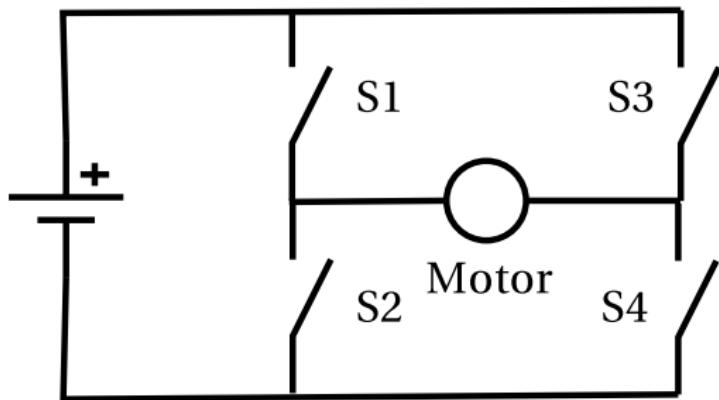
PWM Motor Control

Using a pwm to drive a transistor is the way to get power delivered.

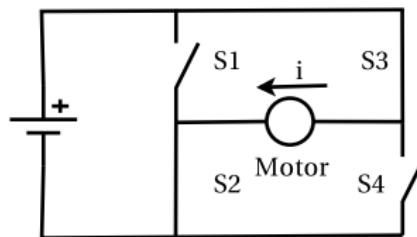
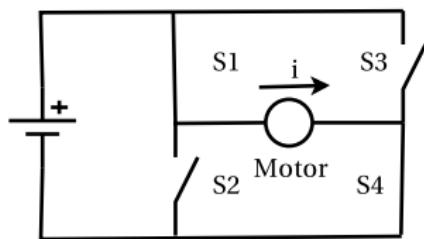


One minor problem is that this only runs one way. An H-bridge is a way to provide a reverse current.

H-Bridge



H-Bridge



H-Bridge

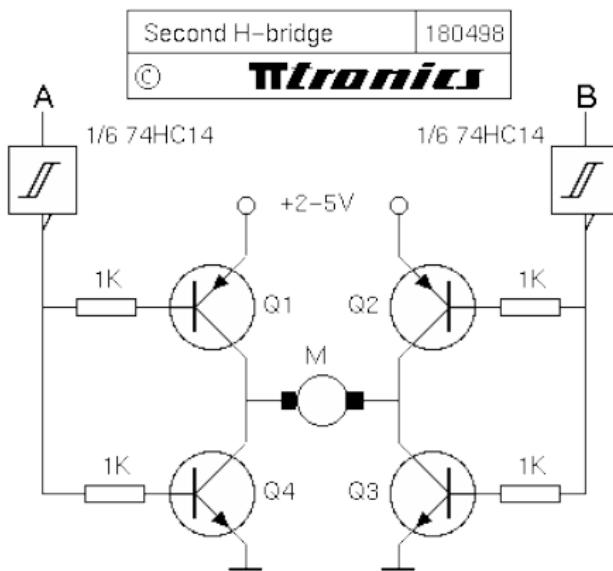
Q1, Q2 = BC327-25 or 2N2905A
Q3, Q4 = BC337-25 or 2N2219A
M = Servo motor

When A and B are equal (both 0 or both 1), the motor doesn't run

When A = 1 and B = 0, the motor turns in one direction; when A = 0 and B = 1, it turns the other way.

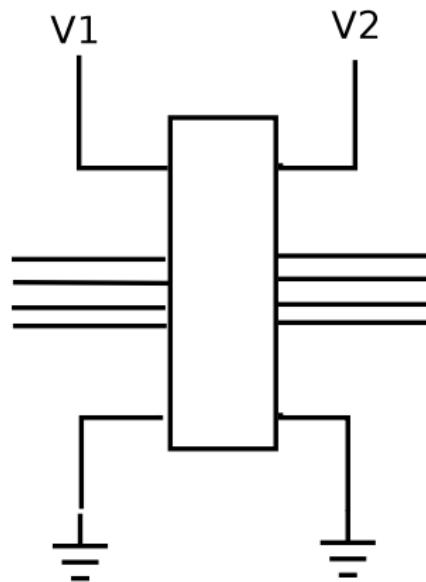
The circuit has handled a stalled motor @ 360mA with no trouble.

@ Vcc = 4.5V, the 74HC standard output sinks and sources 4mA



Interfacing

Level Shifters - interfacing hardware



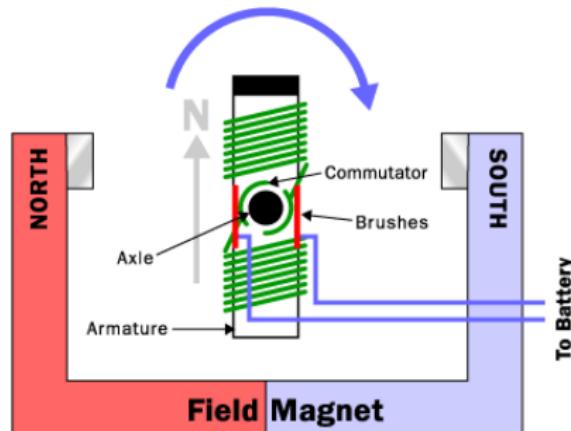
Motor Controller

A motor controller is

- ▶ H-Bridge
- ▶ FETs
- ▶ Micro-controller
- ▶ PWM unit

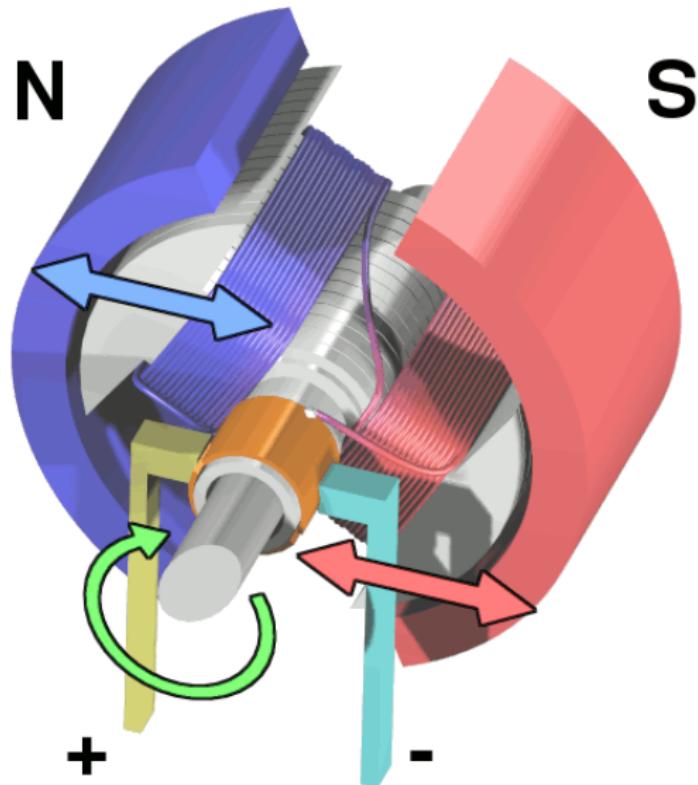
Motors and Servos

DC Motors

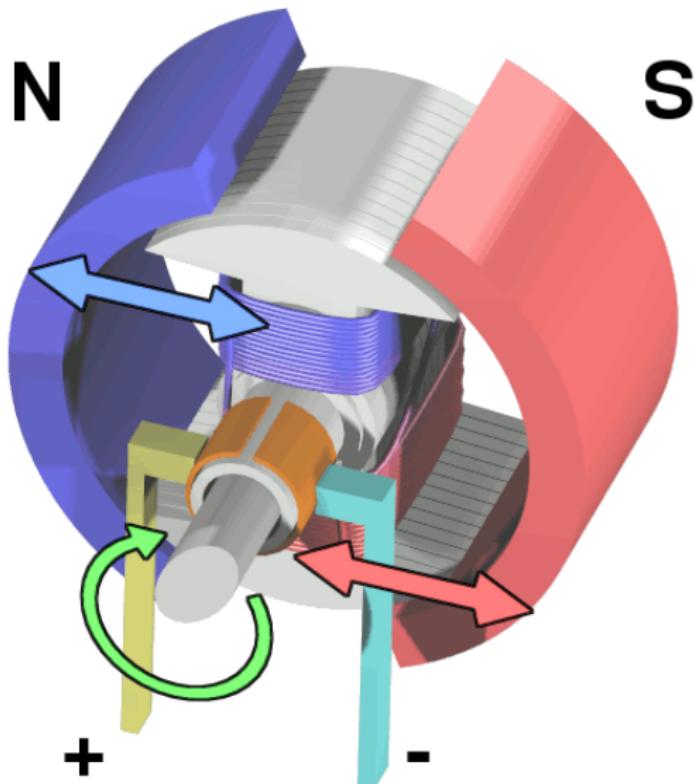


©2001 HowStuffWorks

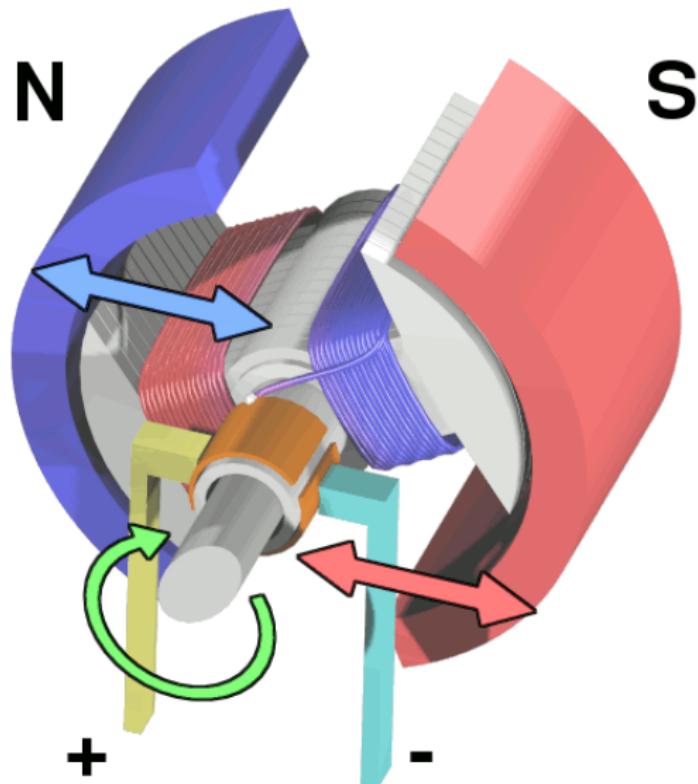
DC Motors



DC Motors

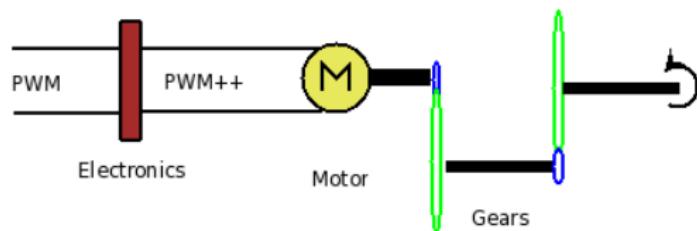


DC Motors

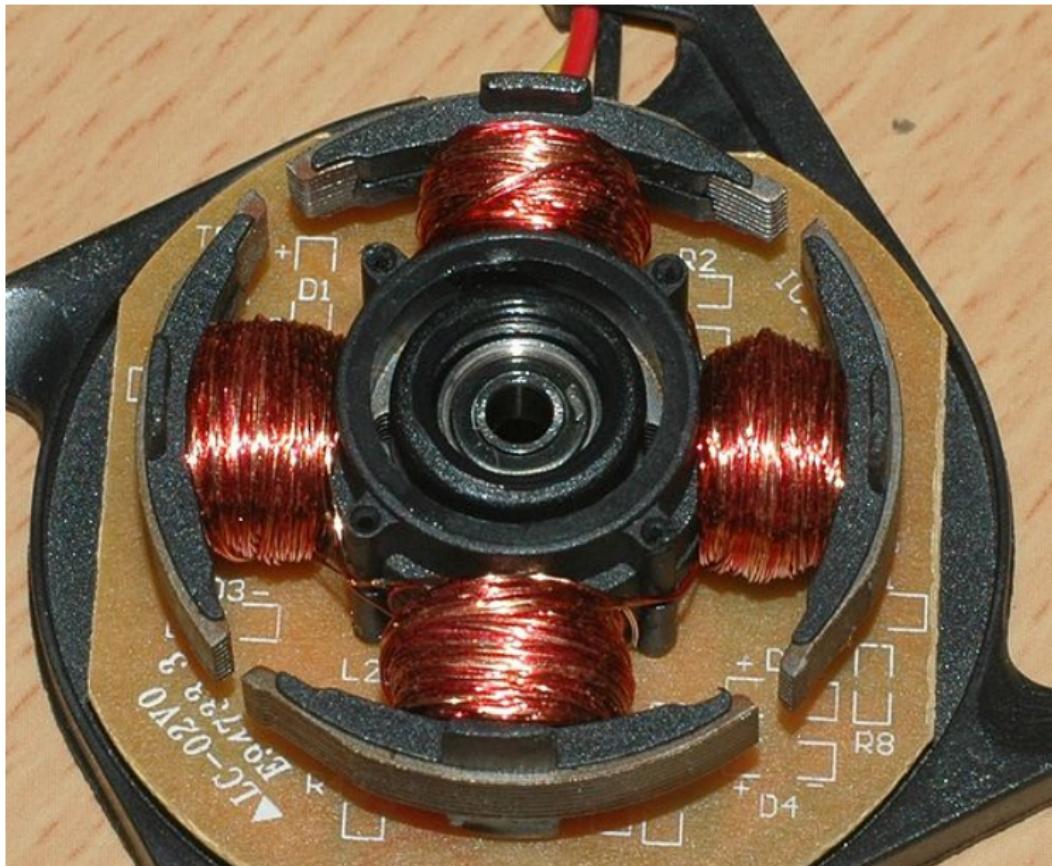


Servo

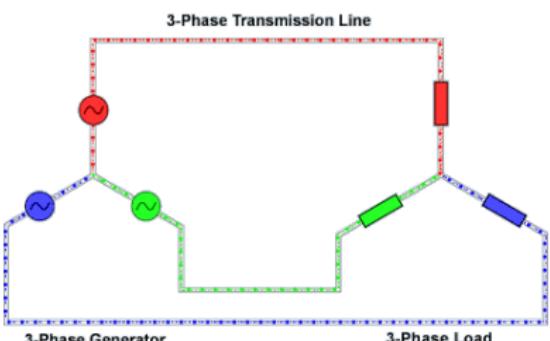
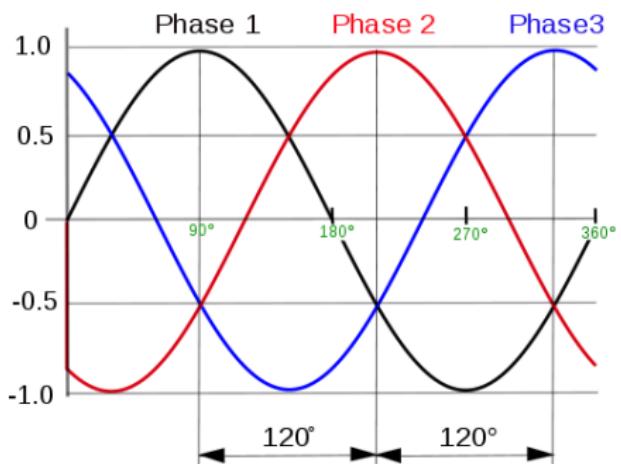
A servo is an electric motor, some electronics and a gear box



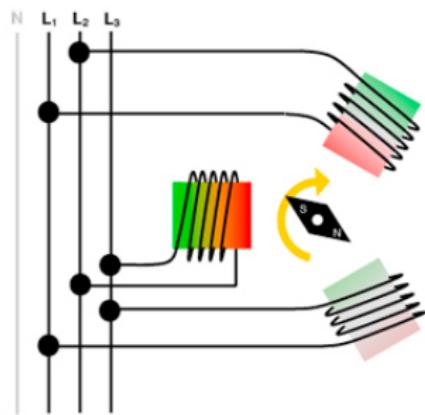
Motor detail



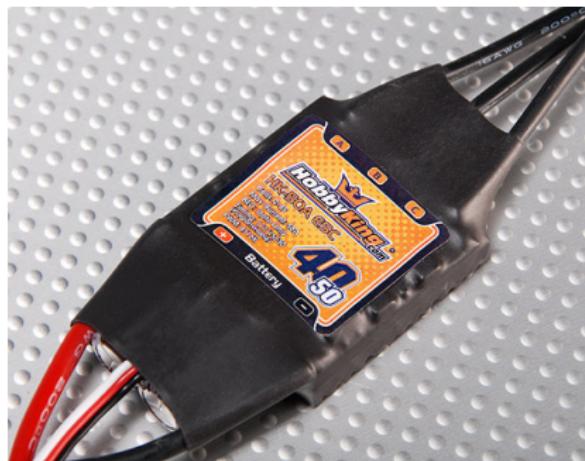
Three Phase



More three phase



More three phase



Sensors

Sensor Background

- ▶ Why should a robotics engineer know about sensors?
 - ▶ Key technology for perceiving the environment
 - ▶ Understanding the physical principles enables appropriate use
- ▶ Understanding the physical principle behind sensors enables us
 - ▶ To properly select the sensors for a given application
 - ▶ To properly model the sensor system, e.g. resolution, bandwidth, uncertainties
 - ▶ To define the needs in collaboration with the sensor system suppliers

Sensor Classes

Classes

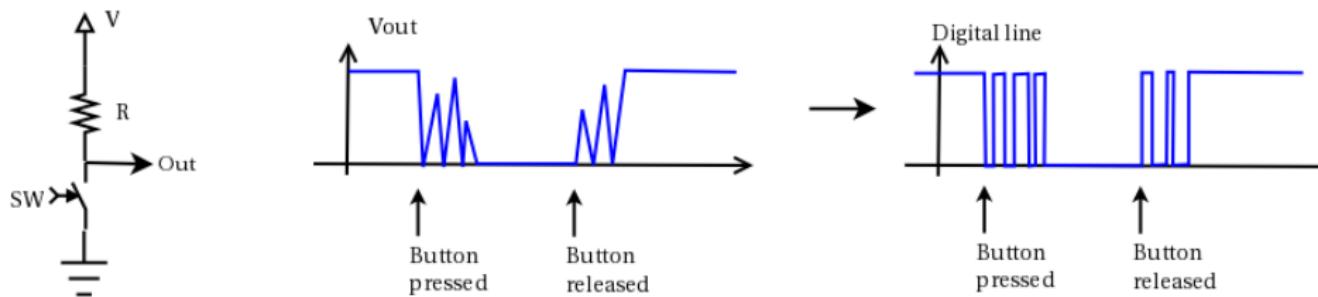
- ▶ Properioceptive
 - ▶ measure values internally to the system (robot)
 - ▶ values like motor speed, wheel load, orientation, battery status
- ▶ Exteroceptive
 - ▶ information from the robot's environment
 - ▶ distances to objects, intensity of ambient light, unique features

How

- ▶ Passive sensors
 - ▶ energy from environment
- ▶ Active sensors
 - ▶ emit energy in the correct configuration and measure the reaction
 - ▶ better performance but influences the environment

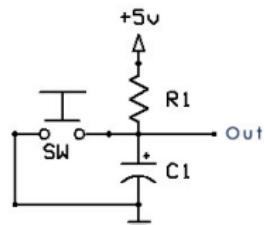
Switches

The most elementary sensor - Switches ...

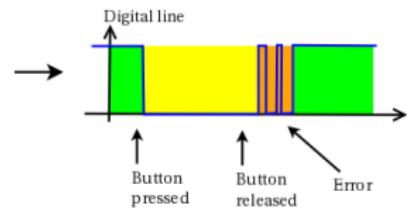
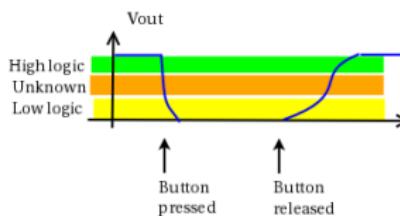


Switches

Debounce issues

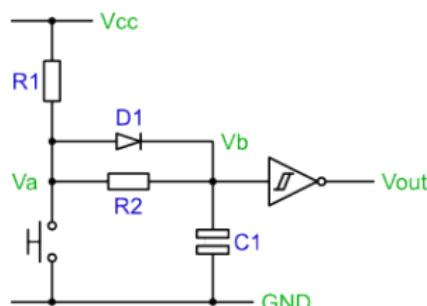
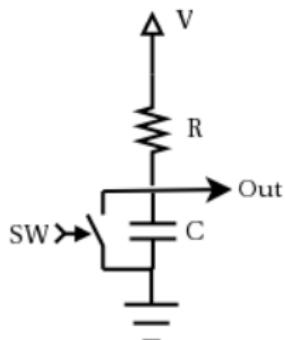


www.ikalogic.com



Switches

Debounce solutions in hardware



Software solutions are also available and normally depend on waiting a short time.

LEDs

Light emitting diodes - LEDs

These are diodes which emit at specific EM bands.

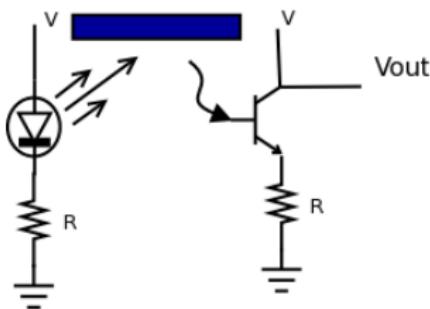


LEDs have a variety of operating specs.

Assume we have one that operates in the 3-6 volt range and current level is 20mA. If we select $V = 5$ then the resistor should be $R = V/I = 5/.02 = 250$. Since 250 is not a standard value, we select $R = 270$ ohms.

LEDs and obstacle detection

LEDs can emit in non-visible ranges as well - UV or IR. They can be used for simple object detection in combination with a phototransistor:



This system can be used for simple occupancy detection or close obstacle detection.

Encoders

Wheel encoders

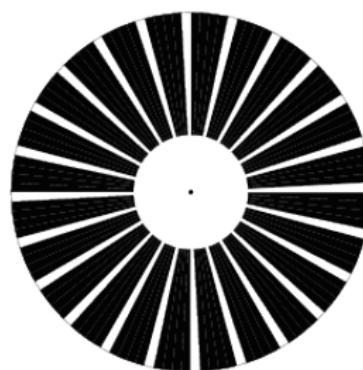
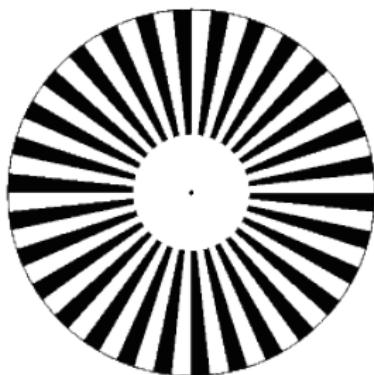
Navigation and localization can be challenging problems in mobile robotics. One approach to measuring motion is to measure the amount of rotation of a wheel.

- ▶ Measure position or speed of a wheel or dial.
- ▶ One may then integrate to get an estimate of the position: odometry.
 - ▶ Really this is a summation.
 - ▶ Rather error prone and methods are needed to correct it.
- ▶ Optical encoders are proprioceptive sensors
 - ▶ the position estimate in relation to a fixed reference frame is only valuable for short movements.
- ▶ typical resolutions: 64 - 2048 increments per revolution

Encoders

Two elements are required: an encoder and a detector.

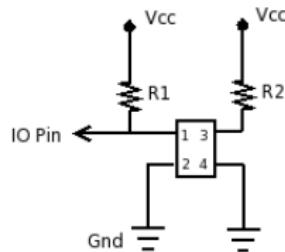
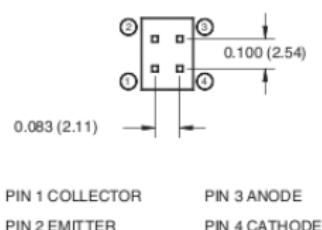
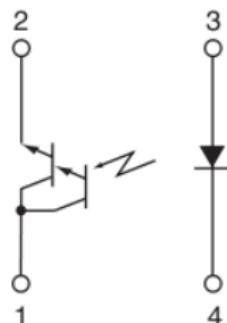
An encoder pattern may be attached (glued) to the inside of a robot wheel.
Simple encoder patterns are just alternating black and white radial stripes.
Two examples are



Encoder wiring

To read the encoder, you will need an optical sensor. Typically one uses an IR LED (IR light emitting diode) and phototransistor pair.

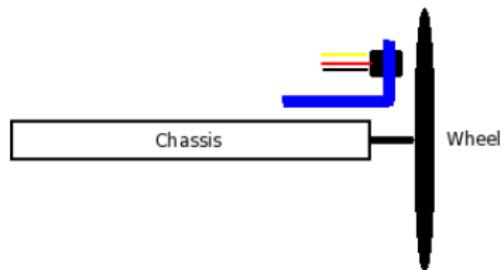
Schematic



These are packaged in single units, for example the Fairchild QRD1313. This has the LED and the phototransistor packaged into a unit that is 6.1mm x 4.39mm x 4.65mm (height).

Encoder mounting

Glue the encoder pattern to the inside of the wheel and mount the detector:

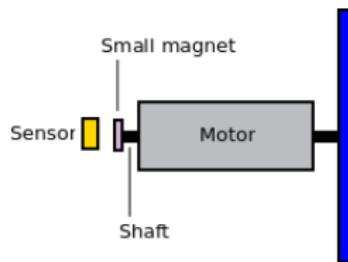


Compass

- ▶ Used for over 4000 years
 - ▶ Chinese suspended a piece of natural magnetite from a silk thread and used it to guide a chariot over land
- ▶ Magnetic field on earth
 - ▶ absolute measure for orientation
 - ▶ special property of the liquid iron core
 - ▶ pole reversals occur
- ▶ Large variety of solutions to measure the earth's magnetic field
 - ▶ mechanical magnetic compass
 - ▶ direct measure via Hall-effect or magnetoresistive sensors
- ▶ Major drawbacks
 - ▶ weakness of the earth's field
 - ▶ easily disturbed by magnetic objects or other sources
 - ▶ not always good indoors or on planets without significant fields

Magnetic encoding

It is possible to use Hall-effect or other similar devices to do encoding.



Orientation

Gyroscopes

- ▶ Heading sensors that keep the orientation to a fixed frame
 - ▶ absolute measure for the heading of a mobile system
- ▶ Two categories
 - ▶ Mechanical Gyroscopes
 - ▶ Standard gyro (angle)
 - ▶ Rate gyro (speed)
 - ▶ Optical Gyroscopes
 - ▶ Rate gyro (speed)



Acceleration

Accelerometers: Measures acceleration in a particular direction.

- ▶ IMU - Inertial Measurement Unit
 - ▶ 3 - Accelerometers, 3 - Gyroscopes \Rightarrow 6DOF
 - ▶ 3 - Accelerometers, 3 - Gyroscopes, 3 axis compass \Rightarrow 9DOF
- ▶ Basis for AHRS: Attitude and Heading Reference System

Inclineometer: tilt sensor

These devices often have serial (or USB) connections and return text strings of data at some Hz.



AHRS: Attitude and Heading Reference System

AHRS consist of either solid-state or MEMS gyroscopes, accelerometers and magnetometers on all three axes.

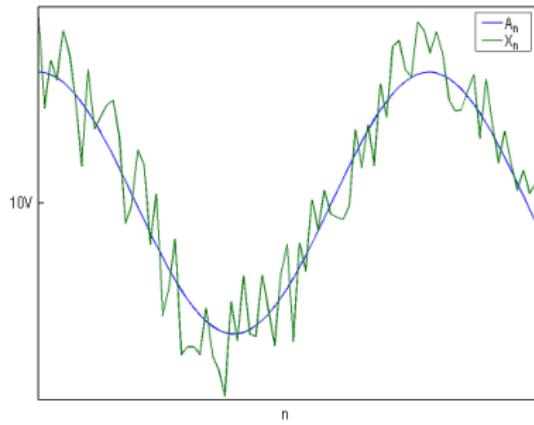
The key difference between an IMU and an AHRS -

is the addition of an on-board processing system in an AHRS which provides solved attitude and heading solutions versus an IMU which just delivers sensor data to an additional device that solves the attitude solution.

AHRS - Filtering

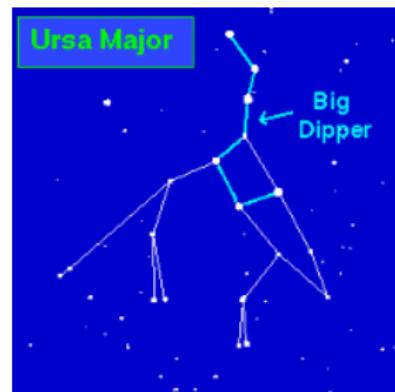
A form of non-linear estimation such as a Kalman filter is typically used to compute the solution from these multiple sources.

AHRS differs from traditional inertial navigation systems (INS) by attempting to estimate only attitude (i.e. roll, pitch, yaw a.k.a heading) states, rather than attitude, position and velocity as is the case with an INS.



Beacons

- ▶ Elegant way to solve the localization problem
- ▶ Beacons are signaling systems with known positions
- ▶ Beacon base navigation is very old
 - ▶ Natural beacons: stars, mountains, sun
 - ▶ Artificial beacons: lighthouses
- ▶ GPS, Global Positioning System, revolutionized navigation
 - ▶ One of the key sensor systems in mobile robotics
 - ▶ Outdoor only
- ▶ Indoor beacon systems require
 - ▶ Modify the environment
 - ▶ Limited flexibility to changing conditions



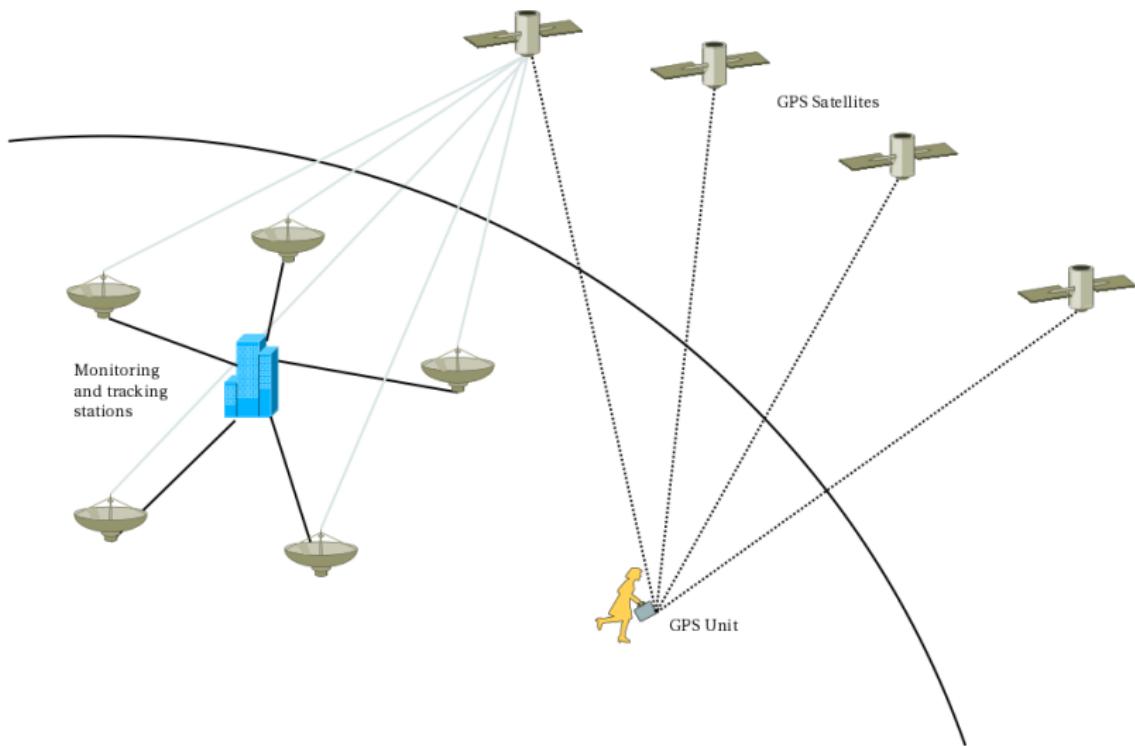
Overview

- ▶ Developed for military use
- ▶ Recent availability for non-military
- ▶ 24 satellites (with 3 spares) orbiting the earth every 12 hours at an altitude of 20,190 km
- ▶ Four satellites in each of the six planes inclined 55 degrees with respect to the plane of the earth's equator
- ▶ Location of a GPS receiver is determined by a time of flight measurement

Challenges

- ▶ Time synchronization between satellites and GPS receiver
- ▶ Real time update of exact location of the satellite
- ▶ Precise measurement of time of flight
- ▶ Interference with other signals

GPS



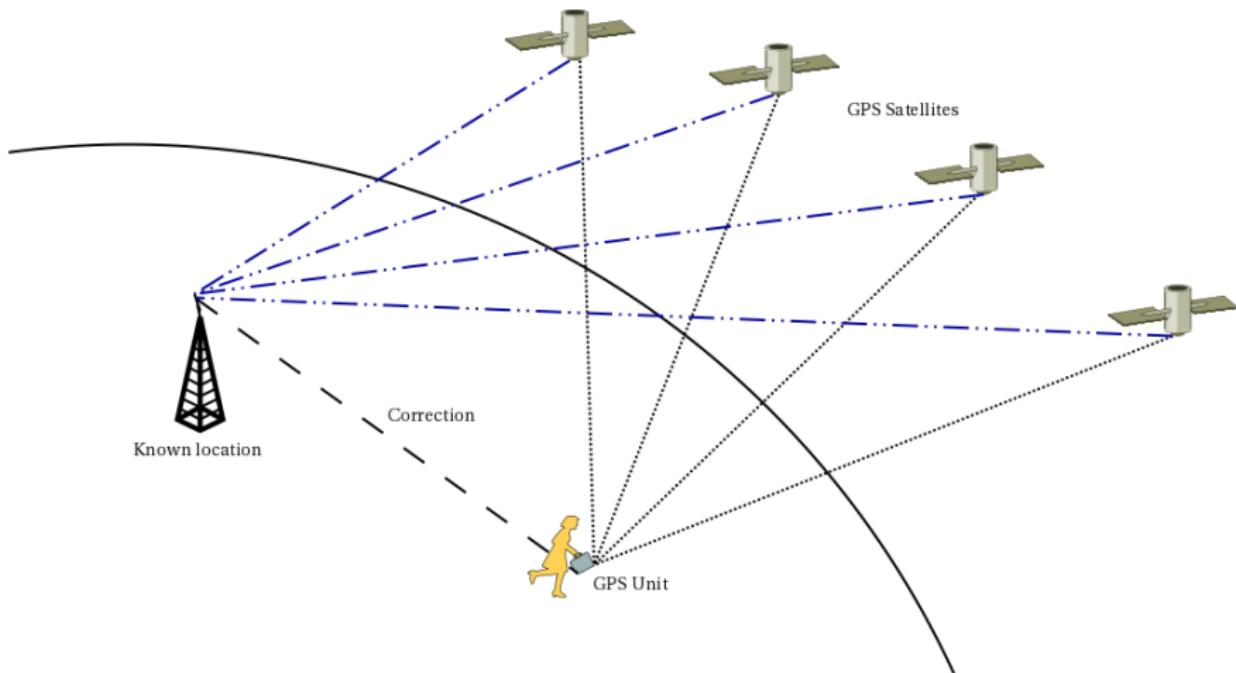
GPS Details

- ▶ Time synchronization:
 - ▶ atomic clocks on each satellite
 - ▶ monitoring them from different ground stations
- ▶ Ultra-precision time synchronization is extremely important
 - ▶ electromagnetic radiation propagates at light speed - 0.3m per nanosecond
 - ▶ position accuracy proportional to precision of time measurement
- ▶ Real time update of exact location of the satellites
 - ▶ monitoring the satellites from a number of widely distributed ground stations
 - ▶ master station analyses all the measurements and transmits the actual position (of each) to each of the satellites.

GPS Details

- ▶ Exact measurement of the time of flight
 - ▶ the receiver correlates a pseudocode with the same code coming from the satellite
 - ▶ the delay time for the best correlation represents the time of flight
 - ▶ quartz clock on gps receivers are not very precise
 - ▶ the range measurement with four satellites allows one to identify the three values (x, y, z) for the position and the clock correlation Δt .
- ▶ Commercial GPS receivers allows position accuracies down to a couple of meters.

Differential GPS



Partial Worked Example

Assume that you have four beacon towers located in roughly a square over a 10km x 10km patch of land. You place a coordinate system on the land and measure the beacon locations.

The locations in meters are B1 (0,0), B2 (56, 9752), B3 (9126, 7797), B4 (9863, 218).

If the beacons transmit a packet with a time stamp, then a mobile system with an accurate clock can determine its location in the instrumented area.

Determine locations if $t_1 = 22793$ ns, $t_2 = 15930$ ns, $t_3 = 20817$ ns, $t_4 = 29793$ ns.

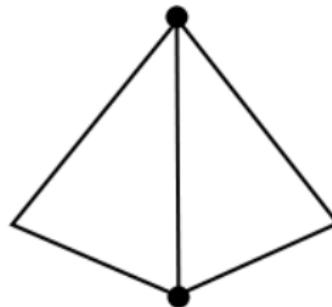
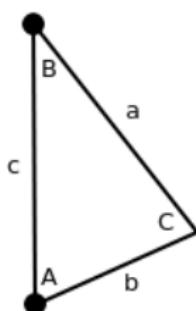
Example

Since this is strictly a 2D problem (over 10km the curvature of the earth is not a significant issue), we can use trig, specifically the law of cosines, to solve the problem.

The distances are found via $d = ct$: $d_1 = 6838m$, $d_2 = 4779m$, $d_3 = 6245m$, $d_4 = 8938m$.

From the figure below we have that

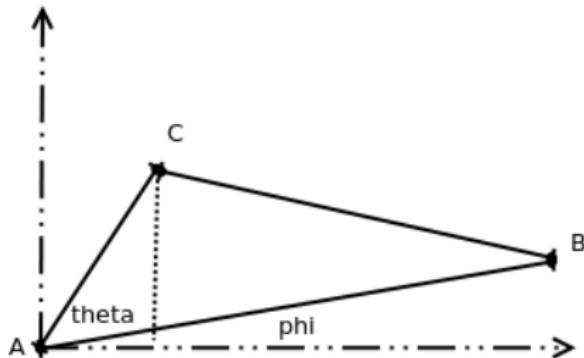
$$b^2 = a^2 + c^2 - 2ac \cos(B), \quad a^2 = c^2 + b^2 - 2cb \cos(A).$$



Example

The right figure shows the basic dual solution. Using the region of interest we can eliminate one point. Using the figure below, we can compute ϕ from beacon location data.

We already computed θ from the law of cosines. This tells us the actual angle off of the axis $\theta + \phi$.



Example

We know the distance of the segment AC which is the basic range data (call this R).

Using

$$x = R \cos(\theta + \phi)$$

$$y = R \sin(\theta + \phi)$$

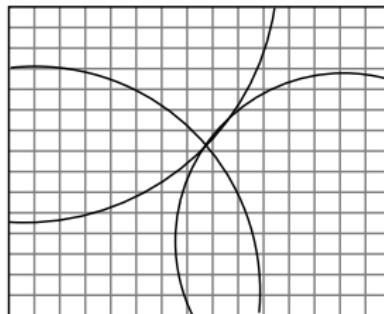
We obtain a location (x, y) . This can be repeated for the three other beacon pairs. Take the three closest points and average their values.

Example

One way to approach this problem is to rework it into a optimization problem.

If we are a certain distance (in two dimensions) away from a beacon, then we lie on a circle where the radius of the circle is the distance away from the beacon.

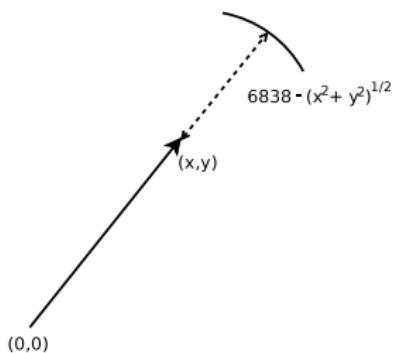
The object must lie on all of the circles which are have the given distance.



Example

We would like to minimize the distance that our selected point (x, y) lies off of each circle.

The distance the point misses the circle from B1 is $|\sqrt{x^2 + y^2} - 6838|$



This is a hard error to work with later and so we use a proportional error term

$$(x^2 + y^2 - 6838^2)^2$$

Example

Form the total error function which is the sum of distance errors.

$$\begin{aligned} E = & |x^2 + y^2 - 6838^2| \\ & + |(x - 56)^2 + (y - 9752)^2 - 4779^2| \\ & + |(x - 9126)^2 + (y - 7797)^2 - 6245^2| \\ & + |(x - 9863)^2 + (y - 218)^2 - 8938^2| \end{aligned}$$

If $E = 0$ then we are at the (x, y) point that matches all four distances.

Since there is measurement error, $E > 0$, so we are looking for the minimum value for E . It is possible to do this via calculus, but ... try gradient descent.

Example

Modify the error function to have

$$E = [x^2 + y^2 - 6838^2]^2 + [(x - 56)^2 + (y - 9752)^2 - 4779^2]^2 + [(x - 9126)^2 + (y - 7797)^2 - 6245^2]^2 + [(x - 9863)^2 + (y - 218)^2 - 8938^2]^2$$

The gradient is $\nabla E = \langle \partial E / \partial x, \partial E / \partial y \rangle$.

Julia can compute this automatically, we can compute using a symbolic package or we can try a numerical method to estimate.

Example

Set $x_0 = 5000$, $y_0 = 5000$, $k = 0$, $t = 1$

While ($t > t_0$)

- ▶ $u = \nabla E(x_k, y_k) / \|\nabla E(x_k, y_k)\|$

- ▶ $(a, b) = (x_k, y_k) - tu$

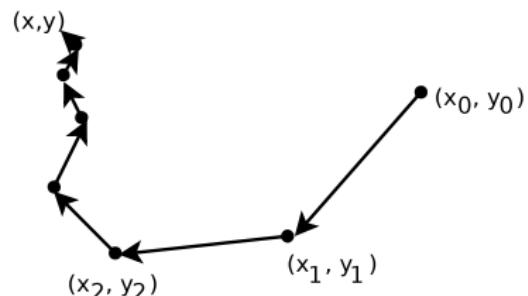
- ▶ while [$E(a, b) > E(x_k, y_k)$]

- $t = t/2$

- $(a, b) = (x_k, y_k) - tu$

- ▶ $k = k + 1$

- ▶ $(x_k, y_k) = (a, b)$



$x = 3120$ $y = 6085$

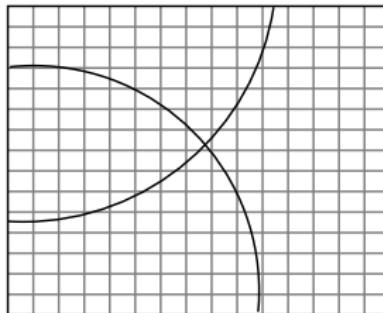
There are plenty of other ways to treat this problem.

- (1) One may intersect two circles to provide the location of the two intersecting points and then proceed over all combinations:

$$(x - a_1)^2 + (y - b_1)^2 = r_1^2, \quad (x - a_2)^2 + (y - b_2)^2 = r_2^2$$

- (2) Use a grid and step through each grid point.

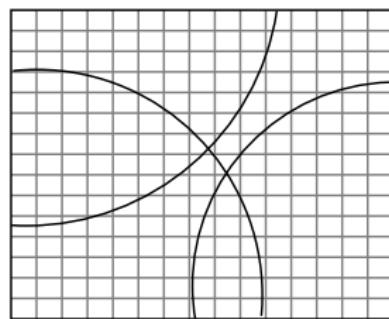
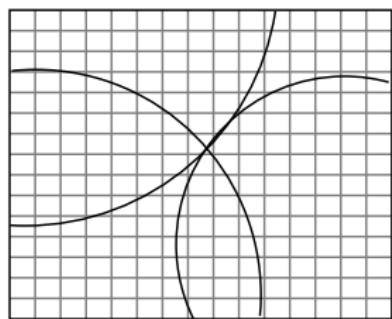
The grid cell with the most correct distances is the answer.



See if you can come up with other approaches.

Issues

The ideal case and the case with noise:



Range Sensors

- ▶ Large range distance measurement called *range sensors*
- ▶ Short range measurements sometimes known as proximity detectors
- ▶ Range information:
 - key element for localization and environment modeling
- ▶ Ultrasonic sensors as well as laser range sensors make use of propagation speed of sound or light respectively.
- ▶ The traveled distance of a sound wave or light wave is given by

$$d = c \cdot t$$

- ▶ where
 - ▶ d is the distance traveled (round trip)
 - ▶ c is the speed of the wave
 - ▶ t is the time of flight

Range Sensors

Some details ...

- ▶ Speed of sound: 0.3 m/ms
- ▶ Speed of light: 0.3 m/ns (10^6 times faster than sound)
- ▶ 3 meters
 - ▶ is 10 ms for an ultrasonic system
 - ▶ is 10 ns for a laser range sensor
 - ▶ time of flight with electromagnetic signals is not easy
 - ▶ laser rangers are expensive and delicate

The quality of time of flight range sensors mainly depends on

- ▶ Uncertainties about the exact time of arrival of the reflected signal
- ▶ Inaccuracies in the time of flight measurement
- ▶ Opening angle of transmitted beam
- ▶ Interaction with the target
- ▶ Variation of the propagation speed
- ▶ Speed of robot and target

Transmit a packet of ultrasonic pressure waves.

Distance d of the echoing object can be calculated based on the propagation speed of sound, c , and the time of flight, t :

$$d = \frac{c \cdot t}{2}$$

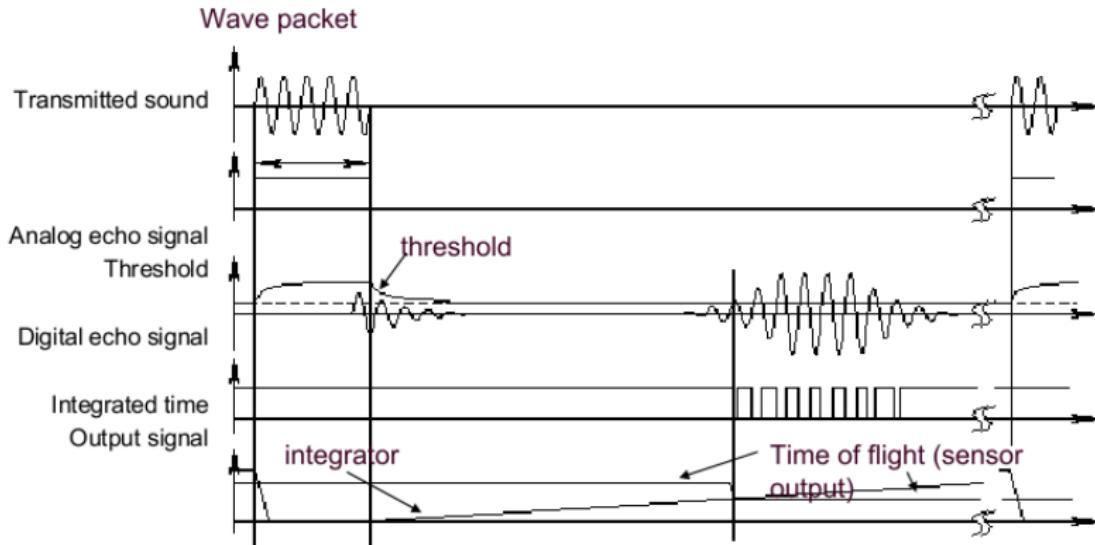
The speed of sound, c (about 340 m/s), in air is given by

$$c = \sqrt{\gamma R T}$$

where

- ▶ γ adiabatic index
- ▶ R gas constant
- ▶ T gas temperature in Kelvin

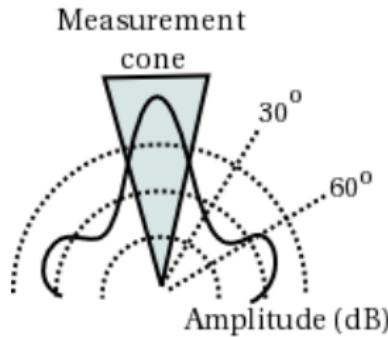
Sonar Echos



Signals of an ultrasonic sensor

Sonar Details

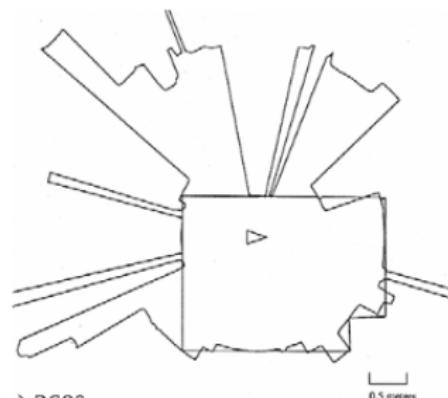
- ▶ Typical frequency: 40 - 180 KHz
- ▶ Generation of sound waves via a Piezo transducer
 - ▶ transmitter and receiver separated or not
- ▶ Sound beam propagates in a cone (approx)
 - ▶ opening angles around 20 to 40 degrees
 - ▶ regions of constant depth
 - ▶ segments of an arc (sphere for 3D)



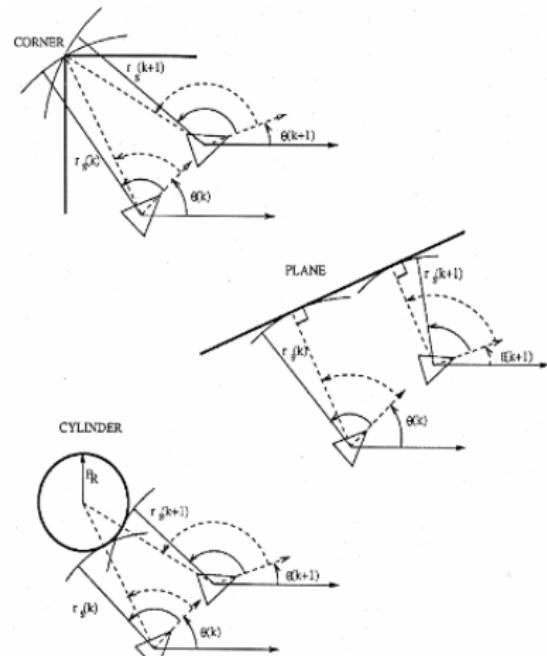
Sonar Reflection

Issues:

- ▶ soft surfaces that absorb sound energy.
- ▶ surfaces that are angled away from the direction of sound travel .

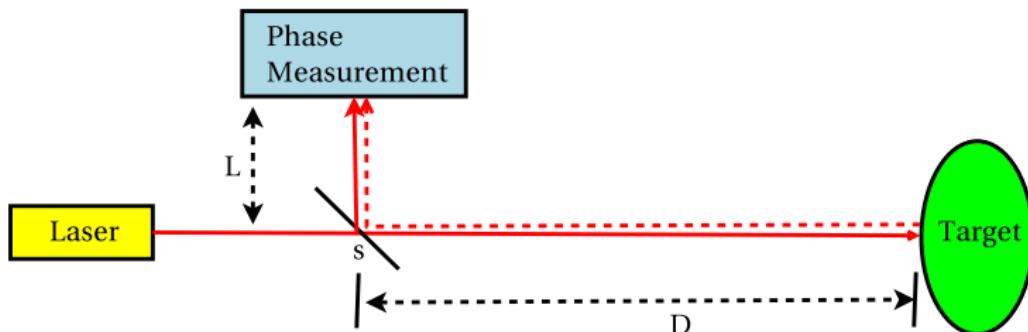


a) 360° scan



b) results from different geometric primitives

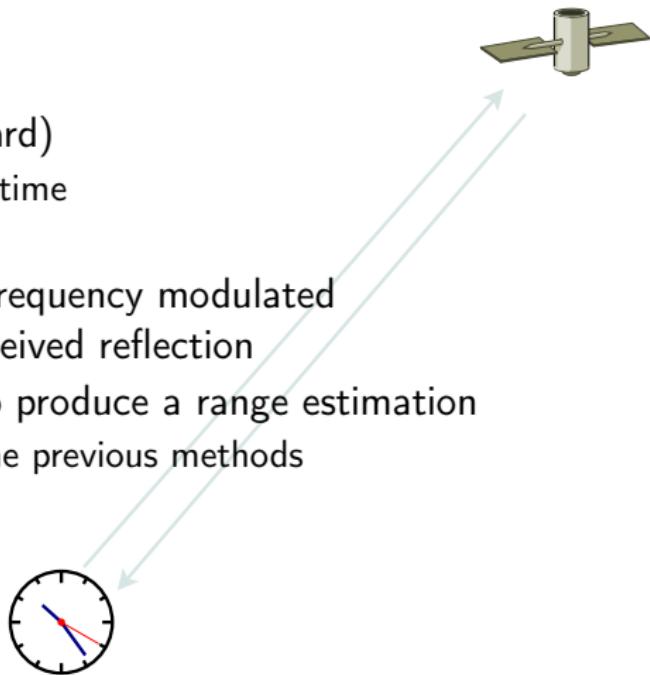
Laser Ranging



- ▶ Transmitted and received beams coaxial
- ▶ Transmitter illuminates a target with a collimated beam (laser)
- ▶ Receiver detects the time needed for round trip
- ▶ A mechanical mechanism with mirror sweeps (2D or 3D).

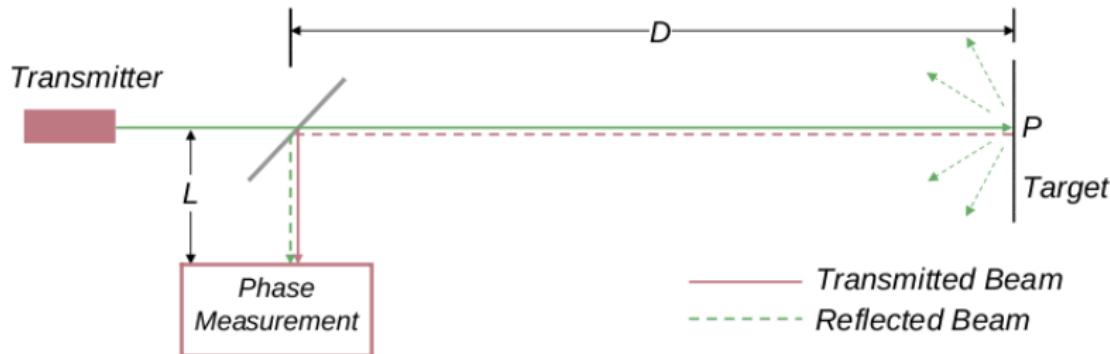
Time of flight measurement

- ▶ Pulsed laser (current standard)
 - ▶ measurement of elapsed time
 - ▶ resolving in picoseconds
- ▶ Beat frequency between a frequency modulated continuous wave and its received reflection
- ▶ Phase shift measurement to produce a range estimation
 - ▶ technically easier than the previous methods



LIDAR

- Phase-Shift Measurement



$$D' = L + 2D = L + \frac{\theta}{2\pi}\lambda \quad \lambda = \frac{c}{f}$$

Where:

c is the speed of light, f is the modulating frequency, D' is the distance covered by the beam.

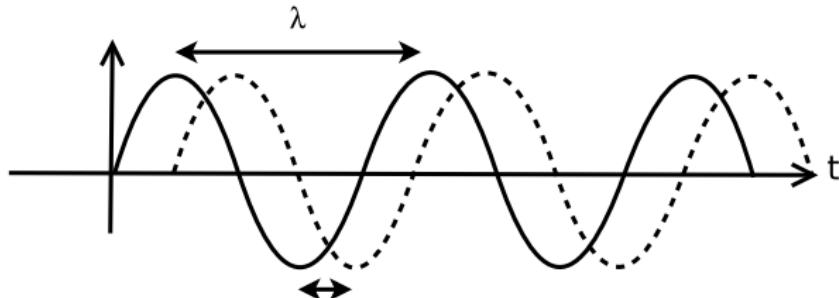
Note: if $f = 5 \text{ Mhz}$ then $\lambda = 60 \text{ meters}$.

- Distance D , the distance between the beam splitter and the target,

$$D = \frac{\lambda}{4\pi}\theta$$

where θ is the phase difference between the transmitted and reflected beam

- Range estimate is not unique,
for example if $\lambda = 60$ then a target at 5, 35, 65, ... meters give the same phase shift.



Example

Assume you are using a laser diode to build a distance sensor.

- ▶ What is the wavelength of the modulated frequency of 12MHz?
- ▶ If you measure a 20 degree phase shift, this value corresponds to what distances?
- ▶ What other modulation frequency would be a good choice to isolate the value? (show this)
- ▶ How would you do the modulation and phase shift measurement?

Example

Wavelength = $\lambda = c/f = 3(10^8)/(12(10^6)) = 25$ meters.

A 20 degree shift is $(20/360)$ of the wavelength, so we get

$$25/18 \approx 1.389m$$

The actual distance is $1/2$ due to the round trip: 0.6945m.

Focus on the full trip. This would correspond to

$$1.389 + 25n \text{ for } n = 0, 1, 2, 3\dots$$

$$= 1.389, 26.389, 51.389, \dots$$

Example

Which distance???? Select another frequency and see what it tells you. As long as our values are relatively prime, frequency selection is pretty open. Factors of 12 are 2, 3, 4.

So 5 Mhz would work (as would 17 Mhz and many others) for some distance. Using 5Mhz, we have a wavelength of 60 meters.

We measure and we find that we have a 158 degree phase shift. This gives us 26.33 meters.

$$26.33 + 60m, \quad m = 0, 1, 2\dots$$

as values.

We conclude that the distance is 26.3 meters.

Example

How far out is this valid? Meaning what is the distance before it overlaps again?

To find where the wavelengths give the same value, set

$$1.398 + 25n = 26.389 + 60m,$$

and obtain

$$m = 5(n - 1)/12.$$

We thus need $5(n - 1)/12$ to be an integer for these two to agree.

Also, one wants the smallest value. This will allow one to determine the region that you can determine distance.

Step up the values: $n = 0, 1, 2, 3\dots$ and see when you get an integer for m .

Example

```
#include<stdio.h>

int main()
{
    int i,j;
    float x;

    for(i=0; i<20; i++)
    {
        x = 5.0*(i-1)/12;
        printf("%d,    %f\n", i, x);
    }
}
```

1,	0.000000
2,	0.416667
3,	0.833333
4,	1.250000
5,	1.666667
6,	2.083333
7,	2.500000
8,	2.916667
9,	3.333333
10,	3.750000
11,	4.166667
12,	4.583333
13,	5.000000

Example

So $m = 5$.

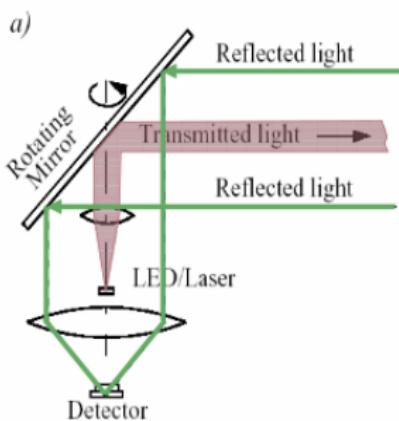
This gives isolation out to about 165 meters using two waves with a much shorter wavelength.

Modulation and phase shift?

How about trying a PWM unit and some gpio.

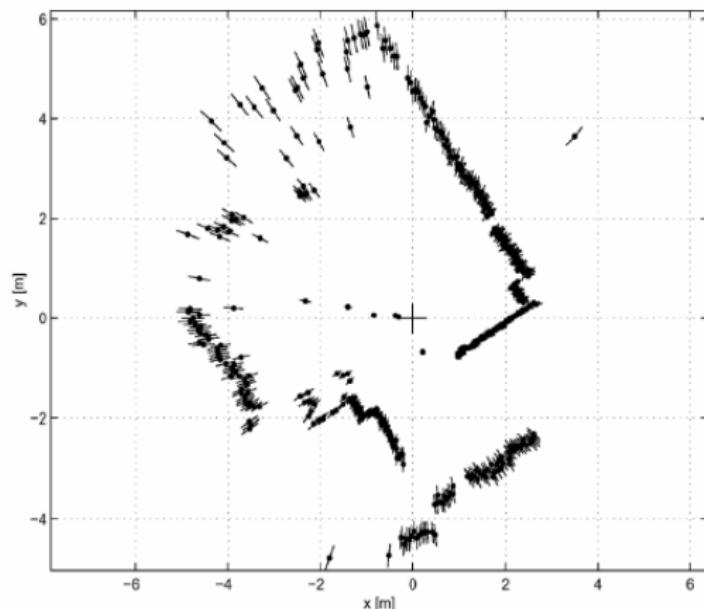
LIDAR Hardware

- ▶ Confidence in the range (phase/time estimate) is inversely proportional to the square of the received signal amplitude.
- ▶ Dark distant objects do not produce as good of range estimate as closer brighter objects.



LIDAR Map Construction

Typical range image of a 2D laser range sensor with a rotating mirror. The length of the lines through the measurement points indicate the uncertainties.

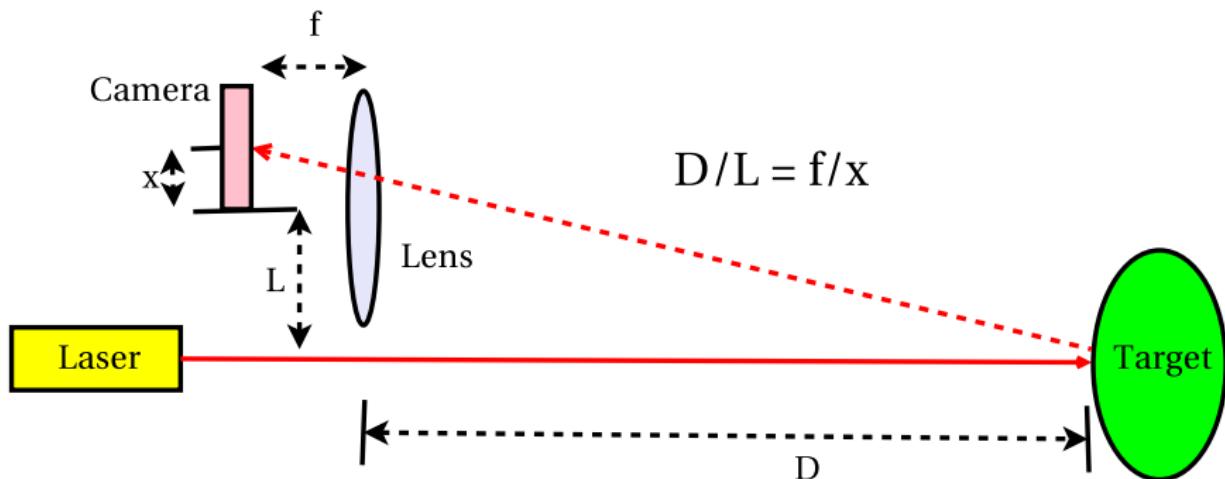


Triangulation

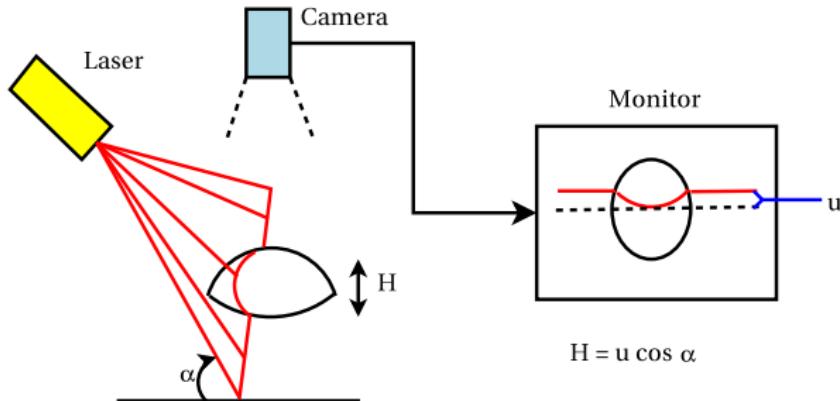
Geometrical Properties of the image or object used to establish a distance measurement.

- ▶ Project a well defined light pattern (points, lines) onto the environment
 - ▶ reflected light is then captured by a photo-sensitive line or matrix (camera) sensor device
 - ▶ simple triangulation allows computation of distance
 - ▶ the standard IR range sensor uses this approach
 - ▶ Kinect and ASUS sensors use arrays of project IR dots - size of dots in IR image indicates distance
- ▶ size of object known
 - ▶ triangulation can be done without projecting light
 - ▶ computer vision techniques can recover relative image size

Laser Triangulation



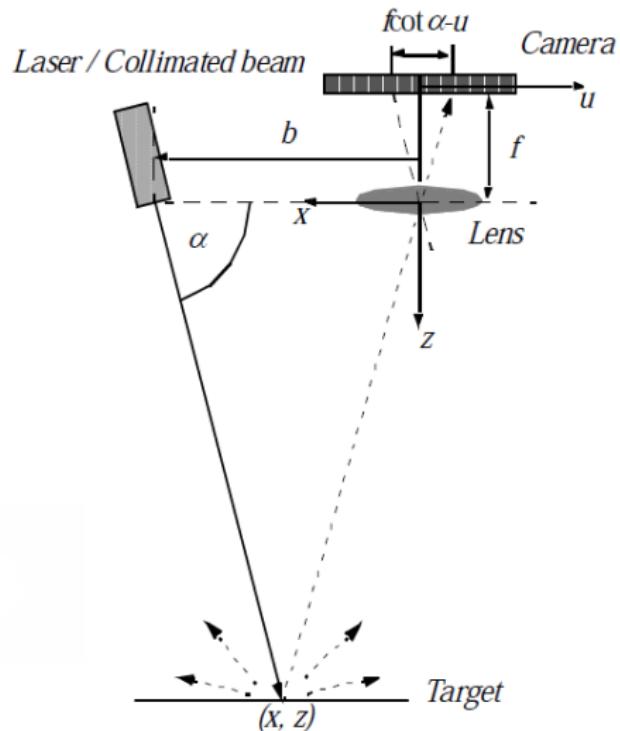
Structured Light



- ▶ Eliminate the correspondence problem by projecting structured light on the scene.
- ▶ Light stripes (laser via rotating mirror)
- ▶ Camera records stripes
- ▶ Range to object can be determined by geometry

Structured Light Optics

Common Industrial Computer Vision Setup



Structured Light Optics

Look at the diagram and see what formulas can we derive. Note that:

$$z/x = f/u, \quad \text{and} \quad \tan(\alpha) = z/(b - x).$$

Flip the second formula:

$$\cot(\alpha) = (b - x)/z.$$

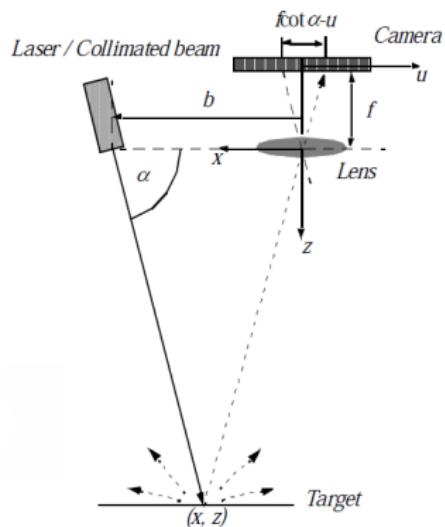
Then multiply by z :

$$z \cot(\alpha) = b - x$$

and move the x over:

$$z \cot(\alpha) + x = b.$$

From the first ratio: $z = fx/u$.



Structured Light Optics

Plug this in for z :

$$(fx/u) \cot(\alpha) + x = b.$$

Factor out the x and divide the rest over:

$$x = \frac{b}{(f/u) \cot(\alpha) + 1} \Rightarrow x = \frac{bu}{f \cot(\alpha) + u}.$$

Then using

$$z = (f/u)x = \left(\frac{f}{u}\right) \frac{bu}{f \cot(\alpha) + u} \Rightarrow z = \frac{bf}{f \cot(\alpha) + u}.$$

Structured Light Optics

What are x & z if $b = 20\text{cm}$, $f = 2\text{cm}$, $\alpha = 60\text{deg}$, and $u = 7\text{mm}$?

So, using these formulas:

$$x = 20 * 0.7 / (2 \cot(60) + 0.7) = 7.55\text{cm},$$

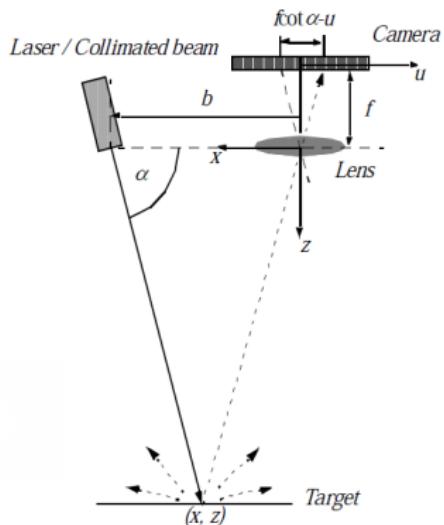
$$z = 20 * 2 / (2 \cot(60) + 0.7) = 21.57\text{cm}.$$

Structured Light Optics

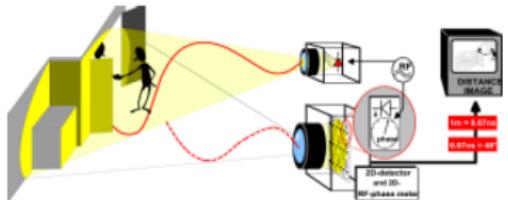
Note that we obtain

$$x = \frac{b \cdot u}{f \cot \alpha + u}$$

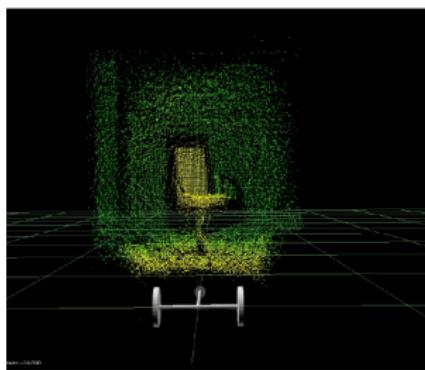
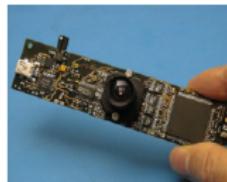
$$z = \frac{b \cdot f}{f \cot \alpha + u}$$



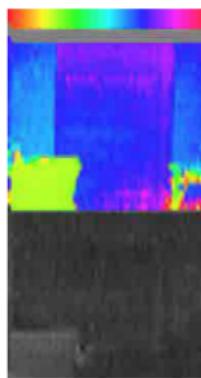
3D Camera



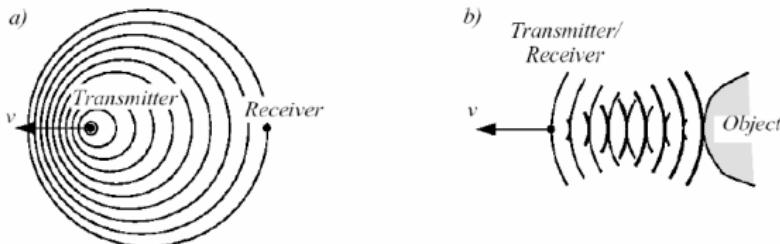
Swiss Ranger (CSEM)



Video CSEM



Measuring Velocity



- ▶ $f_r = f_t(1 + v/c)$ if the transmitter is moving
- ▶ $f_r = f_t/(1 + v/c)$ if the receiver is moving
- ▶ $\Delta f = f_t - f_r = (2f_t v \cos \theta)/c$
- ▶ θ is relative angle between direction of motion and beam axis.

Filtering

Determination of the robot state.

Filtering

The question we ask here is ... what is the current status of the robot?

What are all the values of all the relevant parameters for the robot's relation to the environment?

Robot environment

Dynamical system with state.

$$x_k = f(x_{k-1}, x_{k-2}, \dots)$$

However, we cannot directly view the (true) state.

- ▶ State: x_k , given by the pose (positions, angles, velocities etc)
- ▶ Complete state: if it is the best predictor of the future.
- ▶ Markov process: if the current state of something x_k is given by x_{k-1}

$$x_k = f(x_{k-1})$$

- ▶ Incomplete state: missing information
- ▶ Environmental data: z_k , $z_{t_1:t_2}$ - measurements from t_1 to t_2 .
- ▶ Control data: u_k (one action per step)

State variables

For any state variable we have

- ▶ True value: y
 - ▶ Measured value: \tilde{y}
 - ▶ Estimated value: \hat{y}
-
- ▶ The true value is not known. It is what we seek.
 - ▶ Measured value comes from the sensor. Subject to error as listed above.
 - ▶ Estimated value comes from measurement and system model.

Errors

Basic errors that are used:

- ▶ Measurement error: $v = y - \tilde{y}$
- ▶ Residual error: $e = \tilde{y} - \hat{y}$

We don't know v obviously. The residual error, e , is based on a model of the system and is known explicitly.

Basic error types: we are concerned with two fundamental error types

- ▶ Systematic error - deterministic
- ▶ Random error - non-deterministic

Errors

Systematic errors

- ▶ Incorrectly mounted sensor
- ▶ Blocked sensor
- ▶ Sensor biased by hardware
- ▶ Sampling issues
- ▶ Resolution issues
- ▶ Incomplete measurements
- ▶ Sensitivity
- ▶ Nonlinearity

Random errors

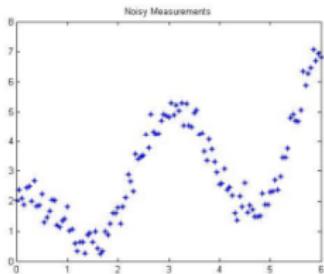
- ▶ Based on white or Gaussian noise
- ▶ Actually could be any distribution, but Gaussian is standard.

Filtering in more Detail

Use the measurement PLUS the model to provide a better estimate of the state.

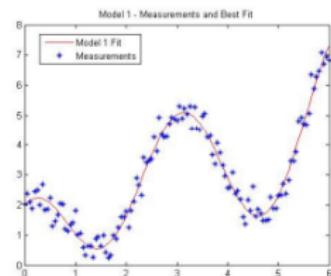
State Estimation = Measurement + Model

Filtering



MEASUREMENT

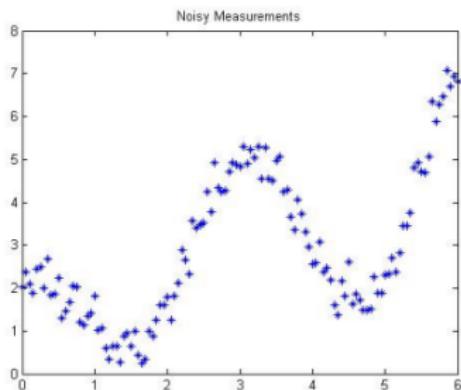
$$y(t) = x_1 t + x_2 \sin(t) + x_3 \cos(2t)$$



MODEL

ESTIMATE/PREDICT

Filter Example



$$\text{Model: } y(t) = x_1 t + x_2 \sin(t) + x_3 \cos(2t)$$

$$\text{Basis Functions: } H = [t \quad \sin(t) \quad \cos(2t)]$$

Using least squares we find the coefficients:
 $(\hat{x}_1, \hat{x}_2, \hat{x}_3) = [0.9967 \quad 0.9556 \quad 2.0030]^T$

Filter issues

Requires a model!!

This is the hard part. It is where the Kalman Filter enters.

The Kalman filter uses a linear time stepping model and environmental data to improve state estimation.

Filters

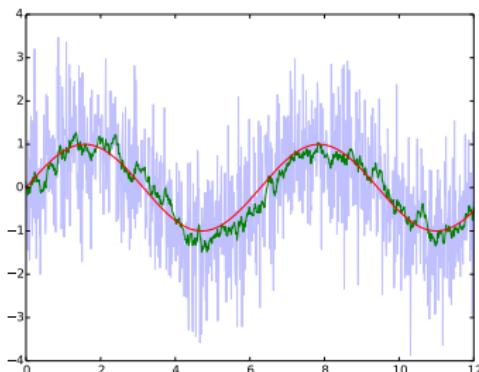
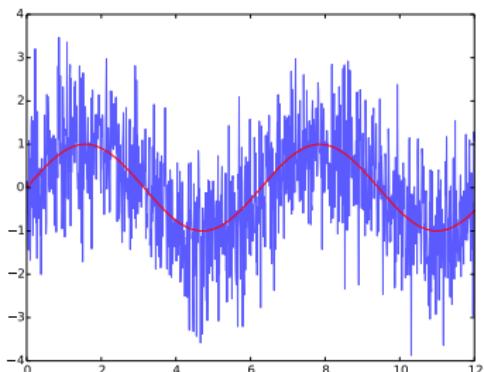
What does one mean by filter?

In this case we are attempting to filter out noise. Simple filters in signal processing often filter in the frequency domain.

For example filtering out high frequencies since this is often noise. We can filter out noise by fitting the data to a model. We assume that the data represents a constant and so we can compute the mean of the data.

This model can also be represented by the distribution that the data appears to have come from, e.g. a normal distribution.

Low Pass Filters

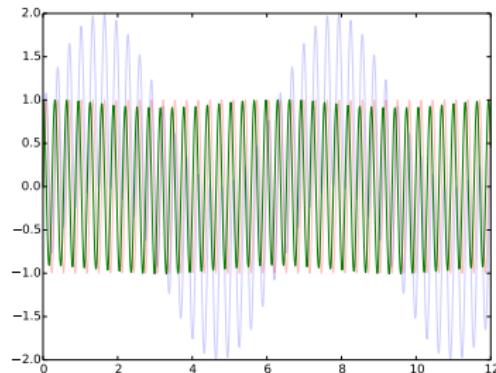
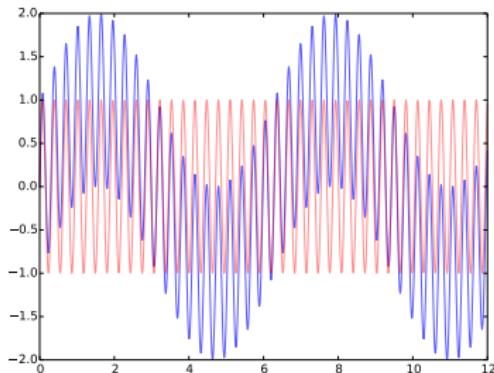


Left: Signal in red, noisy version of the signal in blue.

Right: Noisy signal in blue, filtered signal in green.

```
for i from 1 to n
    y[i] := y[i-1] + a * (z[i] - y[i-1])
```

High Pass Filters

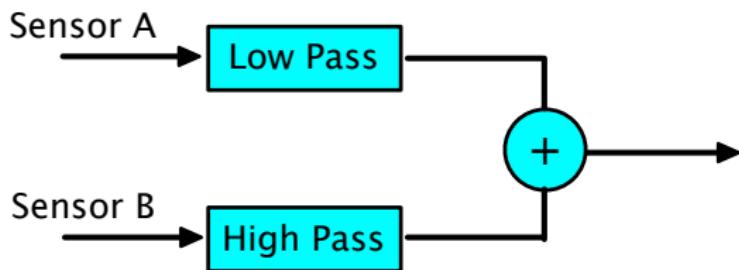


Left: Signal in red, noisy version of the signal in blue.

Right: Noisy signal in blue, filtered signal in green.

```
for i from 1 to n
    y[i] := a * (z[i] - z[i-1])
```

Complementary Filters



Simple Example of Sensor Fusion

Consider a system with n sensors each making a single measurement:

$$z_i, \quad i = 1, \dots, n$$

of some unknown quantity x . The measurements really are described by

$$z_i = x + v_i, \quad i = 1, \dots, n$$

We seek an optimal estimate of x based on a linear combination of these measurements:

$$\hat{x} = \sum_{i=1}^n k_i z_i$$

How?

Simple Example of Sensor Fusion

For this example, we will assume that the noise v_i is zero mean white noise (normally distributed) and independent. This implies that

$$E(v_i) = 0, \quad \text{and} \quad E(v_i v_j) = 0, \quad i \neq j,$$

where $E(x)$ is the expected value of x .

We want the estimate to be unbiased which means that $E(\hat{x} - x) = 0$.

Lemma 1: $E[\hat{x} - x] = 0$ for $\hat{x} = \sum_{i=1}^n k_i z_i$ implies $\sum_{i=1}^n k_i = 1$

Proof in text.

Simple Example of Sensor Fusion

We define optimality as minimizing the mean square error:

$$E[(\hat{x} - x)^2]$$

Lemma 2:

$$E[(\hat{x} - x)^2] = \sum_{i=1}^n k_i^2 \sigma_i^2$$

where σ_i are the standard deviations for v_i , $E((v - E(v))^2) = \sigma^2$.

Proof in text.

Simple Example of Sensor Fusion

We want to minimize the mean square error subject to the constraint of having the unbiased estimate:

- ▶ Minimize $\sum_{i=1}^n k_i^2 \sigma_i^2$ minimize mean square error
- ▶ Subject to $\sum_{i=1}^n k_i = 1$ unbiased estimate

Using Lagrange Multipliers

$$L = \sum_{i=1}^n k_i^2 \sigma_i^2 - \lambda \left(\sum_{i=1}^n k_i - 1 \right)$$

we must solve

$$\nabla_k L = 0 \quad \text{with} \quad \sum_{i=1}^n k_i = 1$$

Simple Example of Sensor Fusion

Thus

$$\nabla L = [2k_1\sigma_1^2 - \lambda, 2k_2\sigma_2^2 - \lambda, \dots, 2k_n\sigma_n^2 - \lambda] = \vec{0}$$

$$\sum_{i=1}^n k_i = 1$$

Solve for k_i in each gradient equation and sum

$$\sum_{i=1}^n k_i = \sum_{i=1}^n \frac{\lambda}{2\sigma_i^2} = 1$$

So, we have that

$$\lambda = \left(\sum_{i=1}^n \frac{1}{2\sigma_i^2} \right)^{-1}$$

Simple Example of Sensor Fusion

This provides k_i

$$k_i = \frac{1}{2\sigma_i^2} \left(\sum_{i=1}^n \frac{1}{2\sigma_i^2} \right)^{-1}$$

and we obtain the estimate

$$\hat{x} = \frac{\sum_{i=1}^n \frac{z_i}{\sigma_i^2}}{\sum_{i=1}^n \frac{1}{\sigma_i^2}} = \frac{1}{S} \sum_{i=1}^n s_i z_i$$

where $s_i = 1/\sigma_i^2$ and $S = \sum_{i=1}^n \frac{1}{\sigma_i^2}$

Simple example ...

If the variances are the same, $\sigma_i = \sigma$, then

$$\sum_{i=1}^n \frac{1}{\sigma_i^2} = \frac{1}{\sigma^2} \sum_{i=1}^n 1 = \frac{n}{\sigma^2}$$

and so

$$\hat{x} = \frac{\frac{1}{\sigma^2} \sum_{i=1}^n z_i}{\frac{n}{\sigma^2}} = \frac{1}{n} \sum_{i=1}^n z_i$$

which is the average.

Simple example - specifics

Say you measure something three different ways and you want to merge these measurements into a single estimate. How does specifically go about it.

Assume that the three devices do return normally distributed measurements. But what is the actual distribution?

For normal distributions, we only need the mean and standard deviation.

The mean:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

The standard deviation:

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \mu)^2}$$

(why $n - 1$?)

Simple example - specifics

2.28333	1.87365	2.12419
2.26493	1.77675	1.80968
2.33626	1.85706	2.00608
2.13676	1.83520	2.12145
2.24820	1.88644	2.22002
2.06169	1.88376	1.74890
2.28497	1.88709	2.00458
2.26484	1.80524	1.96498
2.29407	1.89292	2.29052
2.17190	1.80001	2.15290
2.18591	1.83913	2.35088
2.18288	1.78332	1.83405
2.06597	1.84840	2.17884
2.15693	1.89227	1.97303
2.04323	1.88291	1.96533
2.38279	1.87482	1.86713
2.14289	1.86792	1.86616
2.21151	1.88855	2.20027
2.17112	1.95257	1.77513
2.19798	1.82083	2.25617

Means:

2.20548 1.85962 2.04204

Standard Deviations:

0.08698 0.04282 0.17674

More example

$$P_i(x|\mu, \sigma) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(x - \mu_i)^2}{2\sigma_i^2}}$$

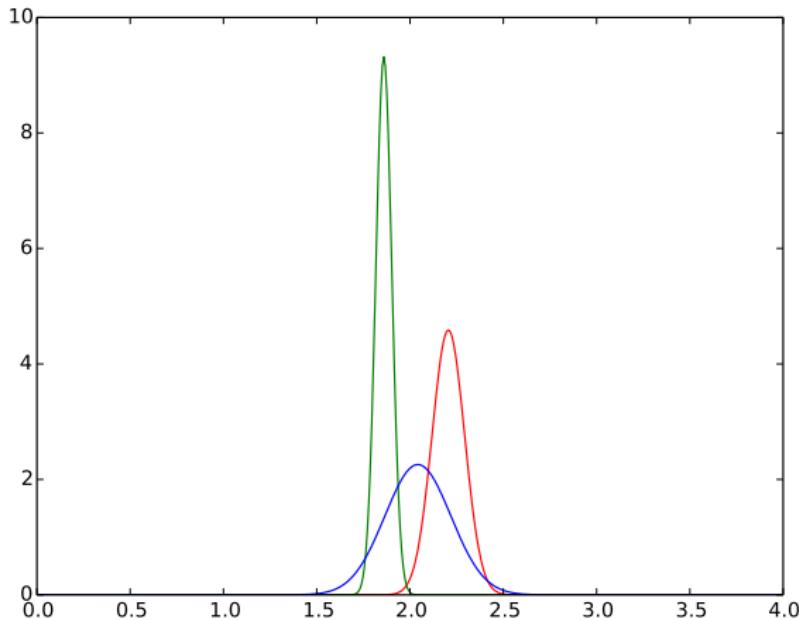
The distributions above are

$$P_1(x) = \frac{1}{0.08698 \sqrt{2\pi}} e^{-\frac{(x - 2.20548)^2}{2(0.08698)^2}}$$

$$P_2(x) = \frac{1}{0.04282 \sqrt{2\pi}} e^{-\frac{(x - 1.85962)^2}{2(0.04282)^2}}$$

$$P_3(x) = \frac{1}{0.17674 \sqrt{2\pi}} e^{-\frac{(x - 2.04204)^2}{2(0.17674)^2}}$$

Example Normal Curves

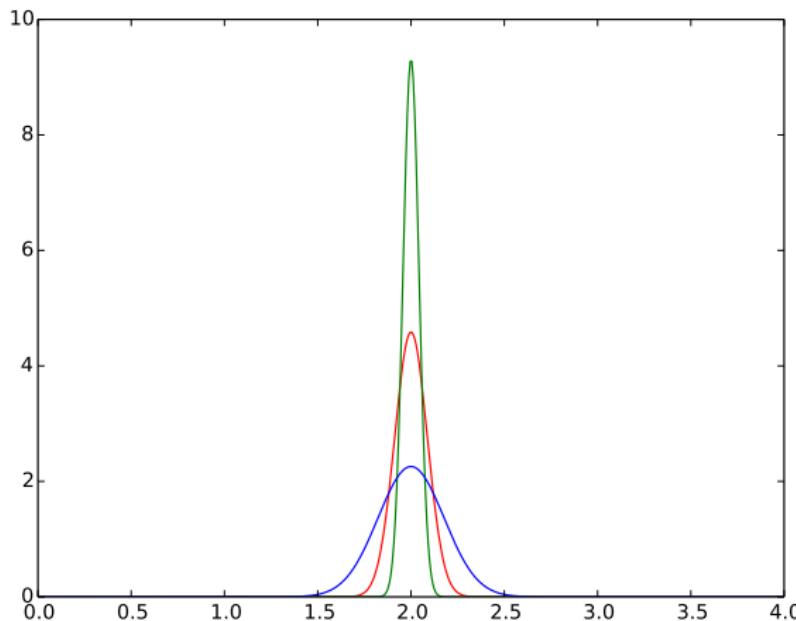


Shifting the curves

x shift (add) = -0.205476607108

y shift (add) = 0.140376647675

z shift (add) = -0.0420388951565



Estimation

Measurement:

Sensor A measures distance at 2.22685 meters

Sensor B measures distance at 1.90326 meters

Sensor C measures distance at 2.17253 meters

The corrected measurements for sensors A, B and C:

$$z_1 = 2.02137, z_2 = 2.04363, z_3 = 2.13049$$

$$\hat{x} = \frac{\sum_{i=1}^n \frac{z_i}{\sigma_i^2}}{\sum_{i=1}^n \frac{1}{\sigma_i^2}} = \frac{\frac{2.02137}{0.08698^2} + \frac{2.04363}{0.04282^2} + \frac{2.13049}{0.17674^2}}{\frac{1}{0.08698^2} + \frac{1}{0.04282^2} + \frac{1}{0.17674^2}} = 2.1063$$

Recursive Filtering²

Example: running average

$$\hat{x}_n = \frac{1}{n} \sum_{i=1}^n z_i$$

A new data point provides a new estimate:

$$\hat{x}_{n+1} = \frac{1}{n+1} \sum_{i=1}^{n+1} z_i$$

Pull the last value out of the sum and rework the weight in front of the sum:

$$\hat{x}_{n+1} = \frac{n}{n+1} \left(\frac{1}{n} \sum_{i=1}^n z_i \right) + \frac{1}{n+1} z_{n+1} = \frac{n}{n+1} \hat{x}_n + \frac{1}{n+1} z_{n+1}$$

²This is subtly different than the computer science notion.

Recursive Filtering

$$\begin{aligned} &= \frac{n+1-1}{n+1} \hat{x}_n + \frac{1}{n+1} z_{n+1} = \hat{x}_n - \frac{1}{n+1} \hat{x}_n + \frac{1}{n+1} z_{n+1} \\ &= \hat{x}_n + \frac{1}{n+1} (z_{n+1} - \hat{x}_n) \end{aligned}$$

So we have

$$\hat{x}_{n+1} = \hat{x}_n + K_n (z_{n+1} - \hat{x}_n), \quad K_n = \frac{1}{n+1}$$

Recursive Filtering Example

```
x = 0
n = 1

f = open('data2.txt','r')
for line in f:
    item = line.split()
    z = eval(item[0])
    x = x + (z - x)/(n)
    n = n+1

print x
```

Recursive Filtering II

Example: running weighted average

$$\hat{x}_n = \frac{1}{S_n} \sum_{i=1}^n \frac{z_i}{\sigma_i^2}, \quad S_n = \sum_{i=1}^n \frac{1}{\sigma_i^2}$$

A new data point provides a new estimate:

$$\hat{x}_{n+1} = \frac{1}{S_{n+1}} \sum_{i=1}^{n+1} \frac{z_i}{\sigma_i^2}, \quad S_{n+1} = \sum_{i=1}^{n+1} \frac{1}{\sigma_i^2}$$

Pull the last value out of the sum and rework the weight in front of the sum:

$$\hat{x}_{n+1} = \frac{S_n}{S_{n+1}} \left(\frac{1}{S_n} \sum_{i=1}^n \frac{z_i}{\sigma_i^2} \right) + \frac{1}{S_{n+1}} \frac{z_{n+1}}{\sigma_{n+1}^2} = \frac{S_n}{S_{n+1}} \hat{x}_n + \frac{1}{S_{n+1}} \frac{z_{n+1}}{\sigma_{n+1}^2}$$

Recursive Filtering II

so,

$$\hat{x}_{n+1} = \frac{S_n + \frac{1}{\sigma_{n+1}^2} - \frac{1}{\sigma_{n+1}^2}}{S_{n+1}} \hat{x}_n + \frac{z_{n+1}}{S_{n+1} \sigma_{n+1}^2} = \hat{x}_n + \frac{1}{\sigma_{n+1}^2 S_{n+1}} (z_{n+1} - \hat{x}_n)$$

using

$$S_{n+1} = S_n + \frac{1}{\sigma_{n+1}^2}$$

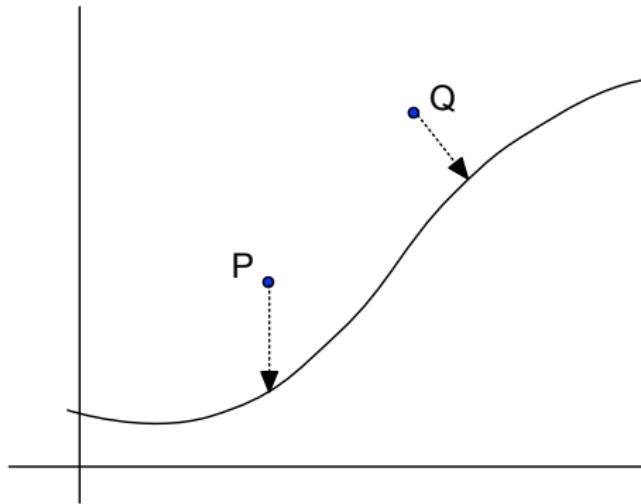
$$\hat{x}_{n+1} = \hat{x}_n + K_{n+1} (z_{n+1} - \hat{x}_n) \quad K_{n+1} = \frac{1}{\sigma_{n+1}^2 S_{n+1}}.$$

You have now seen two important aspects to the Kalman Filter we will derive later. The concept of sensor fusion which is merging data from different distributions and the concept of recursive filtering which follows a Markov formulation.

Filtering with a model

Assume that you have a model and data points P and Q .

We can “filter” by projecting the data onto the model (curve).



The projection can occur at a fixed value of x , P , or nearest point on the curve, Q .

Filtering with a model

Assume the curve is given by $y = f(x)$, $P : (x_1, y_1)$ and $Q : (x_2, y_2)$.

- ① Projecting down: $(x_1, y_1) \rightarrow (x_1, f(x_1))$
- ② Nearest point projecting requires that you solve for x

$$\min_x \sqrt{(x_2 - x)^2 + (y_2 - f(x))^2} \rightarrow (x, f(x))$$

which can be done via

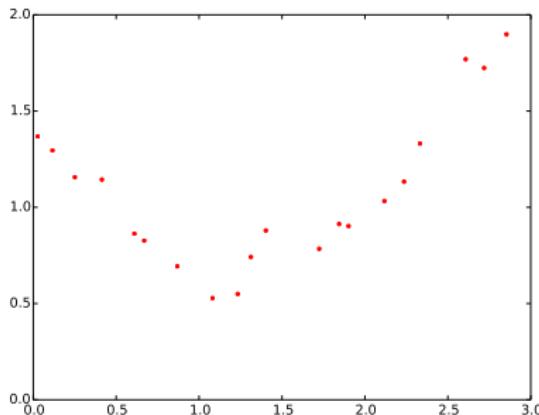
$$\frac{d}{dx} \sqrt{(x_2 - x)^2 + (y_2 - f(x))^2} = 0$$

$$(x_2 - x) + (y_2 - f(x))f'(x) = 0$$

and solve for x .

Filtering with a model

Specifically we can “clean up” the noise in the following data set.



If we have the model then we just project onto it. What if the model is known abstractly, meaning not a specific curve.

We can still proceed by “fitting” the curve to the data.

Least Squares - the Over-Constrained Problem

Say that you have a data set:

$$(x_i, y_i), \quad i = 1, \dots, k.$$

and you want to fit a model to it (project onto a model):

$$y = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0, \quad k \gg n$$

or in general

$$y = a_n \phi_n(x) + a_{n-1} \phi_{n-1}(x) + \dots + a_0 \phi_0(x)$$

How does one use the data to find the coefficients of the model?

Least Squares

Plug the data into the model:

$$y_1 = a_n x_1^n + a_{n-1} x_1^{n-1} + \cdots + a_1 x_1 + a_0$$

$$y_2 = a_n x_2^n + a_{n-1} x_2^{n-1} + \cdots + a_1 x_2 + a_0$$

⋮

$$y_{k-1} = a_n x_{k-1}^n + a_{n-1} x_{k-1}^{n-1} + \cdots + a_{k-1} x_{k-1} + a_0$$

$$y_k = a_n x_k^n + a_{n-1} x_k^{n-1} + \cdots + a_1 x_k + a_0$$

This can be rewritten in the language of linear algebra:

Least Squares

Plug the data into the model:

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix}}_y = \underbrace{\begin{bmatrix} x_1^n & x_1^{n-1} & \dots & x_1 & 1 \\ x_2^n & x_2^{n-1} & \dots & x_2 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ x_k^n & x_k^{n-1} & \dots & x_k & 1 \end{bmatrix}}_X \underbrace{\begin{bmatrix} a_n \\ a_{n-1} \\ \vdots \\ a_1 \\ a_0 \end{bmatrix}}_a$$

The problem is that this system is not usually square and so one cannot just invert the matrix X to find the coefficients a_j .

Least Squares

Expressing our system as

$$y = Xa$$

then formulating the normal equations

$$X^T y = X^T X a$$

we obtain a square system. If $X^T X$ is of full rank, then we can invert

$$a = (X^T X)^{-1} X^T y$$

Once a is found then we may use

$$\hat{y} = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

as a filter.

Least Squares

Before we put this to use, is $X^T X$ of full rank? What does this mean?

The columns must be linearly independent. Only true if we have:

$$\begin{array}{c|c} \begin{array}{|c|c|c|c|} \hline & & & \\ \hline \end{array} & = \\ \hline & \begin{array}{|c|} \hline \end{array} \end{array}$$

It is clear that if we have more rows than columns, the rows cannot be linearly independent. The columns might be L.I.. [If they are not then two of the basis elements $\phi_i(x)$ and $\phi_j(x)$ are the same and we have repeated one.]

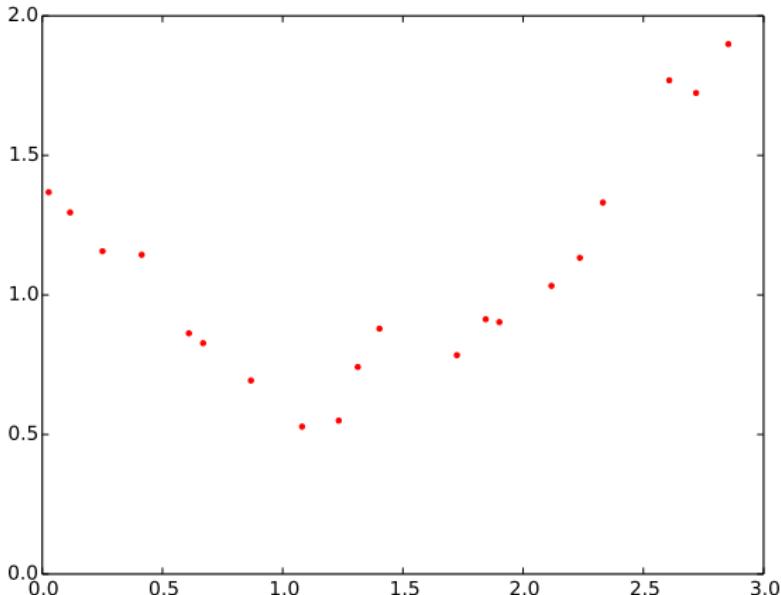
Example

Assume that you have the points:

x_i	y_i
0.026899	1.367895
0.115905	1.295606
0.250757	1.156797
0.413750	1.144025
0.609919	0.862480
0.669044	0.827181
0.868043	0.693536
1.080695	0.528216
1.233052	0.549789
1.312322	0.741778
1.402371	0.879171
1.724433	0.784356
1.844290	0.912907
1.901078	0.902587
2.117728	1.032718
2.235872	1.133116
2.331574	1.331071
2.607533	1.768845
2.719074	1.723766
2.853608	1.898702

x_i y_i

\Rightarrow



Example import from file

```
import numpy as np
import pylab as plt
from scipy import linalg
xl = []
yl = []
f = open('data.txt', 'r')
for line in f:
    item = line.split()
    xt = eval(item[0])
    yt = eval(item[1])
    xl.append(xt)
    yl.append(yt)

plt.plot(x,y, 'ro')
plt.show()
```

Example

Also assume that the model for the data is $y = ax^2 + bx + c$.

Find a, b, c . Note that the system arises:

$$\begin{aligned}(0.026899, 1.367895) \rightarrow \quad & 1.367895 = a(0.026899)^2 + b(0.026899) + c \\(0.115905, 1.295606) \rightarrow \quad & 1.295606 = a(0.115905)^2 + b(0.115905) + c \\(0.250757, 1.156797) \rightarrow \quad & 1.156797 = a(0.250757)^2 + b(0.250757) + c\end{aligned}$$

⋮

which can be written as

$$\begin{bmatrix} (0.026899)^2 & 0.026899 & 1 \\ (0.115905)^2 & 0.115905 & 1 \\ (0.250757)^2 & 0.250757 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1.367895 \\ 1.295606 \\ 1.156797 \\ \vdots \end{bmatrix}$$

Example

The Normal Equations can be formed

$$\begin{bmatrix} (0.026899)^2 & (0.115905)^2 & (0.250757)^2 & \dots \\ 0.026899 & 0.115905 & 0.250757 & \dots \\ 1 & 1 & 1 & \dots \end{bmatrix} \begin{bmatrix} (0.026899)^2 & 0.026899 & 1 \\ (0.115905)^2 & 0.115905 & 1 \\ (0.250757)^2 & 0.250757 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$= \begin{bmatrix} (0.026899)^2 & (0.115905)^2 & (0.250757)^2 & \dots \\ 0.026899 & 0.115905 & 0.250757 & \dots \\ 1 & 1 & 1 & \dots \end{bmatrix} \begin{bmatrix} 1.367895 \\ 1.295606 \\ 1.156797 \\ \vdots \end{bmatrix}$$

Example

One can solve $A^T A u = A^T d$: $u = (A^T A)^{-1} A^T d$

$$\begin{bmatrix} 286.781372 & 122.114670 & 55.443474 \\ 122.114685 & 55.443474 & 28.317947 \\ 55.443474 & 28.317947 & 20.000000 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 71.113922 \\ 33.380650 \\ 21.534540 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \approx \begin{bmatrix} 0.49309569 \\ -1.21285802 \\ 1.42706268 \end{bmatrix}$$

Least Squares Code

```
N = len(xl)
x = np.array(xl)
y = np.array(yl)
xx = x*x
A = np.array([xx, x, np.ones((N))]).T
AT = np.array([xx, x, np.ones((N))])
AA = np.dot(AT, A)
ATy = np.dot(AT, y)

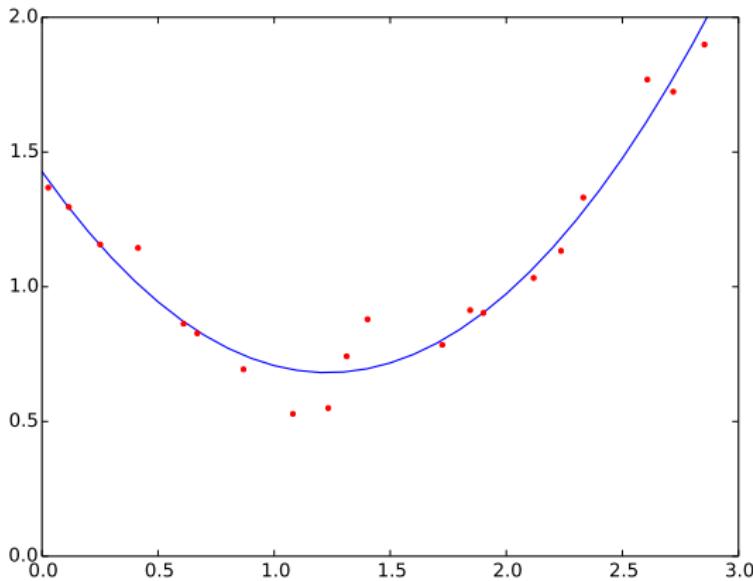
c = linalg.solve(AA, ATy)
t = np.arange(0, 3, 0.1)
tt = t*t
B = np.array([tt, t, np.ones(len(t))]).T
s = np.dot(B, c)
```

Least Squares Code

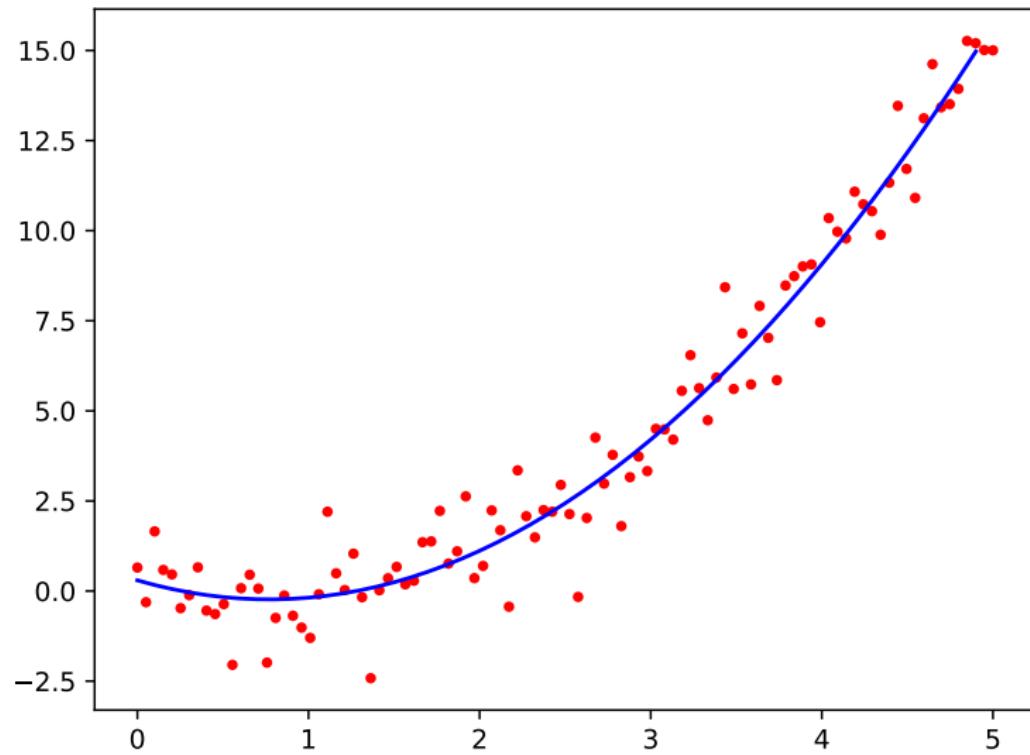
```
plt.plot(t,s, 'b-', x,y, 'ro')
plt.xlim(0,3)
plt.ylim(0,2)
plt.show()
```

Least Squares

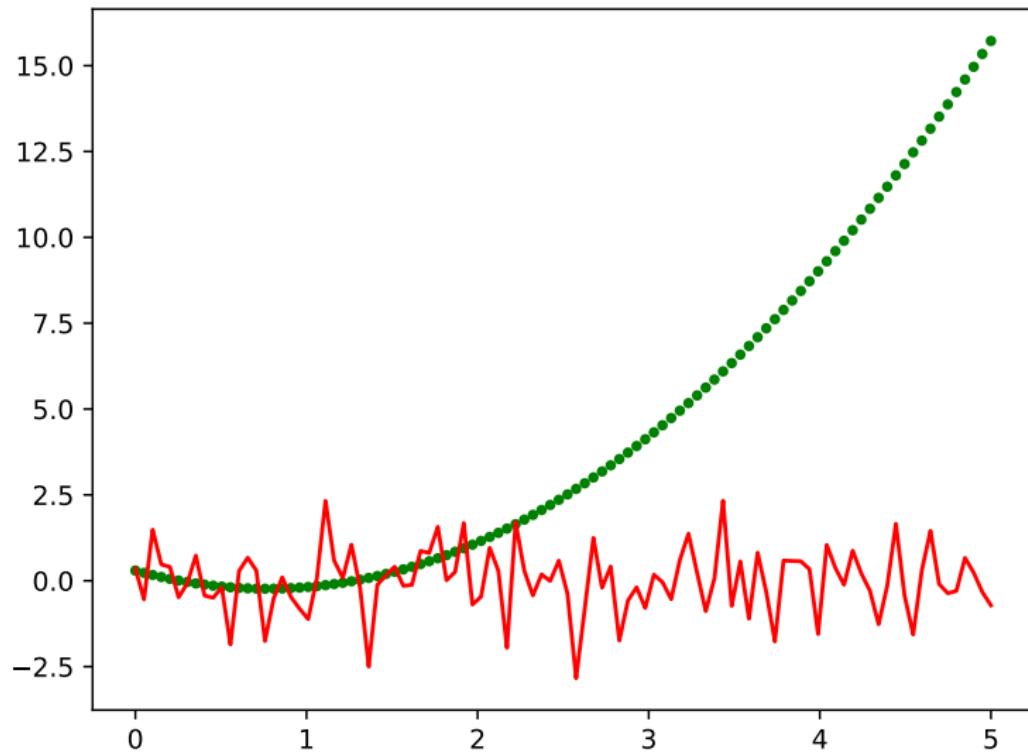
The curve is approximately $y = 0.49x^2 - 1.21x + 1.43$.



Least Squares - Noise

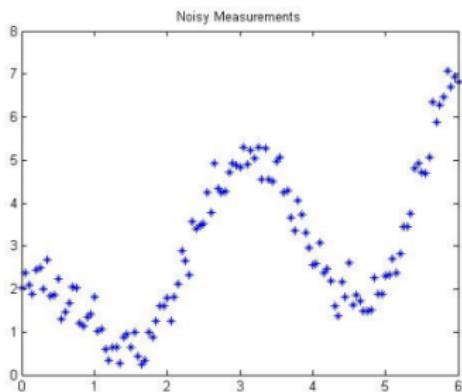


Least Squares - Noise



Least Squares - Noise

Not only do we have many data points and non-square systems, the systems have noise.



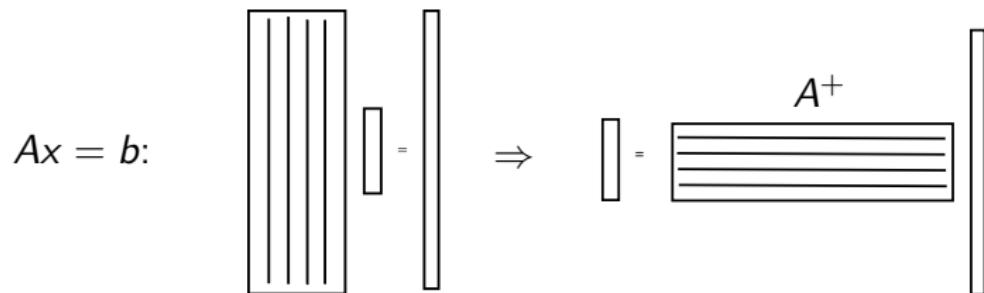
Least Squares is used because there is noise in the data collection and the problems have more data points than parameters.

$$z = x + w$$

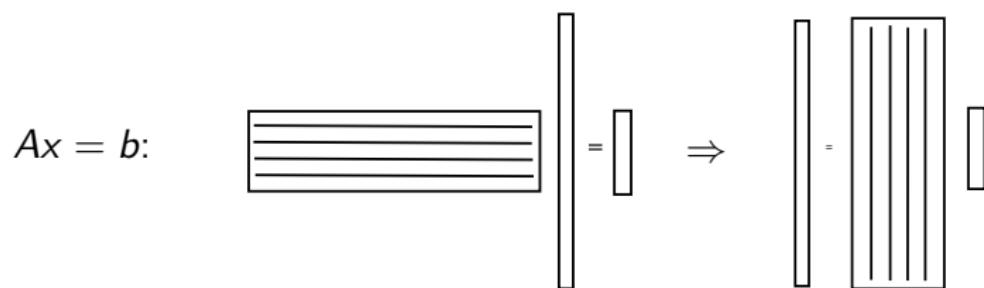
Noise

Moore-Penrose Pseudo-Inverse for $Ax = b$

1. Left Moore-Penrose Pseudo-Inverse: $A^+ = (A^T A)^{-1} A^T$: $A^+ A = I$



2. Right Moore-Penrose Pseudo-Inverse: $A^+ = A^T (AA^T)^{-1}$: $AA^+ = I$



Moore-Penrose Pseudo-Inverse Derivation

- Given $z = Ax$, assume that $(A^T A)$ is full rank.

Multiply both sides of $z = Ax$ by A^T and you can invert:

$$A^T z = A^T A x \rightarrow x = (A^T A)^{-1} A^T z$$

- Given $z = Ax$, assume that (AA^T) is full rank, we have

$$Ax = z \Rightarrow (AA^T) (AA^T)^{-1} Ax = z$$

$$\Rightarrow (AA^T)^{-1} Ax = (AA^T)^{-1} z \Rightarrow Ax = (AA^T) (AA^T)^{-1} z$$

$$\Rightarrow x = A^T (AA^T)^{-1} z$$

Note on Least Squares

When you talked about a Least Squares solution to

$$Ax = b$$

recall that you never did solve for x . You actually found \hat{x} which was the value that minimized

$$\|A\hat{x} - b\|$$

With the curve fitting problem we did not explicitly list the error:

$$Ax + w = b$$

where we had that x was the “true” value. When we wrote $Ax = b$ we really meant find \hat{x} so that $\|A\hat{x} - b\|$ was minimized.

Relation of LS Curve Fitting to Filtering

Given k observations z of state $x \in \mathbb{R}^n$, $k \gg n$, with noise w :

$$z = Hx + w,$$

we aim to find \hat{x} which minimizes the square error:

$$\|z - H\hat{x}\|.$$

This is **exactly** what we have been doing.

$$\hat{x} = (H^T H)^{-1} H^T z$$

The difference between the estimate and the actual value

$$\hat{x} - x = (H^T H)^{-1} H^T (Hx + w) - x = (H^T H)^{-1} H^T w$$

If w has zero mean then $\hat{x} - x$ has zero mean and \hat{x} is referred to as an unbiased estimate.

Recall the Covariance Matrix

Should probably be called Variance-Covariance matrix.

$$\mathbf{X} = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}, \quad \mu_i = E(X_i)$$

$$\Sigma =$$

$$\begin{bmatrix} E[(X_1 - \mu_1)(X_1 - \mu_1)] & E[(X_1 - \mu_1)(X_2 - \mu_2)] & \cdots & E[(X_1 - \mu_1)(X_n - \mu_n)] \\ E[(X_2 - \mu_2)(X_1 - \mu_1)] & E[(X_2 - \mu_2)(X_2 - \mu_2)] & \cdots & E[(X_2 - \mu_2)(X_n - \mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n - \mu_n)(X_1 - \mu_1)] & E[(X_n - \mu_n)(X_2 - \mu_2)] & \cdots & E[(X_n - \mu_n)(X_n - \mu_n)] \end{bmatrix}.$$

Least Squares Observer

The covariance of the estimate error:

$$\begin{aligned}\text{Cov}(\hat{x} - x) &= E[(\hat{x} - x)(\hat{x} - x)^T] = E[\left(H^T H\right)^{-1} H^T w w^T H \left(H^T H\right)^{-1}] \\ &= \left(H^T H\right)^{-1} H^T E[w w^T] H \left(H^T H\right)^{-1}\end{aligned}$$

and define $W = E(w w^T)$, so

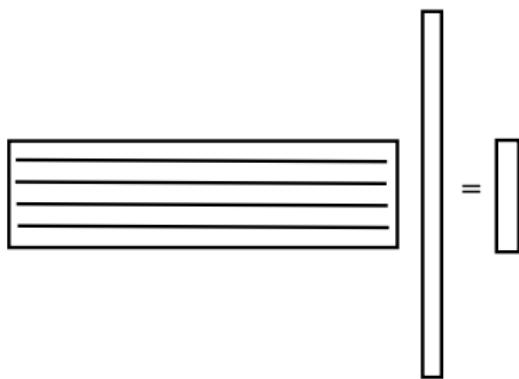
$$\text{Cov}(\hat{x} - x) = \left(H^T H\right)^{-1} H^T W H \left(H^T H\right)^{-1}$$

Least Squares Observer - Under-Constrained Problem

The observation³ z of state x :

$$z = Hx, \quad k < n,$$

has the following structure:



and so the columns are not linearly independent and so $H^T H$ is not invertable. The left sided pseudo-inverse $(H^T H)^{-1}$ does not exist.

³more variables than observations

Observer continued

Given an observation z of state x with noise w :

$$z = Hx + w$$

the \hat{x} which minimizes the square error

$$\|z - H\hat{x}\|$$

$$\hat{x} = H^+ z = W^{-1} H^T \left(H W^{-1} H^T \right)^{-1} z$$

with W the covariance of w and error covariance

$$P = \left(H W^{-1} H^T \right)^{-1}$$

if we take the same weighting as before.

Weighted Least Squares

Traditional least squares is formulated by minimizing using the normal innerproduct:

$$x^T y = \sum_i x_i y_i.$$

If the inner product is weighted:

$$x^T y = \sum_i x_i y_i q_i = x^T Qy$$

then the least squares solution to

$$z = Hx + w$$

is

$$\hat{x} = (H^T Q H)^{-1} H^T Q z$$

Weighted Least Squares

The covariance of this estimate is

$$= \left(H^T Q H \right)^{-1} H^T Q W Q H \left(H^T Q H \right)^{-1}$$

Often one selects the weighting to be inversely proportional to W :

$$Q = W^{-1}$$

[smaller standard deviation means better data, weigh this more]. Thus

$$\hat{x} = \left(H^T W^{-1} H \right)^{-1} H^T W^{-1} z$$

with

$$P = \left(H^T W^{-1} H \right)^{-1}$$

Least Squares Observer

Take a list of measurements which contain noise: “ $z = x + w$ ” where w is noise. The measurements are z_i and the set of equations would be:

The data comes from a noisy observation:

$$z_1 = x + w_1$$

$$z_2 = x + w_2$$

⋮

$$z_n = x + w_n.$$

For explicit noise representation: $z = Hx + w$

But we just have a series of equations:

$$z_1 = x$$

$$z_2 = x$$

⋮

$$z_n = x.$$

and for the curve fitting approach: $z = Hx$

where $z = (z_1, z_2, \dots, z_n)^T$, $w = (w_1, w_2, \dots, w_n)^T$, $H = [1, 1, 1, \dots, 1]^T$

Least Squares Observer

Apply the Least Squares approach to this:

$$\begin{aligned}\hat{x} &= (H^T H)^{-1} H^T z \\ &= \left([1 \quad 1 \quad \dots \quad 1] \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \right)^{-1} \left([1 \quad 1 \quad \dots \quad 1] \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} \right) \\ &= \frac{1}{n} \sum_{i=1}^n z_i\end{aligned}$$

Least Squares Observer - Example 1

Assume that we have two state variables x_1 and x_2 and we are able to observe the first directly (with noise) and the sum of the two (with noise).

This means:

$$z = Hx + w \quad \Rightarrow \quad \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Multiple observations give:

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \\ \vdots \end{pmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \\ \vdots & \vdots \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Least Squares Observer - Example

The solution is

$$\hat{x} = (H^T H)^{-1} H^T z$$

Assume we have data:

0.874328560532

3.25683958794

0.859486711669

2.86834487616

1.25271217589

2.95373764186

0.881013871661

3.09066238259

0.971121996741

3.03754386081

Least Squares Observer - Example

Compute Normal Equation:

$$H^T H = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix} \quad H^T z = \begin{bmatrix} 20.04579167 \\ 15.20712835 \end{bmatrix}$$

Solve $H^T Hx = H^T z$:

Then: $x_1 = 0.96773266$, $x_2 = 2.07369301$

Note that the actual values were $x_1 = 1$, $x_2 = 2$

Least Squares Observer - Example

Assume that we have a noisy data set (x_i, y_i) which we know lies on a line.

```
[[ 0.          -5.65520482]
 [ 0.10204082  4.53774258]
 [ 0.20408163  3.71191423]
 [ 0.30612245  1.44760549]
 [ 0.40816327  0.88024529]
 [ 0.51020408  4.25592703]
 [ 0.6122449   0.81475181]
 [ 0.71428571  0.9275501 ]
 [ 0.81632653  2.70301802]
 [ 0.91836735  5.74002313]
```

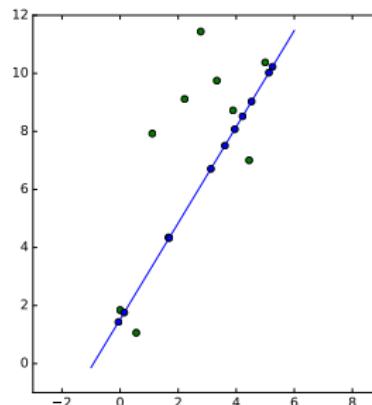
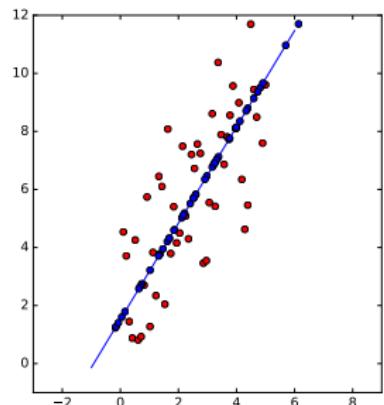
....

The model is $y = a_1x + a_0$. Apply the usual LS process....

We may get something like $a_1 = 2.2231$, $a_0 = 1.0124$,

Least Squares Observer - Example

The Data and Least Squares fit on the left.



New Data and Projection onto the line using this as a filter.

Observer Example

An example of the observer problem...

Say that the system can observe two of three variables: (u, v) from (u, v, θ) ,

$$z_k = Hx_k \quad \Rightarrow \quad \begin{bmatrix} \xi_k \\ \eta_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_k \\ v_k \\ \theta_k \end{bmatrix}$$

The two forms are

$$H^T H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad HH^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

A right pseudo-inverse

$$\begin{bmatrix} u_k \\ v_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} \xi_k \\ \eta_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \xi_k \\ \eta_k \end{bmatrix} = \begin{bmatrix} \xi_k \\ \eta_k \\ 0 \end{bmatrix}$$

Kalman Filter

Linear Dynamical System

An operator, L , is said to be linear if for scalars a, b and vectors x, y we have

$$L(ax + by) = aLx + bLy$$

A dynamical system

$$x_k = Lx_{k-1} \quad (\text{discrete})$$

or

$$\dot{x} = Lx \quad (\text{continuous})$$

is said to be linear if L is a linear operator.

Linearity means we may construct solutions via linear combinations.

Examples of Linear Operators

Let x, y be vectors, u, v be functions and α, β scalars.

A matrix is a linear operator:

$$A(\alpha x + \beta y) = \alpha Ax + \beta Ay$$

The derivative is linear:

$$\frac{d}{dx}(\alpha u + \beta v) = \alpha \frac{du}{dx} + \beta \frac{dv}{dx}$$

The integral is linear:

$$\int (\alpha u + \beta v) dx = \alpha \int u dx + \beta \int v dx$$

Examples of Linear Systems

The Fibonacci Sequence: $a_k = a_{k-1} + a_{k-2}$

Let $x_k = \begin{pmatrix} a_k \\ a_{k-1} \end{pmatrix}$ and $x_{k-1} = \begin{pmatrix} a_{k-1} \\ a_{k-2} \end{pmatrix}$ we have

$$x_k = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} x_{k-1} = L x_{k-1}$$

This is a linear discrete process.

Examples of Linear Systems

Differential equations: $\dot{u} = 2u - v$, $\dot{v} = u + v$.

Let $x(t) = \begin{pmatrix} u(t) \\ v(t) \end{pmatrix}$ and so we have

$$\dot{x} = \begin{pmatrix} 2 & -1 \\ 1 & 1 \end{pmatrix} x$$

Kinematic Models

The dynamics of a differential drive robot

$$\dot{x} = \frac{r}{2}(\dot{\phi}_1 + \dot{\phi}_2) \cos(\theta)$$

$$\dot{y} = \frac{r}{2}(\dot{\phi}_1 + \dot{\phi}_2) \sin(\theta)$$

$$\dot{\theta} = \frac{r}{2L}(\dot{\phi}_1 - \dot{\phi}_2)$$

is **NOT** a linear system.

Linear Gaussian Systems

1. State transition probability $p(x_k|u_k, x_{k-1})$ must arise from

$$x_k = Fx_{k-1} + Gu_k + v_k$$

where x_k , x_{k-1} are state vectors, u_k controls, v_k is the noise, F and G are matrices. v_k is a mean zero normally distributed random variable with covariance matrix V_k .

This is linear system dynamics. Thus the mean of the posterior state is

$$E(x_k) = Fx_{k-1} + Gu_k,$$

$$p(x_k|u_k, x_{k-1}) = \frac{1}{\sqrt{\det(2\pi V_k)}} e^{-\frac{1}{2}(x_k - Fx_{k-1} - Gu_k)^T V_k^{-1} (x_k - Fx_{k-1} - Gu_k)}.$$

Linear Gaussian Systems

2. Measurement Probability $p(z_k|x_k)$ must also be linear

$$z_k = Hx_k + w_k$$

where H is a $m \times n$ matrix and w_k is Gaussian mean zero random variable (noise) with covariance matrix W_k . The mean of the observation

$$E(z_k) = Hx_k,$$

$$p(z_k|x_k) = \frac{1}{\sqrt{\det(2\pi W_k)}} e^{-\frac{1}{2}(z_k - Hx_k)^T W_k^{-1} (z_k - Hx_k)}$$

Linear Gaussian Systems

3. Initial belief, $\text{bel}(x_0)$ must be normally distributed, say with mean \hat{x}_0 and covariance P_0

$$\text{bel}(x_0) = \frac{1}{\sqrt{\det(2\pi P_0)}} e^{-\frac{1}{2}(x_0 - \hat{x}_0)^T P_0^{-1} (x_0 - \hat{x}_0)}$$

If assumptions 1,2,3 hold then $\text{bel}(x_k)$ is also a Gaussian distribution.

Dynamics with Noise

Let x_k be the current state and z_k be the observation.

We study the linear system with noise:

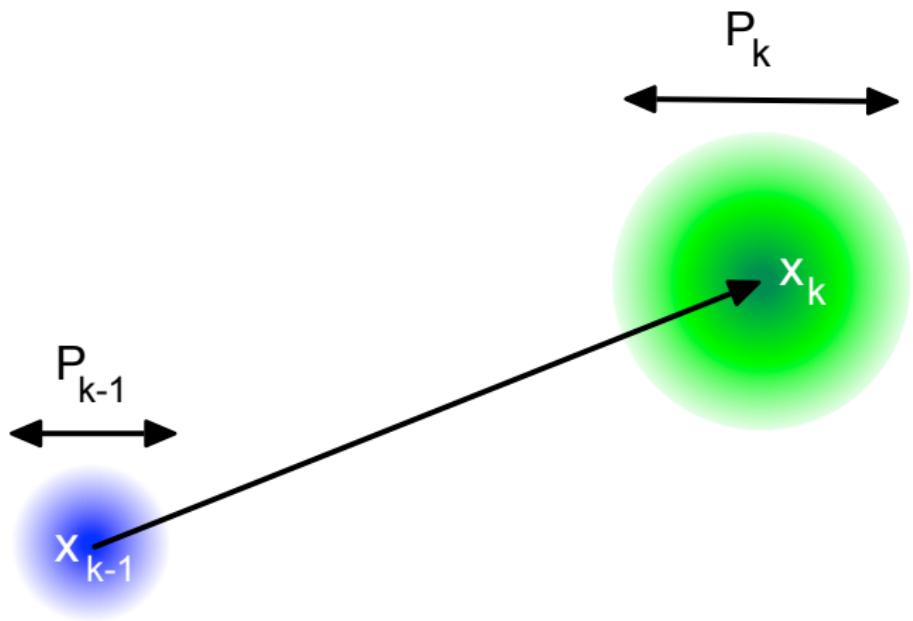
$$x_k = Fx_{k-1} + Gu_k + v_k$$
$$z_k = Hx_k + w_k$$

where v_k , w_k are assumed to be zero mean Gaussian noise with covariance matrices V_k and W_k respectively.

Let $\hat{x}_{k|k}$ represent the mean of x_k and $P_{k|k}$ the covariance of $x_{k|k}$ ($E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T]$).

- ▶ *Prediction:* $\hat{x}_{k|k-1}$, $P_{k|k-1}$
- ▶ *Update:* $\hat{x}_{k|k}$, $P_{k|k}$

Kalman Step



Terminology - details

- ▶ Let V_k be the process noise covariance.
- ▶ Let W_k be the observation noise covariance.
- ▶ Let $\hat{x}_{k-1|k-1}$ be the current state estimate at time $k - 1$.
Note the “hat” indicates that we have an estimate, not the actual value.
- ▶ Let $P_{k-1|k-1}$ be the covariance of the current state $\hat{x}_{k-1|k-1}$
 $(E[(x_{k-1} - \hat{x}_{k-1|k-1})(x_{k-1} - \hat{x}_{k-1|k-1})^T])$

Terminology - details

- ▶ Let $\hat{x}_{k|k-1}$ be the prediction of the next state.
- ▶ Let $P_{k|k-1}$ be the covariance of $\hat{x}_{k|k-1}$
 $(E[(x_k - \hat{x}_{k-1|k-1})(x_k - \hat{x}_{k|k-1})^T])$
- ▶ Let z_k be the observation/measurement.
- ▶ Let $\hat{x}_{k|k}$ be the update based on the observation.
 $\hat{x}_{k|k}$ is our best estimate of x_k
- ▶ Let $P_{k|k}$ be the covariance of $\hat{x}_{k|k}$: $E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T]$

Kalman Filtering: Process Update

- ▶ Predicted state: $\hat{x}_{k|k-1} = \underbrace{F_k \hat{x}_{k-1|k-1}}_{\text{System Dynamics}} + \underbrace{G_k u_k}_{\text{Control}}$
- ▶ Predicted estimate variance: $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + V_k$

What about the measurement? How do we incorporate?

Kalman Filtering: Observation Update

The update formula:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + \underbrace{P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + W_k \right)^{-1}}_{\text{Kalman Gain}} \underbrace{(z_k - H_k \hat{x}_{k|k-1})}_{\text{Innovation}}$$

The covariance update:

$$P_{k|k} = P_{k|k-1} - P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + W_k \right)^{-1} H_k P_{k|k-1}$$

Kalman Filtering - Summary

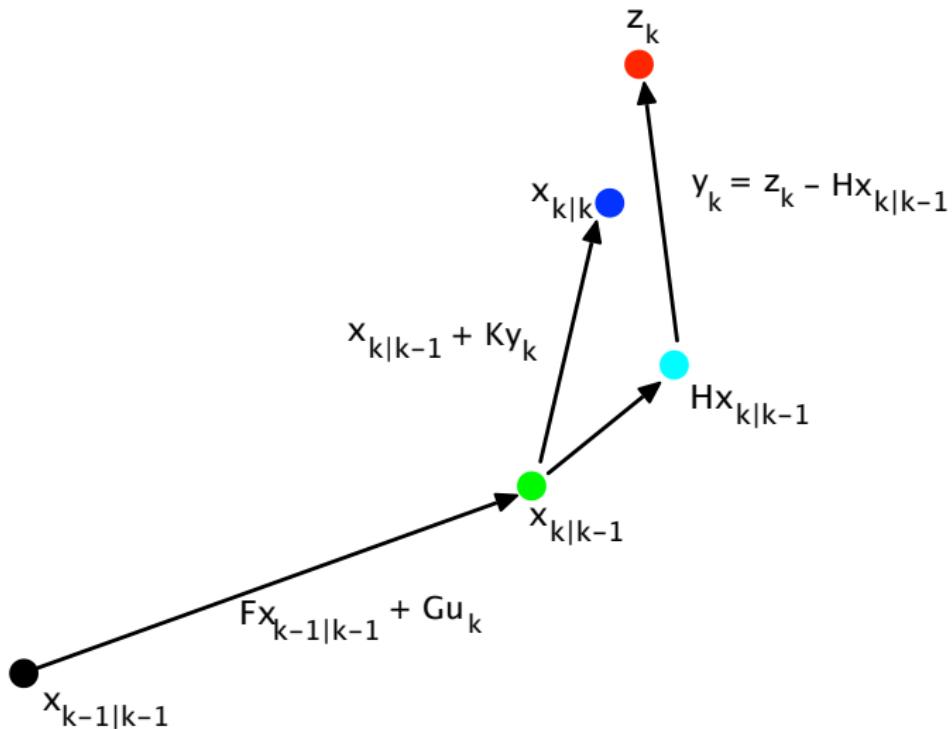
Predict: a priori

- ▶ Predicted state: $\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + G_k u_k$
- ▶ Predicted estimate covariance: $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + V_k$

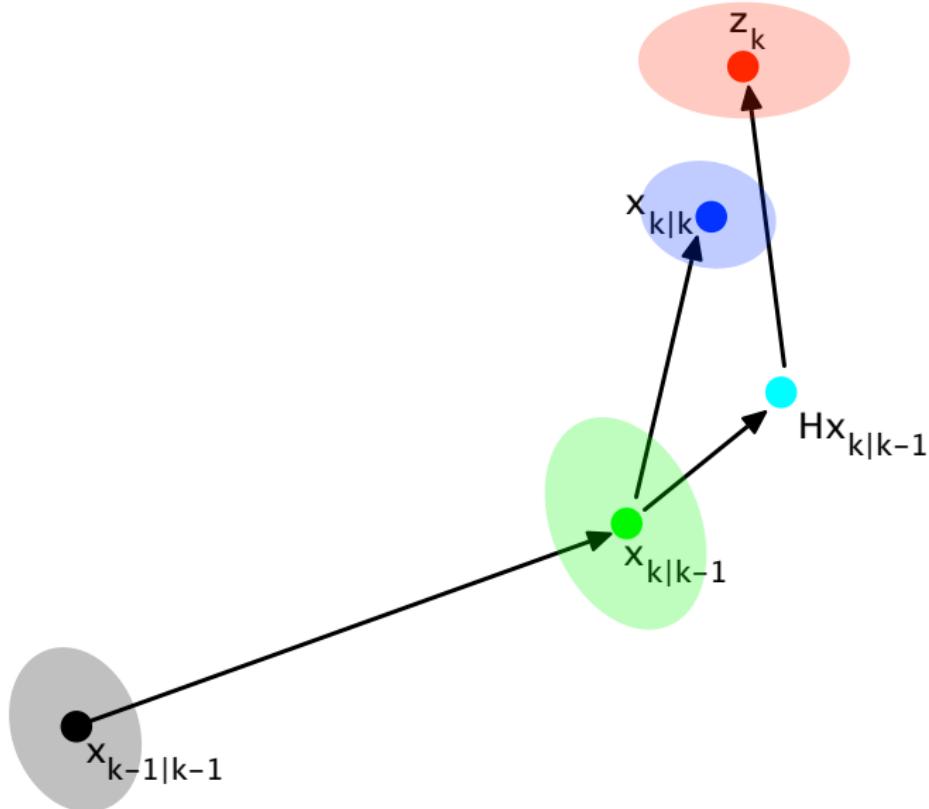
Update: a posteriori

- ▶ Innovation or measurement residual: $y_k = z_k - H_k \hat{x}_{k|k-1}$
- ▶ Innovation (or residual) covariance: $S_k = H_k P_{k|k-1} H_k^T + W_k$
- ▶ Optimal Kalman gain: $K_k = P_{k|k-1} H_k^T S_k^{-1}$
- ▶ Updated state estimate $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k$
- ▶ Updated estimate covariance: $P_{k|k} = (I - K_k H_k) P_{k|k-1}$

Kalman Filtering - Estimate



Kalman Filtering - Estimate Error



Kalman Filter

Input: x_0, P_0

Output: Estimates of x_k, P_k

$k = 0$

while Not Terminated **do**

$k = k + 1$

$$x_k = F_k x_{k-1} + G_k u_k$$

$$P_k = F_k P_{k-1} F_k^T + V_k$$

$$y_k = z_k - H_k x_k$$

$$S_k = H_k P_k H_k^T + W_k$$

$$K_k = P_k H_k^T S_k^{-1}$$

$$x_k = x_k + K_k y_k$$

$$P_k = (I - K_k H_k) P_k$$

end while

Short Matrix Example

Assume that you have the following Gaussian process and observation:

$$\begin{aligned}x_k &= Fx_{k-1} + Gu_k + v_k \\z_k &= Hx_k + w_k\end{aligned}$$

Let

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad F = \begin{bmatrix} 1 & 0 \\ 0.2 & 2 \end{bmatrix}, \quad G = \begin{bmatrix} 0.01 \\ 0 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

$$V = \begin{bmatrix} 0.1 & 0.01 \\ 0.01 & 0.1 \end{bmatrix}, \quad W = 0.25,$$

$$u_k = 0.1, \quad x(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad P(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad z(1) = 2$$

Apply the Kalman Filter process and compute $x(1)$ and $P(1)$.

Simple Example

Process update:

$$\hat{x}_{1|0} = \begin{bmatrix} 1 & 0 \\ 0.2 & 2 \end{bmatrix} \hat{x}_{0|0} + \begin{bmatrix} 0.01 \\ 0 \end{bmatrix} (0.1) = \begin{bmatrix} 1 & 0 \\ 0.2 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.001 \\ 0 \end{bmatrix} = \begin{bmatrix} 1.001 \\ 0.2 \end{bmatrix}$$

Process covariance update:

$$P_{1|0} = FP_{0|0}F^T + V =$$

$$P_{1|0} = \begin{bmatrix} 1 & 0 \\ 0.2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0.2 \\ 0 & 2 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.01 \\ 0.01 & 0.1 \end{bmatrix} = \begin{bmatrix} 1.1 & 0.21 \\ 0.21 & 4.14 \end{bmatrix}$$

Innovation and innovation covariance:

$$y_1 = 2 - [1 \ 0] \hat{x}_{1|0} = 2 - [1 \ 0] \begin{bmatrix} 1.001 \\ 0.2 \end{bmatrix} = 2 - 1.001 = 0.999$$

$$S_1 = HP_{1|0}H^T + W = [1 \ 0] \begin{bmatrix} 1.1 & 0.21 \\ 0.21 & 4.14 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0.25 = 1.35$$

Simple Example

Kalman Gain

$$K_1 = P_{1|0} H_1^T S_1^{-1} = \begin{bmatrix} 1.1 & 0.21 \\ 0.21 & 4.14 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} / 1.35 = \begin{bmatrix} 0.8148 \\ 0.1556 \end{bmatrix}$$

Updated state variables

$$\hat{x}_{1|1} = \hat{x}_{1|0} + K_1 y_1 = \begin{bmatrix} 1.001 \\ 0.2 \end{bmatrix} + \begin{bmatrix} 0.8148 \\ 0.1556 \end{bmatrix} (0.999) = \begin{bmatrix} 1.815 \\ 0.3554 \end{bmatrix}$$

State variable covariance:

$$P_{1|1} = (I - K_1 H_1) P_{1|0} = \left(I - \begin{bmatrix} 0.8148 \\ 0.1556 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} 1.1 & 0.21 \\ 0.21 & 4.14 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1852 & 0 \\ -0.1556 & 1 \end{bmatrix} \begin{bmatrix} 1.1 & 0.21 \\ 0.21 & 4.14 \end{bmatrix} = \begin{bmatrix} 0.2037 & 0.0388 \\ 0.0388 & 4.107 \end{bmatrix}$$

More Information

<https://www.youtube.com/watch?v=0u6ml7iQP70>

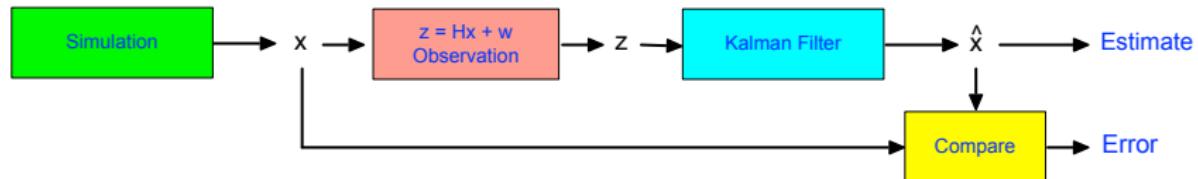
<https://www.youtube.com/watch?v=RXJKdh1KZ0w>

Kalman Filtering - Code Development and Testing

Basic Kalman Code:



Testing the Kalman Code:



Example 2 - Code

Let

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad F = \begin{bmatrix} 0.9 & -0.01 \\ 0.2 & 0.75 \end{bmatrix}, \quad G = \begin{bmatrix} 0.1 \sin(t) \\ 0.05 \cos(t) \end{bmatrix}, \quad H = [1 \quad 0],$$

$$V = \begin{bmatrix} 0.085^2 & 0 \\ 0 & 0.085^2 \end{bmatrix}, \quad W = 0.75^2,$$

$$x(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad P(0) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Run a simulation of the dynamics to create an observation set:

$$x_{k+1} = Fx_k + Gu_k + v_k$$

$$z_{k+1} = Hx_{k+1} + w_{k+1}$$

Code Example: Setup

```
N = 100
t = np.linspace(0, 7, 100)
u1 = 0.1*np.sin(t)
u2 = 0.05*np.cos(t)
mu1, sigma1 = 0.0, 0.085
mu2, sigma2 = 0.0, 0.75
var1 = sigma1*sigma1
x = np.zeros((N,2))
F = np.array([[0.9, -0.01],[0.02,0.75]])
FT = F.T
G = np.array([u1,u2]).T
```

Code Example: Setup

```
H = np.array([1,0])
HT = H.T
V = np.array([[var1,0],[0,var1]])
W = sigma2*sigma2
P = np.zeros((N,2,2))
z = np.zeros(N)
xf = np.zeros((N,2))
```

Code Example: Run Dynamics and Observe

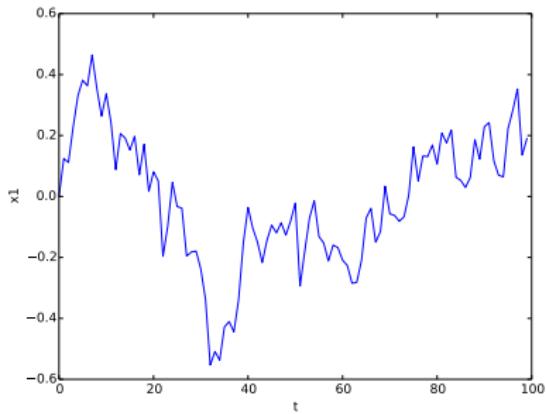
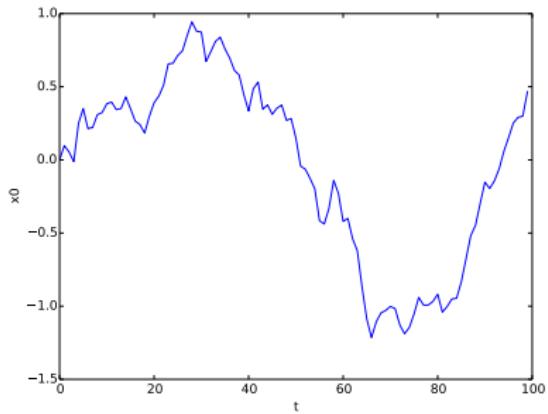
```
k = 1
while (k<N):
    q = np.random.normal(mu1,sigma1,2)
    r = np.random.normal(mu2,sigma2, 1)
    x[k] = np.dot(F,x[k-1]) + G[k-1] + q
    z[k] = np.dot(H,x[k]) + r
    k = k+1
```

Done with fake data.

You don't have to break these into two stages - you could just compute this "on the fly". This way you can capture the z values and save them. Then run this into the next stage if you want the same observation over.

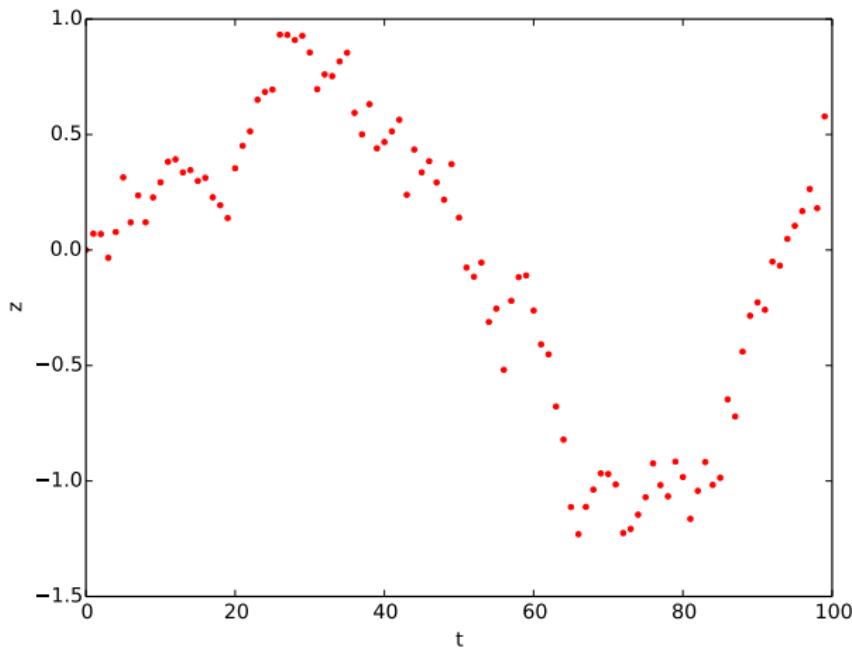
Code Example

The blue line is the actual (hidden) state:



Code Example

The red dots is the observation of the first state:

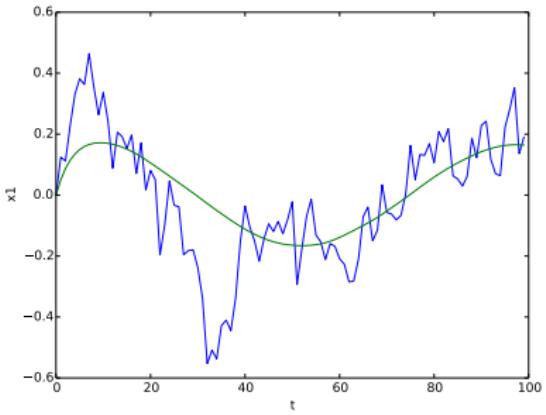
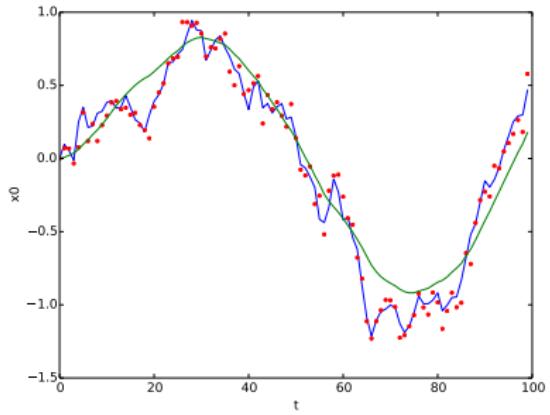


Code Example: Kalman Filter Stage

```
k = 1
while (k<N):
    xp = np.dot(F, xf[k-1]) + G[k-1]
    pp = np.dot(F, np.dot(P[k-1], FT)) + V
    y = z[k] - np.dot(H, xp)
    S = np.dot(H, np.dot(pp, HT)) + W
    kal = np.dot(pp, HT)/S
    xf[k] = xp + y*kal
    P[k] = pp - np.outer(kal, np.dot(H, pp))
    k = k+1
```

Code Example

The green line is the estimate.



Second Matrix Example

Assume that you have the following Gaussian process and observation:

$$x_k = Fx_{k-1} + Gu_k + v_k$$
$$z_k = Hx_k + w_k$$

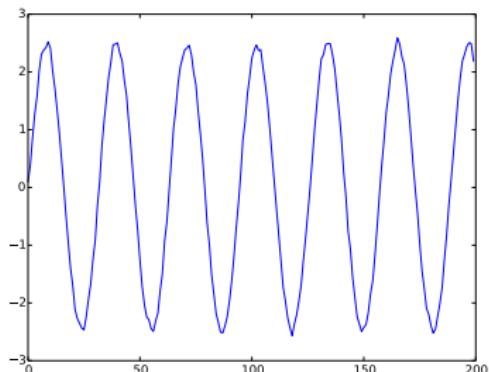
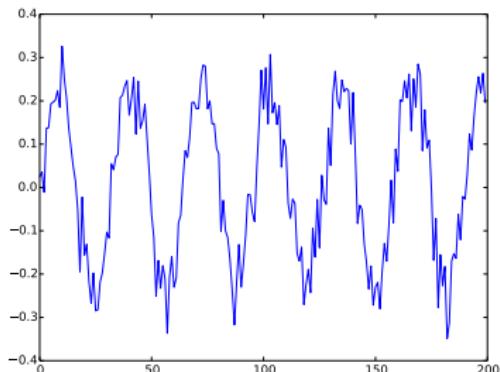
Let

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad F = \begin{bmatrix} 0 & 0.1 \\ -0.02 & 0.2 \end{bmatrix}, \quad G = \begin{bmatrix} 0 \\ 2 \sin(ct) \end{bmatrix}, \quad H = [1 \quad 0],$$

$$V = \begin{bmatrix} 0.0025 & 0 \\ 0 & 0.0025 \end{bmatrix}, \quad W = 0.25^2,$$

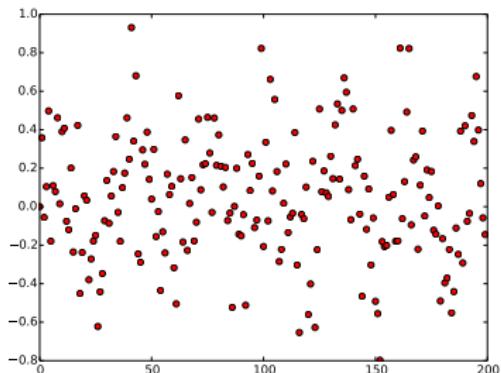
$$x(0) = \begin{bmatrix} 0.025 \\ 0.1 \end{bmatrix}, \quad P(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Second Matrix Example

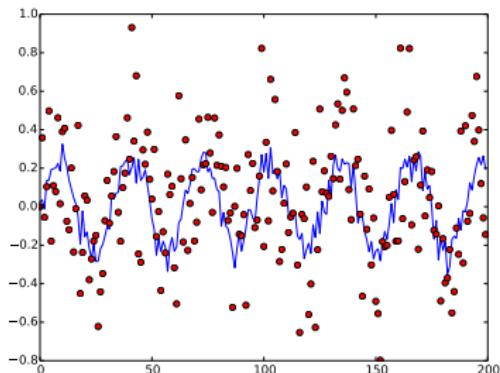


Actual x_0 and x_1 .

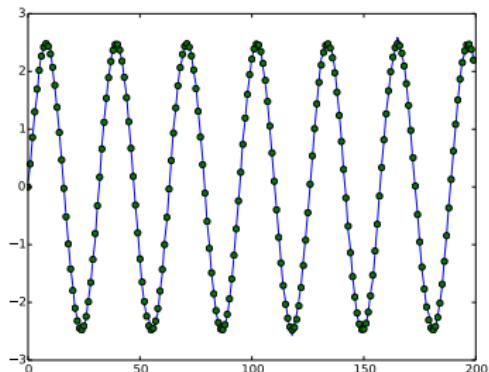
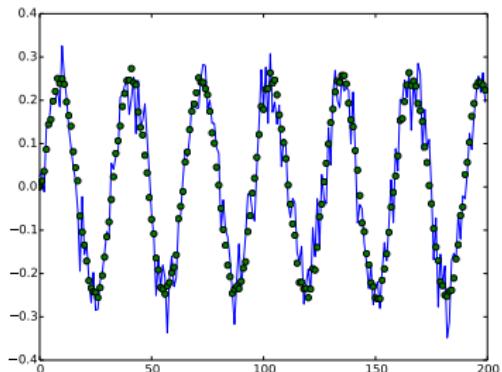
Second Matrix Example



x_0 observation



Second Matrix Example



Blue: actual. Green: estimate.

Third Example

Consider a mobile robot along a track. Let the state $x = [x_r, s_r]$ where x_r and s_r are the vehicle position and speed. Let m denote the mass of the vehicle and u be the force acting on the vehicle. Note that

$$\frac{ds_r}{dt} = \frac{u}{m}$$

Discretize

$$\frac{s_r(t + T) - s_r(t)}{T} \approx \frac{ds_r}{dt}$$

T is the sample rate. Thus

$$s_r(k + 1) = s_r(k) + \frac{T}{m} u(k)$$

and

$$x_r(k + 1) = x_r(k) + Ts_r(k)$$

Load the variables into an array

$$x_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ T/m \end{bmatrix} + v_k$$

Example

Assume that you have some sensors

$$z_{k+1} = [0 \quad 1] x_k + w_k$$

where v and w are zero mean Gaussian noise. Thus

$$F_k = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad G_k = \begin{bmatrix} 0 \\ T/m \end{bmatrix}, \quad H_k = [0 \quad 1]$$

For this example take $m = 1$ and $T = 0.5$. Assume the covariance of v_k

$$V_k = \begin{bmatrix} 0.2 & 0.05 \\ 0.05 & 0.1 \end{bmatrix}$$

Assume the covariance for w_k is $W_k = [0.5]$, and at $k = 0$, $u(0) = 0$ and $\hat{x}_{0|0} = [2 \quad 4]^T$,

$$P_{0|0} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Example

Next we compute one iteration of the Kalman Filter.

- ▶ State estimate prediction:

$$\hat{x}_{1|0} = F_1 \hat{x}_{0|0} + G_1 u_1 = \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} 0 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$$

- ▶ Covariance prediction

$$P_{1|0} = F_1 P_{0|0} F_1^T + V_1$$
$$= \begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0.5 & 1 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.05 \\ 0.05 & 0.1 \end{bmatrix} = \begin{bmatrix} 1.7 & 1.05 \\ 1.05 & 2.1 \end{bmatrix}$$

Assume that you measure and obtain

$$z_1 = 3.8$$

Example

- ▶ Innovation:

$$y_k = z_1 - H\hat{x}_{1|0} = 3.8 - \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 4 \end{bmatrix} = -.2$$

- ▶ The matrix S

$$S_1 = HP_{1|0}H^T + W_1 = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 1.7 & 1.05 \\ 1.05 & 2.1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 0.5 = 2.6$$

- ▶ The matrix K (Kalman Gain)

$$K_1 = P_{1|0}H^TS_1^{-1} = \begin{bmatrix} 1.7 & 1.05 \\ 1.05 & 2.1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} (2.6)^{-1} = \begin{bmatrix} 1.05/2.6 \\ 2.1/2.6 \end{bmatrix} = \begin{bmatrix} 0.404 \\ 0.808 \end{bmatrix}$$

Example

- ▶ The estimate update:

$$\hat{x}_{1|1} = \hat{x}_{1|0} + K_1 y_1 = \begin{bmatrix} 4 \\ 4 \end{bmatrix} + \begin{bmatrix} 0.404 \\ 0.808 \end{bmatrix} (-.2) = \begin{bmatrix} 3.9192 \\ 3.8384 \end{bmatrix}$$

- ▶ The covariance estimate update:

$$\begin{aligned} P_{1|1} &= (I - K_1 H) P_{1|0} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.404 \\ 0.808 \end{bmatrix} \begin{bmatrix} 0 & 1 \end{bmatrix} \right) \begin{bmatrix} 1.7 & 1.05 \\ 1.05 & 2.1 \end{bmatrix} \\ &= \begin{bmatrix} .4242 & .8484 \\ .8484 & 1.6968 \end{bmatrix} \end{aligned}$$

Noise

Assume that the robot moves along according to the kinematic model F and G plus the noise, we have

$$x_{k+1} = Fx_k + Gu_k + v_k$$

This produces the robot path as a vector of values $\{x\}$.

At each step along the computed path, we can make an observation (z_k) which is noise added to the exact values $x_k + v_k$ where v_k is Gaussian noise. Since z_k is not added back into the computation for x_{k+1} , the observation noise, w_k , does not accumulate. The process is the following:

$$x_{k+1} = Fx_k + Gu_k + v_k$$

$$z_{k+1} = Hx_k + w_k$$

These can be computed together.

Noise

```
k = 1
while (k<N):
    q = np.random.normal(mu1,sigma1,2)
    r = np.random.normal(mu2,sigma2, 1)
    x[k] = np.dot(F,x[k-1]) + G[k-1] + q
    z[k] = np.dot(H,x[k]) + r
    k = k+1
```

Noise

Assume that we have two signals

$$a(t) = \cos(t), \quad b(t) = 20 \cos(t)$$

and to them we add mean zero Gaussian noise with standard deviation $\sigma = 0.25$, v :

$$a_1(t) = \cos(t) + v, \quad b_1(t) = 20 \cos(t) + v$$

or we multiply that noise

$$a_2(t) = v \cos(t), \quad b_2(t) = 20v \cos(t)$$

We then subtract off the signal and compute the standard deviations. For a_1 and b_1 , it is mathematically clear that you would get $\sigma = 0.25$ back - if the sample size large enough. What about the product?

Noise

```
>>> c = np.cos(t)
>>> a1 = c + np.random.normal(0, 0.25,100)
>>> b1 = 20*c + np.random.normal(0, 0.25,100)
>>> a2 = np.random.normal(0, 0.25,100)*c
>>> b2 = 20*np.random.normal(0, 0.25,100)*c
>>> a1sub = a1 - c
>>> b1sub = b1 - 20*c
>>> a2sub = a2 - c
>>> b2sub = b2 - 20*c
>>> np.std(a1sub)
0.26168514491592509
>>> np.std(b1sub)
0.20957486503563907
>>> np.std(a2sub)
0.73517338736953186
>>> np.std(b2sub)
14.687819454616823
```

Modifications

- ▶ What does one do if the observation is not available?
Skip the update. Run several state predictions.
- ▶ How can one merge multiple observations from different sensors? (In a single update or if no dynamics exists.)
Skip the state prediction and just run the update steps.

Sensor Fusion

Predict:

- ▶ $\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + G_k u_k$
- ▶ $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + V_k$

Update:

- ▶ foreach sensor i :
 - ▶ $y_k = z_k^i - (H^i)_k \hat{x}_{k|k-1}$
 - ▶ $S_k = (H^i)_k P_{k|k-1} (H^i)_k^T + W_k^i$
 - ▶ $K_k = P_{k|k-1} (H^i)_k S_k^{-1}$
 - ▶ $\hat{x}_{k|k-1} = \hat{x}_{k|k-1} + K_k y_k$
 - ▶ $P_{k|k-1} = (I - K_k (H^i)_k) P_{k|k-1}$
- ▶ $\hat{x}_{k|k} = \hat{x}_{k|k-1}$
- ▶ $P_{k|k} = P_{k|k-1}$

Sensor Fusion

Sensor Fusion via Kalman

- ▶ Innovation (or residual) covariance: $S_k = H_k P_{k-1} H_k^T + W_k$
- ▶ Optimal Kalman gain: $K_k = P_{k-1} H_k^T S_k^{-1}$
- ▶ Updated estimate $\hat{x}_k = \hat{x}_{k-1} + K_k (z_k - H_k \hat{x}_{k-1})$
- ▶ Updated estimate covariance: $P_k = (I - K_k H_k) P_{k-1}$

Sensor Fusion

So we have: Let W_k is the variance for the sensor and $H_k = I$ (full observation).

- ▶ Set $x_0 = z_0$, $P_0 = W_0$
- ▶ Let $k = 1$ and repeat:
 - ▶ $S_k = P_{k-1} + W_k$
 - ▶ $K_k = P_{k-1} S_k^{-1}$
 - ▶ $\hat{x}_k = \hat{x}_{k-1} + K_k (z_k - \hat{x}_{k-1})$
 - ▶ $P_k = (I - K_k)P_{k-1}$

```
while (k<n):  
    y = z[k] - x  
    S = P + W[k]  
    kal = np.dot(P, linalg.inv(S))  
    x = x + np.dot(kal, y)  
    P = P - np.dot(kal, P)  
    k = k+1
```

Information Fusion Example

Say you have three sensors measuring a single state x_k .

How do you apply the previous result?

So, P_k is the variance of \hat{x}_k , W_k is the variance for the sensor and $H_k = 1$:

$k = 1$,

Repeat:

- ▶ $S_k = P_{k-1} + W_k$
- ▶ $K_k = P_{k-1}S_k^{-1}$
- ▶ $\hat{x}_k = \hat{x}_{k-1} + K_k(z_k - \hat{x}_{k-1})$
- ▶ $P_k = (I - K_k)P_{k-1}$

Try $P_0 = W_0$ and $x_0 = z_0$.

Different loop frequencies

Predict:

- ▶ $\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + G_k u_k$
- ▶ $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + V_k$

Update:

- ▶ Loop over available sensor data during Δt :
 - ▶ $y_k = z_k^i - (H^i)_k \hat{x}_{k|k-1}$
 - ▶ $S_k = (H^i)_k P_{k|k-1} (H^i)_k^T + W_k^i$
 - ▶ $K_k = P_{k|k-1} (H^i)_k^T S_k^{-1}$
 - ▶ $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k y_k$
 - ▶ $P_{k|k} = (I - K_k (H^i)_k) P_{k|k-1}$
- ▶ $\hat{x}_{k|k} = \hat{x}_{k|k-1}$
- ▶ $P_{k|k} = P_{k|k-1}$

Kalman Filtering - Numerics

If you have n equations, the work (multiplications) in the filter is:

- ① $\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + G_k u_k : O(n^2)$
- ② $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + V_k : O(n^3)$
- ③ $K_k = P_{k|k-1} H_k^T [H_k P_{k|k-1} H_k^T + W_k]^{-1} : O(m!) + O(n^2 m)$
- ④ $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H_k \hat{x}_{k|k-1}) : O(n^2)$
- ⑤ $P_{k|k} = (I - K_k H_k) P_{k|k-1} : O(n^3)$

The largest work is in step 3. By using an LU factorization, we can move this down to $\max(O(m^3), O(n^2 m))$ work. Step 2 can exploit symmetry to reduce work as only 1/2 the matrix needs to be computed. For small matrices, explicit formulas for the inverse can be used.

Nonlinearity

What does one do in the case where the process or the observation is a nonlinear function?

$$x_k = f(u_k, x_{k-1}) + v_k,$$

$$z_k = h(x_{k-1}) + w_k$$

where v_k has variance V_k and w_k has variance W_k .

Nonlinearity

For now, focus on 2D.

Use a Taylor expansion on f :

$$f_1(x, y) = f_1(x_0, y_0) + \frac{\partial}{\partial x} f_1(x_0, y_0) \Delta x + \frac{\partial}{\partial y} f_1(x_0, y_0) \Delta y + \dots$$

Or ...

$$f_1(x, y) - f_1(x_0, y_0) = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + \dots$$

Thus

$$\begin{bmatrix} \Delta f_1 \\ \Delta f_2 \end{bmatrix} = \begin{bmatrix} f_1(x, y) - f_1(x_0, y_0) \\ f_2(x, y) - f_2(x_0, y_0) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + \dots$$

Nonlinearity

Thus for higher dimensions:

$$x_k = f(u_k, x_{k-1}) + v_k,$$

$$z_k = h(x_{k-1}) + w_k$$

where v_k has variance V_k and w_k has variance W_k , and let

$$F_k = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}, H_k = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \cdots & \frac{\partial h_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \frac{\partial h_m}{\partial x_2} & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix}$$

This is a Taylor expansion approach to dealing with nonlinear mappings.

Extended Kalman Filter

- ① Predicted state:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k)$$

- ② Predicted estimate covariance:

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + V_k$$

- ③ Optimal Kalman gain:

$$K_k = P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + W_k \right)^{-1}$$

- ④ Updated state estimate:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - h(\hat{x}_{k|k-1}))$$

- ⑤ Updated estimate covariance:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Short Example

What is the Extended Kalman Filter formulation of the motion model:

$$\begin{aligned}\dot{x} &= y \\ \dot{y} &= -\cos(x) + k \sin(t)\end{aligned},$$

and observation

$$h(x, y) = \begin{bmatrix} x \\ y \end{bmatrix}$$

with step size $\Delta t = 0.1$, and noise

$$V = \begin{bmatrix} 0.1 & 0.01 \\ 0.01 & 0.1 \end{bmatrix}, \quad , W = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}.$$

Short Example

Using a basic Euler formulation we replace the derivative:

$$\frac{x_{n+1} - x_n}{0.1} = y_n$$

$$\frac{y_{n+1} - y_n}{0.1} = -\cos(x_n) + 0.4 \sin(t_n)$$

This gives the discrete form:

$$x_{n+1} = x_n + 0.1y_n$$

$$y_{n+1} = y_n - 0.1 \cos(x_n) + 0.04 \sin(t_n)$$

and so

$$F = \begin{bmatrix} 1 & 0.1 \\ 0.1 \sin(x_n) & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Short Example

- ① Predicted state:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k)$$

- ② Predicted estimate covariance:

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + V_k$$

- ③ Optimal Kalman gain:

$$K_k = P_{k|k-1} H_k^T \left(H_k P_{k|k-1} H_k^T + W_k \right)^{-1}$$

- ④ Updated state estimate:

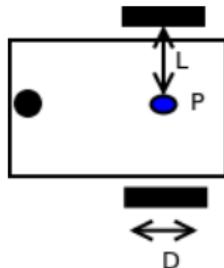
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - h(\hat{x}_{k|k-1}))$$

- ⑤ Updated estimate covariance:

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Recall

Differential drive robot:



$$\dot{x} = \frac{r}{2}(\dot{\phi}_1 + \dot{\phi}_2) \cos(\theta)$$

$$\dot{y} = \frac{r}{2}(\dot{\phi}_1 + \dot{\phi}_2) \sin(\theta)$$

$$\dot{\theta} = \frac{r}{2L}(\dot{\phi}_1 - \dot{\phi}_2)$$

Walk through the process ...

Discrete approximation

We will use Euler's method for solving the differential equations. Let the time between measurements be denoted by Δt . Recall that if x is position then \dot{x} is velocity (and \ddot{x} is acceleration).

One may approximate a derivative by

$$\dot{x} \approx \frac{x(t + \Delta t) - x(t)}{\Delta t}$$

You can convert $\dot{x} = v$ by

$$\frac{x(t + \Delta t) - x(t)}{\Delta t} \approx v(t) \quad \rightarrow \quad x(t + \Delta t) = x(t) + v(t)\Delta t$$

Take a time step of Δt (meaning $t_{k+1} = t_k + \Delta t$), Euler's ("Oil-ler's") method is

$$x(t_{k+1}) = x(t_k) + (\Delta t)x'(t_k) \quad \text{and} \quad y(t_{k+1}) = y(t_k) + (\Delta t)y'(t_k).$$

Discrete approximation

The discretized variables are

$$t_k \equiv k\Delta t, \quad t_{k+1} = (k + 1)\Delta t$$

$$x_k \equiv x(t_k), \quad y_k \equiv y(t_k)$$

$$\omega_{1,k} \equiv \dot{\phi}_1(t_k), \quad \omega_{2,k} \equiv \dot{\phi}_2(t_k)$$

And so we can write our differential equations as difference equations...

Discrete approximation

$$\frac{x(t + \Delta t) - x(t)}{\Delta t} \approx \dot{x} = \frac{r}{2}(\dot{\phi}_1 + \dot{\phi}_2) \cos(\theta)$$

$$\frac{y(t + \Delta t) - y(t)}{\Delta t} \approx \dot{y} = \frac{r}{2}(\dot{\phi}_1 + \dot{\phi}_2) \sin(\theta)$$

$$\frac{\theta(t + \Delta t) - \theta(t)}{\Delta t} \approx \dot{\theta} = \frac{r}{2L}(\dot{\phi}_1 - \dot{\phi}_2)$$

Algebra:

$$x(t + \Delta t) \approx x(t) + \frac{r\Delta t}{2}(\dot{\phi}_1 + \dot{\phi}_2) \cos(\theta)$$

$$y(t + \Delta t) \approx y(t) + \frac{r\Delta t}{2}(\dot{\phi}_1 + \dot{\phi}_2) \sin(\theta)$$

$$\theta(t + \Delta t) \approx \theta(t) + \frac{r\Delta t}{2L}(\dot{\phi}_1 - \dot{\phi}_2)$$

Discrete Differential Drive Model

Using the discrete (sample) variables, $x(t) \rightarrow x_k$, etc, we can rewrite the expression in terms of the discrete variables.

Given starting configuration and wheel velocity measurements, we have:

$$x_{k+1} = x_k + \frac{r\Delta t}{2} (\omega_{1,k} + \omega_{2,k}) \cos(\theta_k)$$

$$y_{k+1} = y_k + \frac{r\Delta t}{2} (\omega_{1,k} + \omega_{2,k}) \sin(\theta_k)$$

$$\theta_{k+1} = \theta_k + \frac{r\Delta t}{2L} (\omega_{1,k} - \omega_{2,k})$$

EKF Example

Write in the EKF notation....

$$x_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}, \quad u_k = \begin{bmatrix} \omega_{1,k} \\ \omega_{2,k} \end{bmatrix}, \quad f(x_k, u_k) = \begin{bmatrix} x_k + \frac{r\Delta t}{2}(\omega_{1,k} + \omega_{2,k}) \cos(\theta_k) \\ y_k + \frac{r\Delta t}{2}(\omega_{1,k} + \omega_{2,k}) \sin(\theta_k) \\ \theta_k + \frac{r\Delta t}{2L}(\omega_{1,k} - \omega_{2,k}) \end{bmatrix}$$
$$F_k = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\frac{r\Delta t}{2}(\omega_{1,k} + \omega_{2,k}) \sin(\theta_k) \\ 0 & 1 & \frac{r\Delta t}{2}(\omega_{1,k} + \omega_{2,k}) \cos(\theta_k) \\ 0 & 0 & 1 \end{bmatrix}$$

EKF Example

Assume that you start the robot with pose $[0, 0, 0]$ and you know this is exact so

$$P_{0|0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let the process noise and measurement noise covariances be

$$V = \begin{bmatrix} 0.2 & 0.01 & 0.1 \\ 0.01 & 0.2 & 0.01 \\ 0.1 & 0.01 & 0.3 \end{bmatrix}, \quad W = \begin{bmatrix} 0.25 & 0 & 0.1 \\ 0 & 0.25 & 0.1 \\ 0.1 & 0.1 & 0.4 \end{bmatrix}$$

and the control inputs be $\omega_{1,0} = 1$, $\omega_{2,0} = 2$. Take $\Delta t = 0.1$, $r = 4$, $L = 6$.

EKF Example

Take

$$h_k(x_k) = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}, \quad H_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and so we expand our process:

- ① $\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k)$
- ② $P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + V_k$
- ③ $K_k = P_{k|k-1} (P_{k|k-1} + W_k)^{-1}$
- ④ $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - \hat{x}_{k|k-1})$
- ⑤ $P_{k|k} = (I - K_k) P_{k|k-1}$

EKF Example

$$\hat{x}_{1|0} = f(\hat{x}_{0|0}, u_0) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} \frac{4(0.1)}{2}(1+2)\cos(0) \\ \frac{4(0.1)}{2}(1+2)\sin(0) \\ \frac{4(0.1)}{12}(1-2) \end{pmatrix} = \begin{pmatrix} 0.6 \\ 0 \\ -0.333 \end{pmatrix}$$

EKF Example

$$F = \begin{bmatrix} 1 & 0 & -\frac{r\Delta t}{2}(\omega_{1,k} + \omega_{2,k}) \sin(\theta_k) \\ 0 & 1 & \frac{r\Delta t}{2}(\omega_{1,k} + \omega_{2,k}) \cos(\theta_k) \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.6 \\ 0 & 0 & 1 \end{bmatrix}$$

so ...

$$\begin{aligned} P_{1|0} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.6 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.6 & 1 \end{bmatrix} + \begin{bmatrix} 0.2 & 0.01 & 0.1 \\ 0.01 & 0.2 & 0.01 \\ 0.1 & 0.01 & 0.3 \end{bmatrix} \\ &= \begin{bmatrix} 0.2 & 0.01 & 0.1 \\ 0.01 & 0.2 & 0.01 \\ 0.1 & 0.01 & 0.3 \end{bmatrix} \end{aligned}$$

EKF Example

$$\begin{aligned} K &= \begin{bmatrix} 0.2 & 0.01 & 0.1 \\ 0.01 & 0.2 & 0.01 \\ 0.1 & 0.01 & 0.3 \end{bmatrix} \left[\begin{bmatrix} 0.2 & 0.01 & 0.1 \\ 0.01 & 0.2 & 0.01 \\ 0.1 & 0.01 & 0.3 \end{bmatrix} + \begin{bmatrix} 0.25 & 0 & 0.1 \\ 0 & 0.25 & 0.1 \\ 0.1 & 0.1 & 0.4 \end{bmatrix} \right]^{-1} \\ &= \begin{bmatrix} 0.2 & 0.01 & 0.1 \\ 0.01 & 0.2 & 0.01 \\ 0.1 & 0.01 & 0.3 \end{bmatrix} \begin{bmatrix} 2.552 & 0.126 & -0.749 \\ 0.126 & 2.317 & -0.400 \\ -0.749 & -0.400 & 1.705 \end{bmatrix} \\ &= \begin{bmatrix} 0.437 & 0.008 & 0.017 \\ 0.043 & 0.461 & -0.070 \\ 0.032 & -0.084 & 0.433 \end{bmatrix} \end{aligned}$$

EKF Example

Assume we have the observation: $z_k = [0.5, 0.025, -0.3]^T$ then the innovation

$$z_1 - \hat{x}_{1|0} = \begin{pmatrix} -.1 \\ 0.025 \\ 0.033 \end{pmatrix}$$

So,

$$\begin{aligned}\hat{x}_{1|1} &= \hat{x}_{1|0} + K_1 (z_1 - \hat{x}_{1|0}) \\ &= \begin{pmatrix} 0.6 \\ 0 \\ -0.333 \end{pmatrix} + \begin{bmatrix} 0.437 & 0.008 & 0.017 \\ 0.043 & 0.461 & -0.070 \\ 0.032 & -0.084 & 0.433 \end{bmatrix} \begin{pmatrix} -0.1 \\ 0.025 \\ 0.033 \end{pmatrix} \\ \hat{x}_{1|1} &= \begin{pmatrix} 0.557 \\ 0.005 \\ -0.324 \end{pmatrix}\end{aligned}$$

EKF Example

$$P_{1|1} = (I - K)P_{1|0} = \begin{bmatrix} 0.563 & -0.008 & -0.017 \\ -0.043 & 0.539 & 0.070 \\ -0.032 & 0.084 & 0.567 \end{bmatrix} \begin{bmatrix} 0.2 & 0.01 & 0.1 \\ 0.01 & 0.2 & 0.01 \\ 0.1 & 0.01 & 0.3 \end{bmatrix}$$

$$P_{1|1} = \begin{bmatrix} 0.111 & 0.004 & 0.051 \\ 0.004 & 0.108 & 0.022 \\ 0.051 & 0.022 & 0.168 \end{bmatrix}$$

EKF example 2

We will take a similar setup as before, with a few values modified, and generate the Python code required. Assume that you start the robot with pose $[0, 0, 0]$ and you know this is exact so

$$P_{0|0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let the process noise and measurement noise covariances be

$$V = \begin{bmatrix} 0.025^2 & 0 & 0 \\ 0 & 0.025^2 & 0 \\ 0 & 0 & 0.025^2 \end{bmatrix}, \quad W = \begin{bmatrix} 0.85^2 & 0 & 0 \\ 0 & 0.85^2 & 0 \\ 0 & 0 & 0.85^2 \end{bmatrix}$$

EKF example 2

The control inputs: $\omega_{1,k} = 1.5 \sin(t_k/10)$, $\omega_{2,k} = \cos(t_k/10)$.

Take $\Delta t = 0.1$, $r = 4$, $L = 6$,

$$F_k = \begin{bmatrix} 1 & 0 & -2\Delta t(\omega_{1,k} + \omega_{2,k}) \sin(\theta_k) \\ 0 & 1 & 2\Delta t(\omega_{1,k} + \omega_{2,k}) \cos(\theta_k) \\ 0 & 0 & 1 \end{bmatrix}$$

and

$$h_k(x_k) = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}, \quad H_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

EKF example 2

```
N = 100
mu1, sigma1 = 0.0, 0.025
mu2, sigma2 = 0.0, 0.85
var1 = sigma1*sigma1
var2 = sigma2*sigma2
dt = 0.1
r = 4
dd = 2.0*dt
L = 6
x = np.zeros((N,3))
z = np.zeros((N,3))
t = np.linspace(0, 10, 100)
w1 = 1.5*np.sin(t)
w2 = 1.0*np.cos(t)
```

EKF example 2

```
k = 1
while (k<N):
    q = np.random.normal(mu1,sigma1,3)
    r = np.random.normal(mu2,sigma2, 3)
    x[k,0] = x[k-1,0] + dd*(w1[k]+w2[k])*cos(x[k-1,2])
            + q[0]
    x[k,1] = x[k-1,1] + dd*(w1[k]+w2[k])*sin(x[k-1,2])
            + q[1]
    x[k,2] = x[k-1,2] + dd*(w1[k]-w2[k])/L + q[2]
    z[k,0] = x[k,0] + r[0]
    z[k,1] = x[k,1] + r[1]
    z[k,2] = x[k,2] + r[2]
    k = k+1
```

EKF example 2

```
H = np.array([[1,0,0],[0,1,0],[0,0,1]])
HT = H.T
V = np.array([[var1,0,0],[0,var1,0],[0,0,var1]])
W = np.array([[var2,0,0],[0,var2,0],[0,0,var2]])
P = np.zeros((N,3,3))
xf = np.zeros((N,3))
xp = np.zeros(3)
sp = np.zeros(3)
```

EKF example 2

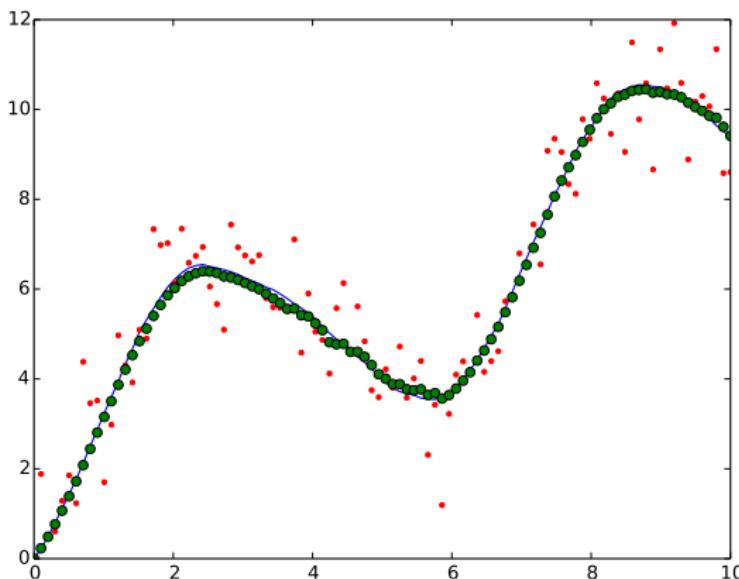
```
k = 1
while (k<N):
    xp[0] = xf[k-1,0] + dd*(w1[k]+w2[k])*cos(xf[k-1,2])
    xp[1] = xf[k-1,1] + dd*(w1[k]+w2[k])*sin(xf[k-1,2])
    xp[2] = xf[k-1,2] + dd*(w1[k]-w2[k])/L
    F1 = [1.0, 0.0, -dd*(w1[k]+w2[k])*sin(xf[k-1,2])]
    F2 = [0,1,dd*(w1[k]+w2[k])*cos(xf[k-1,2])]
    F = np.array([F1,F2,[0,0,1]])
    FT = F.T
    pp = np.dot(F,np.dot(P[k-1],FT)) + V
    y = z[k] - np.dot(H,xp)
    S = np.dot(H,np.dot(pp,HT)) + W
    SI = linalg.inv(S)
    kal = np.dot(pp,np.dot(HT,SI))
    xf[k] = xp + np.dot(kal,y)
    P[k] = pp - np.dot(kal,np.dot(H,pp))
    k = k+1
```

EKF example 2

```
t = np.arange(0,N,1)
plt.plot(t, x, 'b-', t,z,'r.', t, xf,'go')
plt.show()

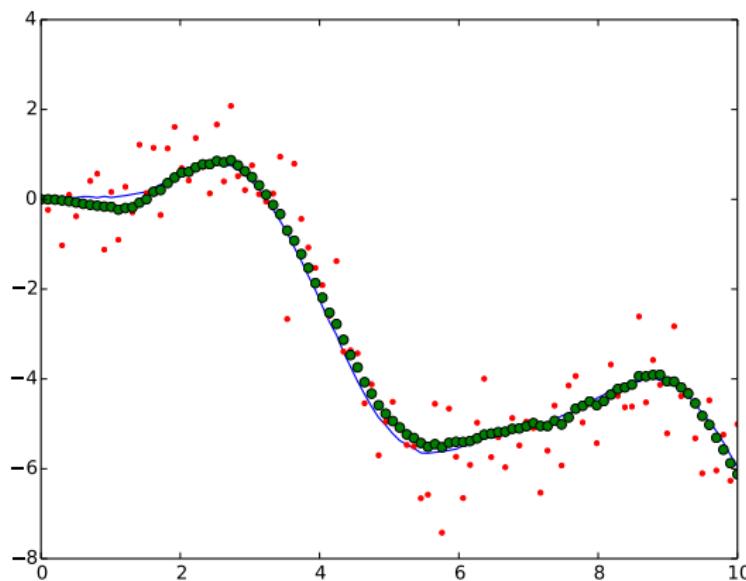
plt.plot(x[:,0], x[:,1], 'b-' ,z[:,0], z[:,1] , 'r.' ,
          xf[:,0], xf[:,1], 'go')
plt.show()
```

Extended Kalman Filter applied to DD robot.



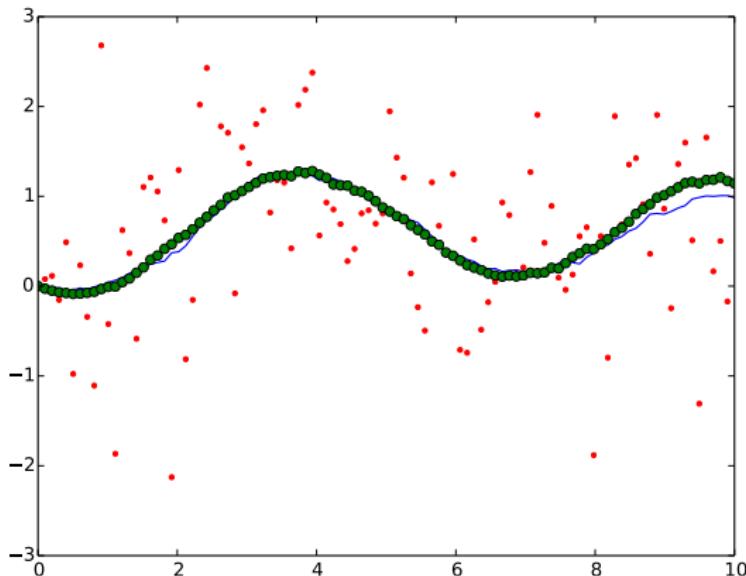
Domain axis is time and vertical axis is \hat{x} . The simulation pose is given by the blue line, the observation of the pose given by the red dots and the pose estimate is given by the green dots.

Extended Kalman Filter applied to DD robot.



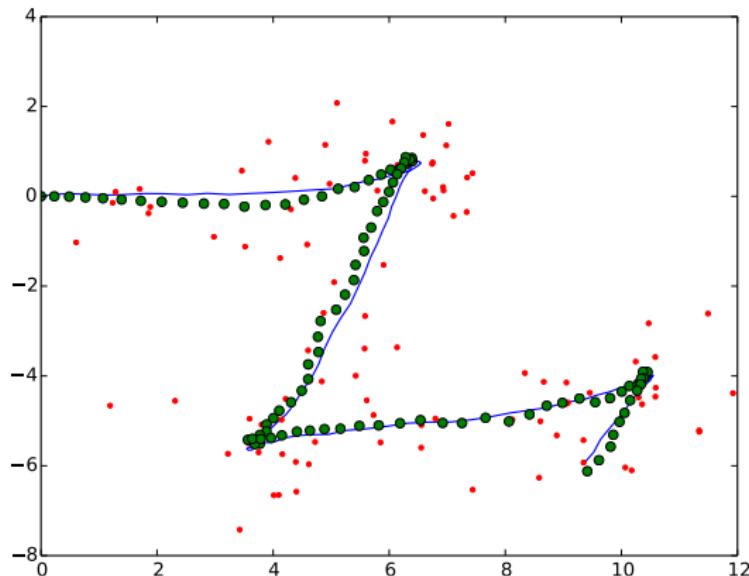
Domain axis is time and vertical axis is \hat{y} . The simulation pose is given by the blue line, the observation of the pose given by the red dots and the pose estimate is given by the green dots.

Extended Kalman Filter applied to DD robot.



Domain axis is time and vertical axis is $\hat{\theta}$. The simulation pose is given by the blue line, the observation of the pose given by the red dots and the pose estimate is given by the green dots.

Extended Kalman Filter applied to DD robot



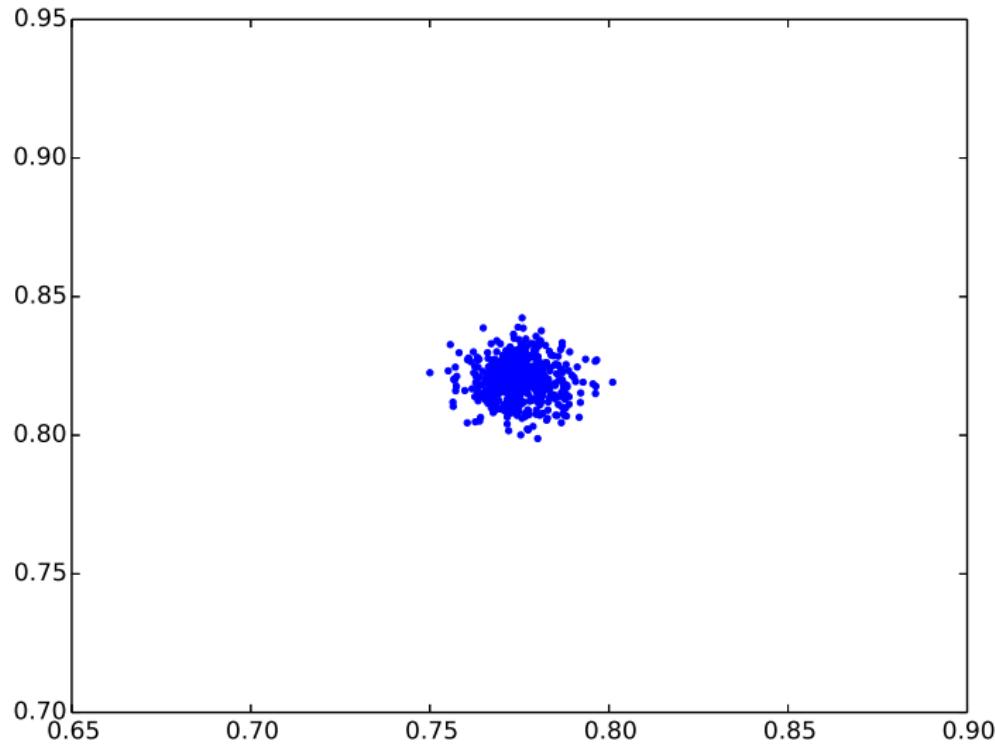
This figure plots the y against the x state variables (θ ignored). The simulation pose is given by the blue line, the observation of the pose given by the red dots and the pose estimate is given by the green dots.

More Examples

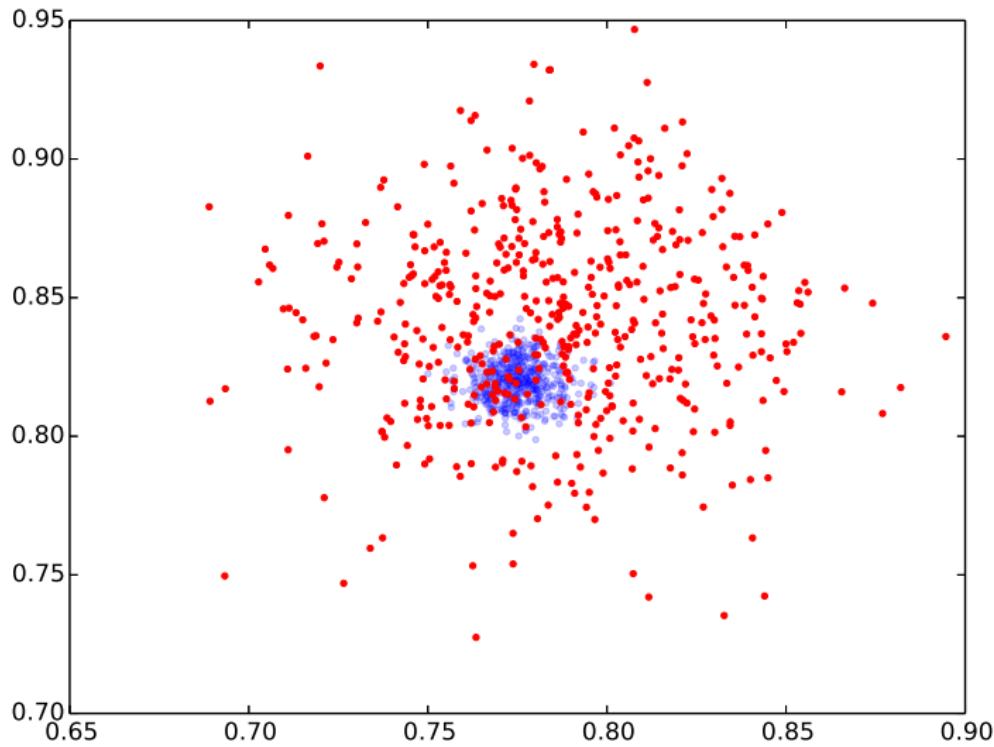
Point Cloud

How does the Kalman process affect a point cloud?

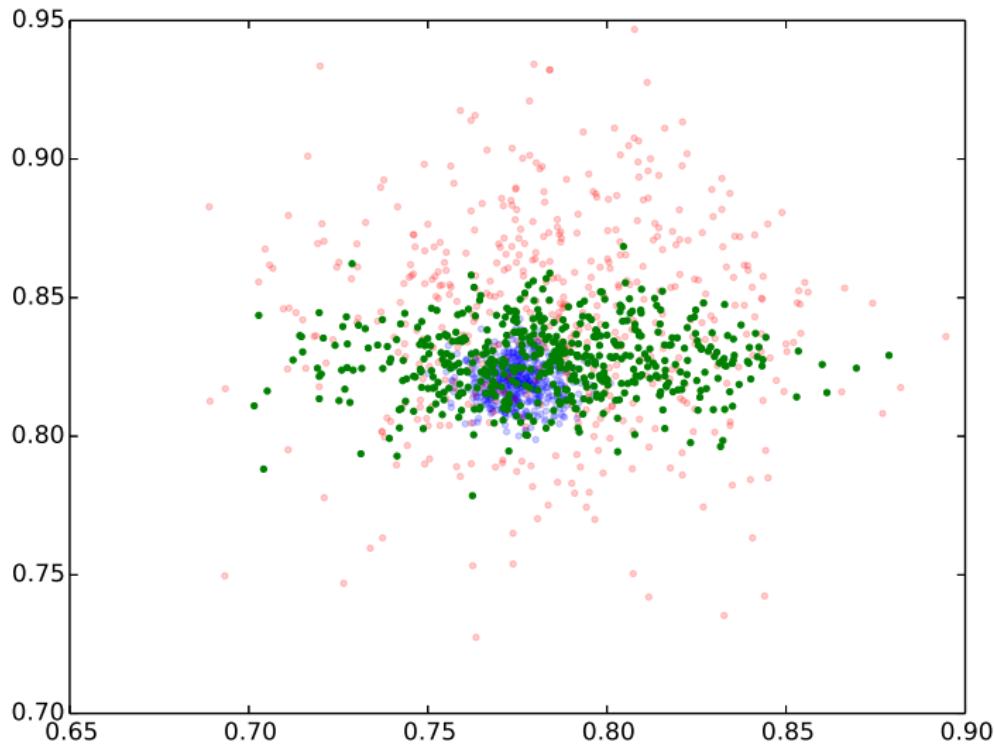
Point distribution after process update



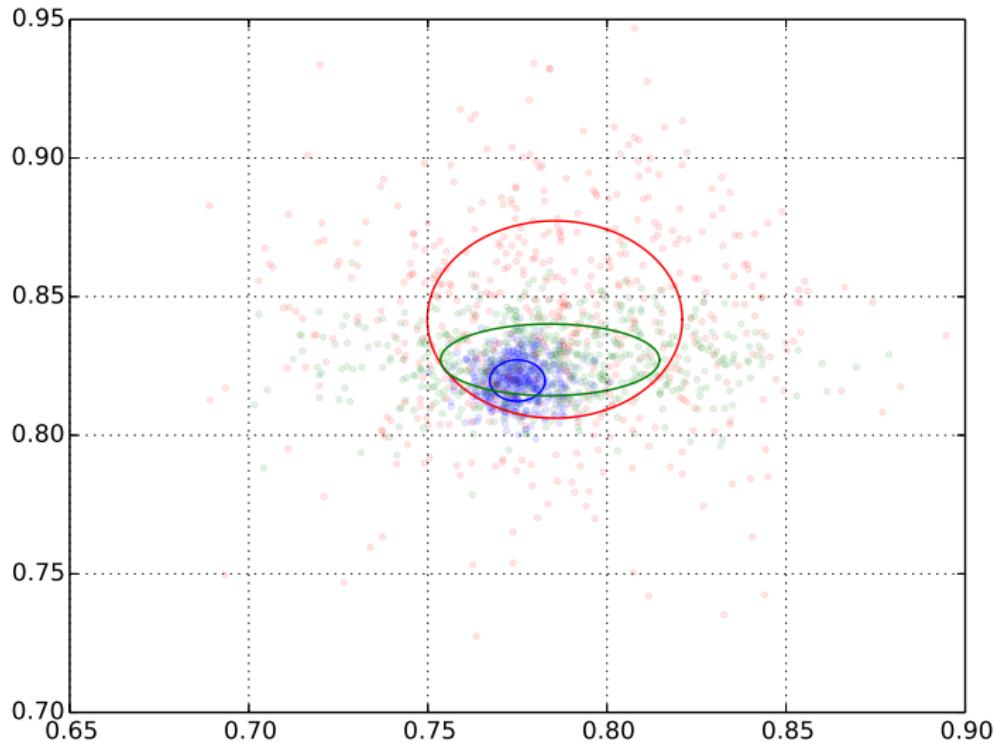
Observed point distribution



Final distribution after update step



The standard deviation based ellipses



Nonlinearity

There are other ways to deal with nonlinearity. One approach is known as the *Unscented Kalman Filter*. This algorithm is given on pg. 70 of Probabilistic Robotics. This filter is more accurate in terms of the approximation to the nonlinear function (second order instead of first order).

- ▶ For a linear system EKF and UKF are the same.
- ▶ For nonlinear systems UKF should produce equal or better results, however for many problems the results are the same.
- ▶ UKF does not use derivatives

Code ...

```
from math import *
import numpy as np
import pylab as plt
import matplotlib.mlab as mlab
from scipy import interpolate
from scipy import stats
from scipy import linalg
```

Code ...

The basic random number setup data:

```
V = np.array([[0.2, 0.02], [0.02, 0.35]])  
W = np.array([[0.4, 0.0], [0.0, 0.4]])
```

```
i = 0  
while (i < M):  
    r = np.random.multivariate_normal(mu1, V, 1)  
    q = np.random.multivariate_normal(mu2, W, 1)  
    xp = np.dot(F, xf0) + G + r  
    pp = np.dot(F, np.dot(P, FT)) + V  
    z = xp + q  
    y = z - xp  
    S = pp + W  
    kal = np.dot(pp, linalg.inv(S))  
    xf = xp + np.dot(kal, y)  
    i = i+1
```

Code ...

Let a , b be the major and minor axes, x_0 , y_0 be the center and α be the tilt angle.

```
def Ellipse(a,b,an,x0,y0):
    #Number of points to construct the ellipse:
    points=100
    cos_a,sin_a=cos(an*pi/180),sin(an*pi/180)
    the=np.linspace(0,2*np.pi,points)
    #Here goes the general ellipse,
    ##      x0, y0 is the origin of the ellipse in xy plane
    X=a*np.cos(the)*cos_a-sin_a*b*np.sin(the)+x0
    Y=a*np.cos(the)*sin_a+cos_a*b*np.sin(the)+y0
    return X,Y
```

Code ...

Setup for the Kalman process:

```
F = np.array([[0.85, -0.1],[0.02, 0.75]])  
FT = F.T  
G = np.array([0.025, 0.05]).T  
P = np.array([[0.01, 0.0],[0.0, 0.001]])  
xf0 = np.array([1,1]).T  
x1 = np.zeros((M))  
y1 = np.zeros((M))  
x2 = np.zeros((M))  
y2 = np.zeros((M))  
x3 = np.zeros((M))  
y3 = np.zeros((M))
```

Code ...

Loop over points and apply one step of the Kalman filter.

```
i = 0
while (i<M):
    r = np.random.multivariate_normal(mu1,V,1)
    q = np.random.multivariate_normal(mu2,W, 1)
    xp = np.dot(F,xf0) + G + r
    pp = np.dot(F,np.dot(P,FT)) + V
    z = xp + q
    res = z - xp
    S = pp + W
    kal = np.dot(pp,linalg.inv(S))
    xf = xp + np.dot(kal,res)
    x1[i] = xp[0]
    y1[i] = xp[1]
    x2[i] = z[0]
    y2[i] = z[1]
    x3[i] = xf[0]
    y3[i] = xf[1]
    i = i+1
```

Code ...

Create some graphics

```
u1,v1=Ellipse(sqrt(x1.var()),sqrt(y1.var()),0,x1.mean(),y1.  
    mean())  
u2,v2=Ellipse(sqrt(x2.var()),sqrt(y2.var()),0,x2.mean(),y2.  
    mean())  
u3,v3=Ellipse(sqrt(x3.var()),sqrt(y3.var()),0,x3.mean(),y3.  
    mean())  
  
plt.xlim([0.65,0.9])  
plt.ylim([0.7,0.95])  
plt.plot(x1,y1,'b.')  
plt.savefig("cloud1.pdf")  
plt.show()  
  
plt.xlim([0.65,0.9])  
plt.ylim([0.7,0.95])  
plt.plot(x1,y1,'b.',alpha = 0.2)  
plt.plot(x2,y2,'r.')  
plt.savefig("cloud2.pdf")  
plt.show()
```

Code ...

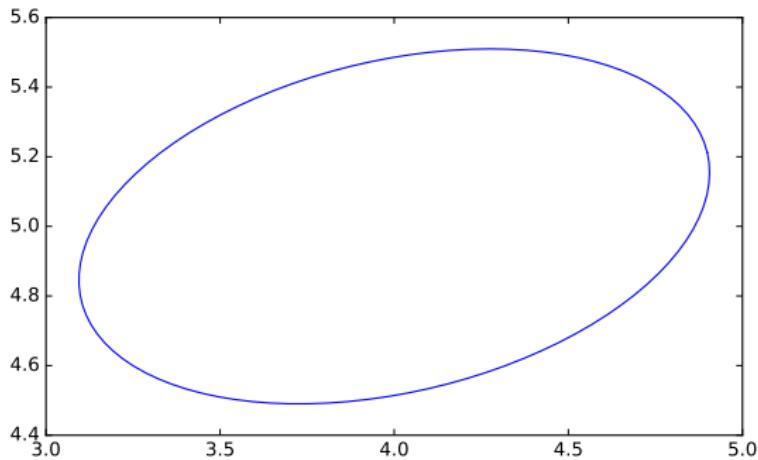
```
plt.xlim([0.65,0.9])
plt.ylim([0.7,0.95])
plt.plot(x1,y1,'b.',alpha = 0.2)
plt.plot(x2,y2,'r.',alpha = 0.2)
plt.plot(x3,y3,'g.')
plt.savefig("cloud3.pdf")
plt.show()

plt.axis([0.65,0.9,0.7,0.95])
plt.grid(True)
plt.plot(x1,y1,'b.',x2,y2,'r.',x3,y3,'g.', alpha = 0.1)
plt.plot(u1,v1,'b-',u2,v2,'r-',u3,v3,'g-')
plt.savefig("cloud4.pdf")
plt.show()
```

Error Ellipses

```
import math
import numpy as np
import pylab as plt
from numpy import linalg
P = np.array([[0.9, 0.1],[0.1, 0.5]])
w, v = linalg.eig(P)
angle = 180*math.atan2(v[0][0],v[0][1])/math.pi
u,v = Ellipse(w[0],w[1],angle, 4,5)
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(u,v,'b-')
ax.set_aspect('equal')
fig.savefig("Ellipse.pdf")
```

Error Ellipses



Scalar Form

Assume that you have a scalar variable x and that you know the system gains f, g, h :

$$\begin{aligned}x_k &= fx_{k-1} + gu_k + v_k \\z_k &= hx_k + w_k\end{aligned}$$

Let the process noise v_k have variance q and the observation noise w_k have variance r_k . We track the estimate (or mean) $\hat{x}_{k|k}$ and the variance $p_{k|k}$.

Scalar Form

The Scalar Kalman Filter can be written as

Predict: a priori

- ▶ Predicted state: $\hat{x}_{k|k-1} = f_k \hat{x}_{k-1|k-1} + g_k u_k$
- ▶ Predicted estimate variance: $p_{k|k-1} = f_k^2 p_{k-1|k-1} + q_k$

Update: a posteriori

- ▶ Optimal Kalman gain: $K_k = h_k p_{k|k-1} / (h_k^2 p_{k|k-1} + r_k)$
- ▶ Updated state estimate $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - h_k \hat{x}_{k|k-1})$
- ▶ Updated estimate variance: $p_{k|k} = (1 - K_k h_k) p_{k|k-1}$

Scalar Example

Assume that you are given a simple scalar process on $0 \leq k < N$:

$$x_k = x_{k-1} + u_k$$

where the control input is

$$u_k = 0.5 * (1 - 1.75k/N).$$

Also assume that you have process noise with standard deviation of 0.2 and observation noise with standard deviation of 0.75.

Scalar Example

When we don't have actual experimental data, we need to simulate the data. To illustrate the filter, we will create a noisy dataset:

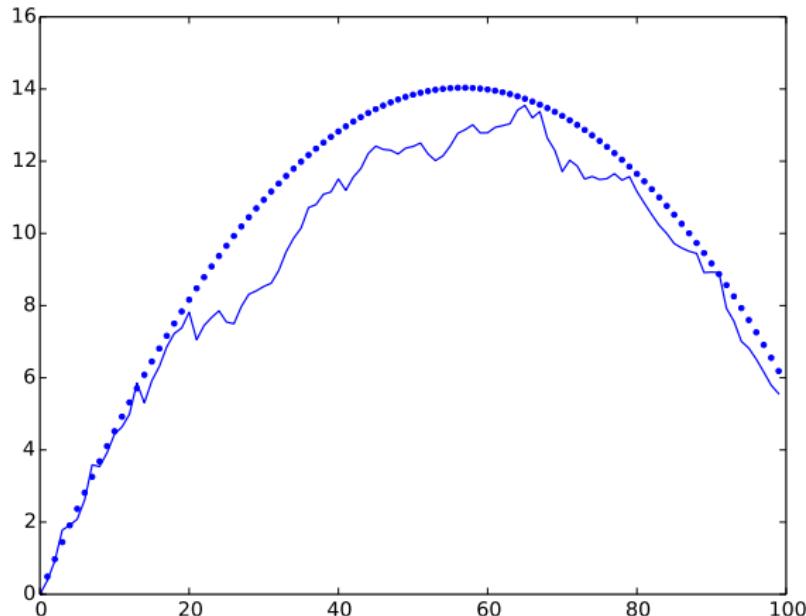
```
N = 100
mu1, sigma1 = 0.0, 0.3
mu2, sigma2 = 0.0, 0.85
r = np.random.normal(mu1, sigma1, N)
q = np.random.normal(mu2, sigma2, N)
x = np.zeros(N)
z = np.zeros(N)
u = np.arange(N)
k = 1
while (k < N):
    x[k] = x[k-1] + 0.5*(N-1.75*u[k])/N + r[k-1]
    z[k] = x[k] + q[k-1]
    k = k+1
```

Scalar Example

Now we pretend to run the dynamical system and get the observations.

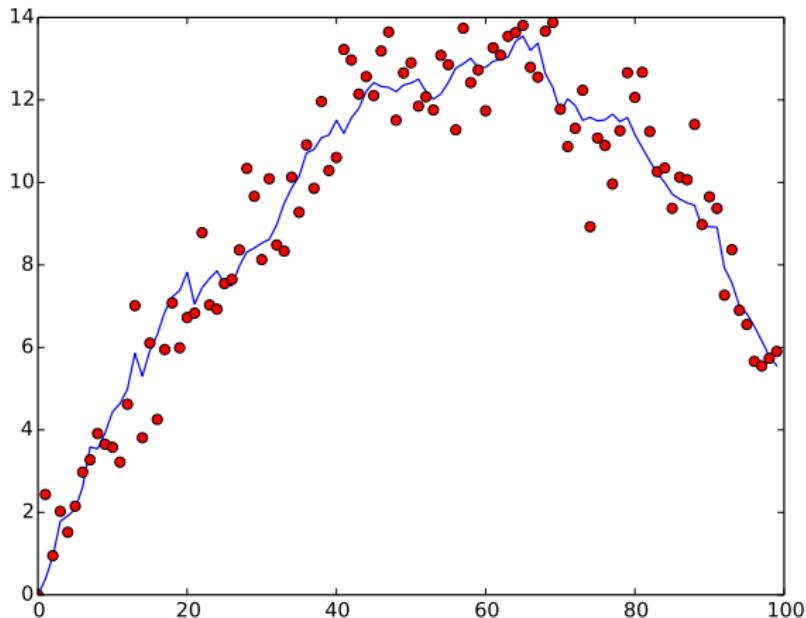
```
xf = np.zeros(N)
pf = np.zeros(N)
k = 1
while (k < N):
    xp = xf[k-1] + 0.5*(N-1.75*u[k])/N
    pp = pf[k-1] + sigma1*sigma1
    kal = pp/(pp + sigma2*sigma2)
    xf[k] = xp + kal*(z[k-1] - xp)
    pf[k] = (1-kal)*pp
    k = k+1
```

Scalar Example - Process

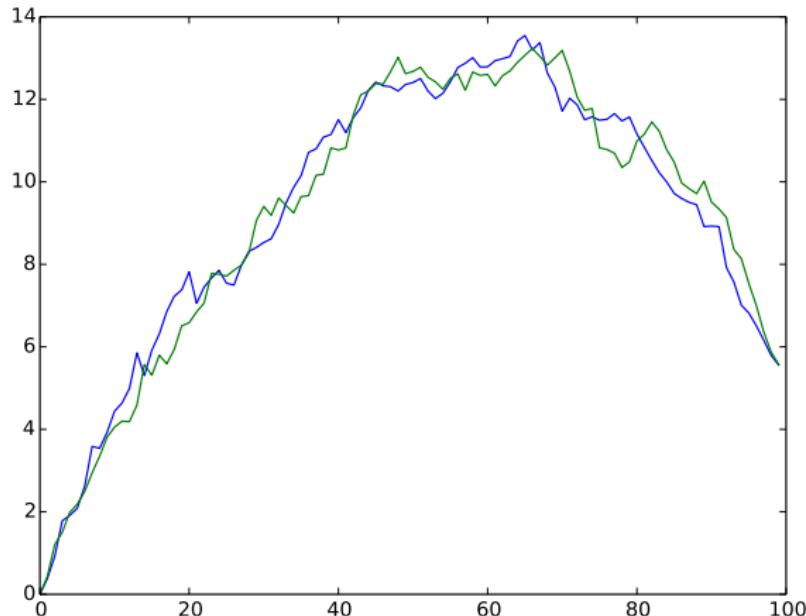


Pure process with dots, process with noise using line.

Scalar Example - Observation of process (with noise)



Scalar Example - Kalman State Estimate



State in blue, estimate in green.

Robot Design

Working with Humans

Robert Williams, an American Auto Worker.

Mr. Williams worked at a Ford Motor Company factory in Flat Rock, Michigan. He was working on January 25, 1979 with a parts-retrieval system that moved material from one part of the factory to another. When the robot began running slowly, Williams reportedly climbed into the storage rack to retrieve parts manually. He was struck in the head by the arm of a 1-ton production-line robot as he was gathering parts and killed instantly. He would go down in history as the first recorded human death by a robot. William's family successfully sued the manufacturers of the robot, Litton Industries, and was awarded \$10 million dollars. The court concluded that there were insufficient safety measures in place to prevent such an accident from happening.

Working with Humans

Kenji Urada, a Japanese maintenance engineer.

Urada worked at the Akashi Kawasaki Heavy Industries plant. On July 4, 1981, Urada was checking on a malfunctioning robot. He leaped over the protective fence and accidentally hit the on-switch, resulting in the robot pushing him into a grinding machine with its hydraulic arm and crushing him to death. Mr. Urada is the second individual to be listed as a death by a robot.

Working with Humans

Wanda Holbrook, an American technician.

In July 2015, Wanda Holbrook, a maintenance technician performing routine duties on an assembly line at Ventra Ionia Main, an auto-parts maker in Ionia, Michigan, was "trapped by robotic machinery" and crushed to death.

Robotics Accident Data

Table 1 Table was compiled from US Government office OSHA, 30 years, 1987-2017, of OSHA Robotics related fatal accident reports.

6/29/87 Employee Crushed And Killed By Robot Arm
11/28/87 Employee Crushed To Death While Working On A Lathe
5/17/89 Employee Crushed And Killed By Industrial Robot
10/1/92 Employee Killed When Crushed By Robot Arm
3/13/93 Employee Killed When Crushed By Robot
4/8/94 Employee Dies From Coronary Insufficiency
9/27/94 Employee Crushed To Death By Activated Robot
2/15/96 Employee Killed When Burned By Robotic Hot Metal Pourer
1/27/97 Crushed By Robot Arm During Machine Cycle
4/29/97 Employee Struck By Material Handling Equipment
5/4/99 Employee Killed When Crushed By Robot Arms
6/8/99 Employee Killed When Crushed By Robot Against Conveyor
8/27/99 Employee Killed In Robotic Weld Cell
12/29/01 Employee Killed When Robot Pinned His Neck
7/28/03 Employee Is Killed When Crushed By Equipment
12/13/03 Employee Is Killed When Caught In Robotic Arm
3/30/04 Employee Was Killed By Industrial Robots
3/22/06 Employee Dies When Struck By Robotic Equipment
7/24/06 Employee Is Killed When Crushed By Robot
10/31/06 Worker Is Killed, Lockout Procedures Not Followed
5/13/07 Employee Dies After Being Struck By Robotic Arm
7/21/09 Employee Is Killed By Robotic Palletizer
8/2/11 Employee Is Killed When Caught In Equipment
12/15/12 Robot Crushes And Kills Worker Inside Robot Work Cell
3/7/13 Maintenance Worker Is Struck And Killed By Robot
6/16/13 Employee Is Struck By Axis Arm, Later Dies
7/7/15 Employee Is Killed When Head Is Crushed By Robot Arm
6/19/16 Employee is Killed While Making Repairs

OSHA

When OSHA was founded in 1971, there was an estimated 14,000 job related fatalities every year. This is roughly 38 deaths per day. In 2017, the number has dropped to around 12 per day.

The vast majority of these deaths are impact related. The deaths are from falling or impact (struck by vehicles or other machinery).

Next in line are workplace violence, electrocutions and drowning. Careful design, planning and implementation of the workspace can prevent injuries and fatalities as well as save considerable finances.

Founded in 1974, Robotics Industries Association, RIA, is the only trade group in North America organized specifically to serve the robotics industry.

Member companies include leading robot manufacturers, users, system integrators, component suppliers, research groups, and consulting firms.

<https://www.robotics.org/>

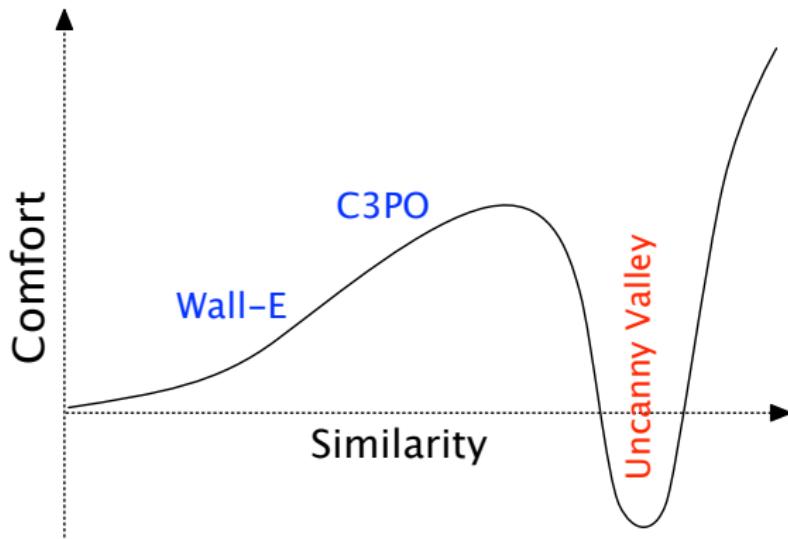
A full hazard identification and analysis is required. The standard identifies the following aspects:

- ① the intended operations at the robot, including teaching, maintenance, setting and cleaning;
- ② unexpected start-up;
- ③ access by personnel from all directions;
- ④ reasonably foreseeable misuse of the robot;
- ⑤ the effect of failure in the control system; and
- ⑥ where necessary, the hazards associated with the specific robot application.

Collaboration with Humans

- ① Environmental Awareness
 - ② Direct Communication
 - ③ Indirect Communication
 - ④ The real world
 - ⑤ Close interactions
 - ⑥ Appearance and the opposite

Uncanny Valley



Cybersecurity

Robotics software is complicated.

Current design approaches use multiple cores and cpus.

Interprocess communication is done via buses or sockets. Effectively a robot is a collection of networked nodes. As such it is prone to all of the security issues found in any distributed system.

It is in effect the IOT (Internet of Things) security problem.

Cybersecurity

Using Garfinkle and Spafford's definition of security, that the computer should behave as expected, there are a number of security issues.

The term security has become associated with preventing system cracking but secure really means that you can trust the system. You may not care about intrusion or data exposure, but you do care that the system operates the way you need it to.

ROS, the Robot Operating System

ROS is not an operating system, but a middleware layer and software collection.

ROS manages socket communication over multiple nodes.

ROS currently only runs on Linux.

Unix and thus Linux, was not designed with security in mind. It was designed for ease of use, flexibility, extensibility and an expectation of user sophistication. Much of Unix was developed by students and researchers to facilitate projects and not as production (engineered) code.

Security Planning

- ① Aspects of Planning
- ② Risk Assessment
- ③ Cost-Benefit Analysis
- ④ Creating Policies
- ⑤ Implementation
- ⑥ Validation

Risk Assessments

- ▶ What am I trying to protect?
Human life and limb, data, control, expensive hardware, ...
- ▶ What do I need to protect against?
The elements, environmental variation, hostile humans or software, ...
- ▶ How much money, time and effort am I willing to spend to obtain adequate protection?
Cost of loss vs cost of protection.

Immediately after you can perform the following steps:

- ① Identify assets
- ② Identify threats
- ③ Calculate risks

Network issues to be aware of ...

- ▶ Network wiretapping
- ▶ Port scans and Idle scans
- ▶ Denial-of-service attack
- ▶ Spoofing
- ▶ Man in the middle
- ▶ ARP spoofing
- ▶ Smurf attack
- ▶ Buffer overflow
- ▶ Heap overflow
- ▶ SQL injection

Cybersecurity

- ▶ Insecure Embedded Devices
- ▶ Computer Vision Vulnerabilities
- ▶ Sensor Compromise