上海科技大学
ShanghaiTech University

CS283: **Robotics Fall 2017: Summary**

Sören Schwertfeger / 师泽仁

ShanghaiTech University

STAR CENTER

Important Info!

- Exam:
  - Place: This lecture room + Robotics Teaching Lab
  - Time: January 3^{th}, 10:00 – 12:00
- Material allowed:
  - Any robotics book
  - Any other printed material except:
- Material **not** allowed:
  - Printout of any lecture slides (handwritten copies are ok)
  - Any electronics (Computer, Smartphone, Smartwatch, Calculator, …)

- Paper will be provided – bring your own pens ;)

- Only answers given in English will be accepted.

Why **Autonomous** Mobile Robotics?

- Tele-operated robots: boring and inefficient
- Autonomous robots: Robots that act by their own reasoning
  - Human operator might be present: Gives high level tasks
- Why autonomy?
  - Autonomous behaviors might be **better** than remote control by humans
  - Remote control might be **boring** or **stressful** and **tiresome**
  - Human operators might be a **scarce** resource or **expensive**
  - Multi robot approaches: One operator for many robots
- Semi-autonomy:
  - Autonomous behaviors that help the operator, for example:
  - Way-point navigation, autonomous stair climbing, assisted manipulation
  - Gradual development from tele-operation to full autonomy possible

- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.

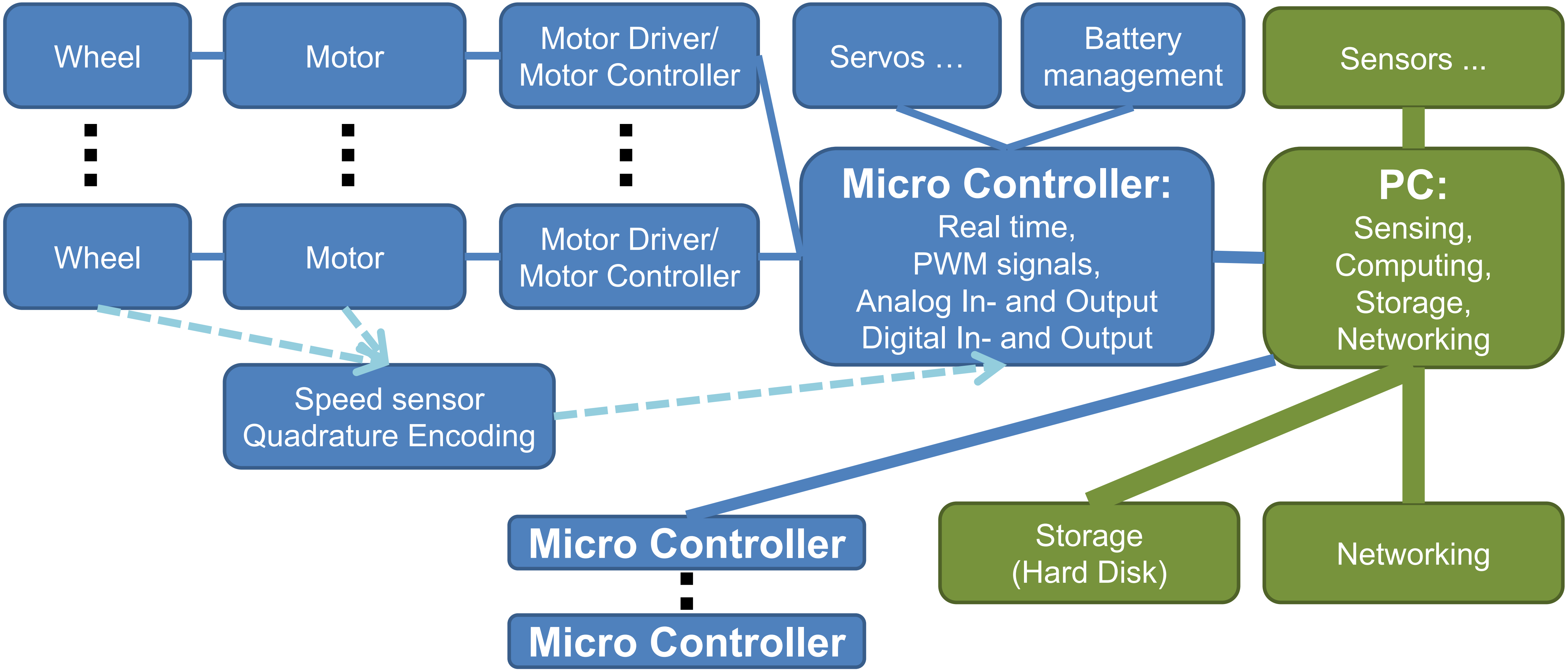
- Where am I?
  - Global Positioning System: outdoor, meters of error
  - Guiding system: (painted lines, inductive guides), markers, iBeacon
  - Model of the environment (Map), Localize yourself in this model
    - Build the model online: Mapping
    - Localization: determine position by comparing sensor data with the map
    - Do both at the same time: Simultaneous Localization and Mapping (SLAM)

- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.

- Where is my goal?
- Two part problem:
  - What is the goal?
    - Expressed using the world model (map)
    - Using object recognition
    - No specific goal (random)
  - Where is that goal?
    - Coordinates in the map
    - Localization step at the end of the object recognition process
    - User input

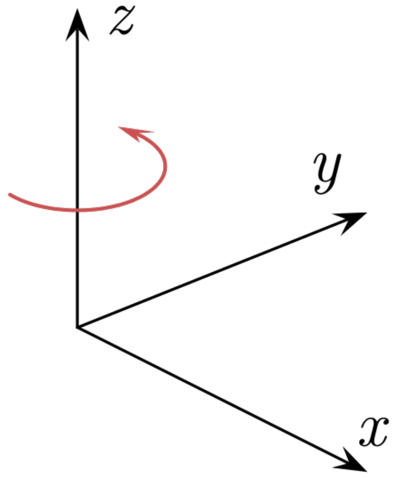
- Autonomous mobile robots move around in the environment. Therefore **ALL** of them:
  - They need to know **where** they **are**.
  - They need to know **where** their **goal** is.
  - They need to know **how** to get there.
- Different levels:
  - Control:
    - How much power to the motors to move in that direction, reach desired speed
  - Navigation:
    - Avoid obstacles
    - Classify the terrain in front of you
    - Predict the behavior (motion) of other agents (humans, robots, animals, machines)
  - Planning:
    - Long distance path planning
    - What is the way, optimize for certain parameters

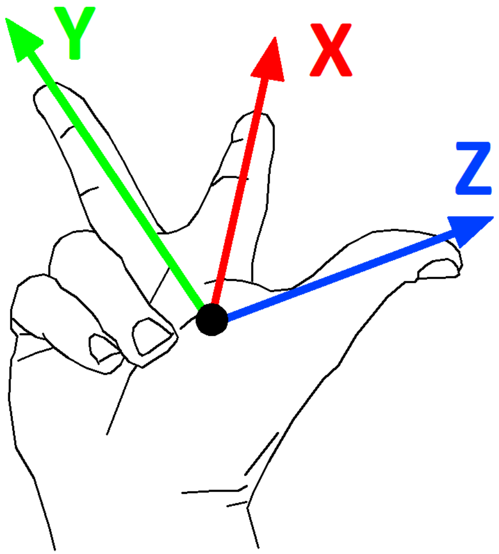
Overview Hardware

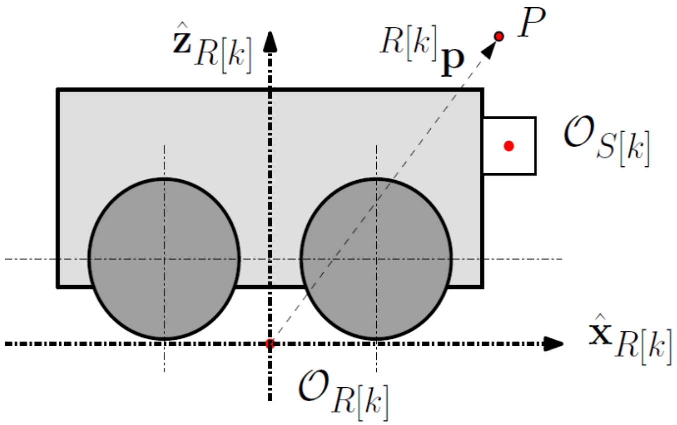


Right Hand Coordinate System

- Standard in Robotics
- Positive rotation around X is anti-clockwise
- Right-hand rule mnemonic:
  - Thumb: z-axis
  - Index finger: x-axis
  - Second finger: y-axis
  - Rotation: Thumb = rotation axis, positive rotation in finger direction
- Robot Coordinate System:
  - X front
  - Z up (Underwater: Z down)
  - Y ???



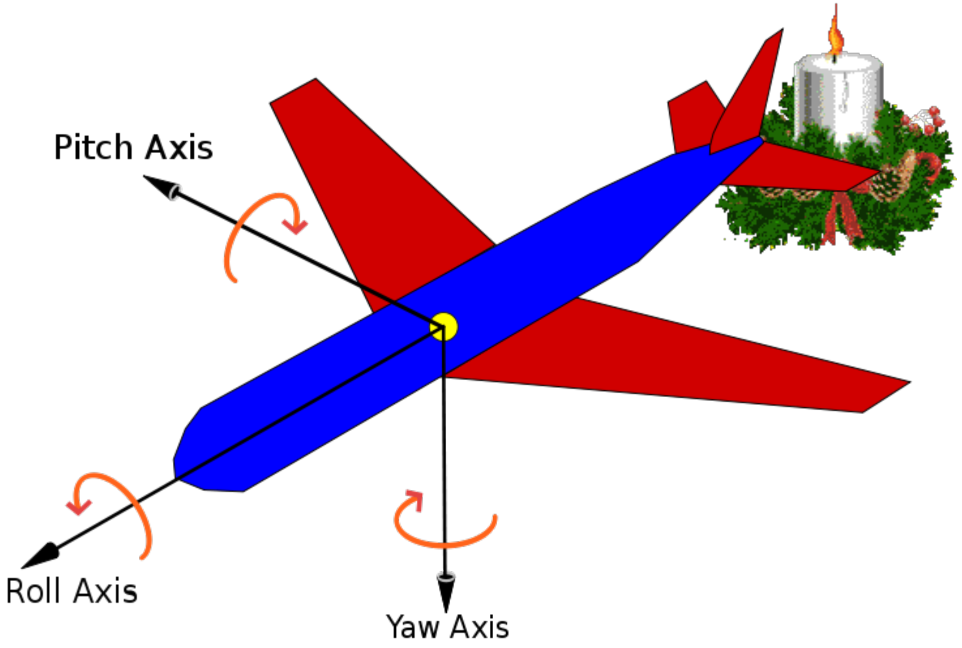




Right Hand Rule http://en.wikipedia.org/wiki/Right-hand_rule

3D Rotation

- Euler angles: Roll, Pitch, Yaw
  - ☹ Singularities
- Quaternions:
  - Concatenating rotations is computationally faster and numerically more stable
  - Extracting the angle and axis of rotation is simpler
  - Interpolation is more straightforward
  - Unit Quaternion: norm = 1
  - Scalar (real) part: q_0 , sometimes q_w
  - Vector (imaginary) part: $\mathbf{q}$
  - Over determined: 4 variables for 3 DoF

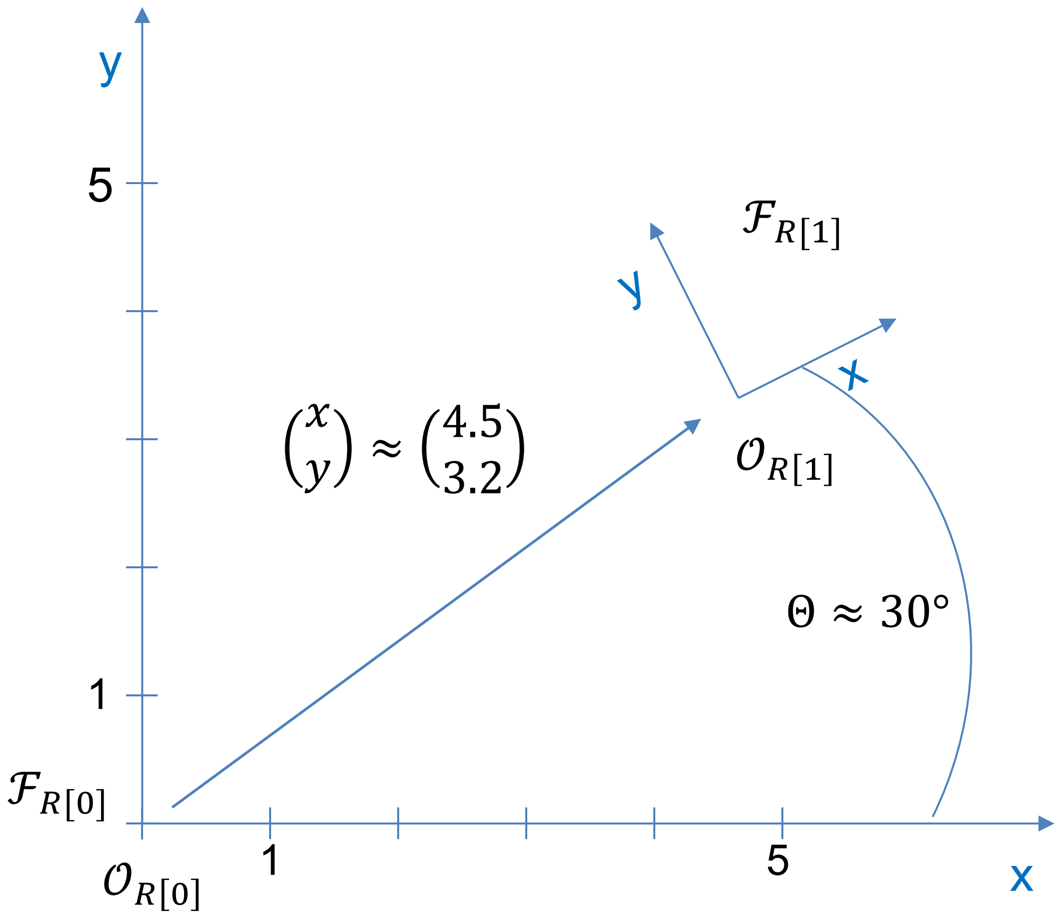


$$\check{\mathbf{p}} \equiv p_0 + p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}$$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

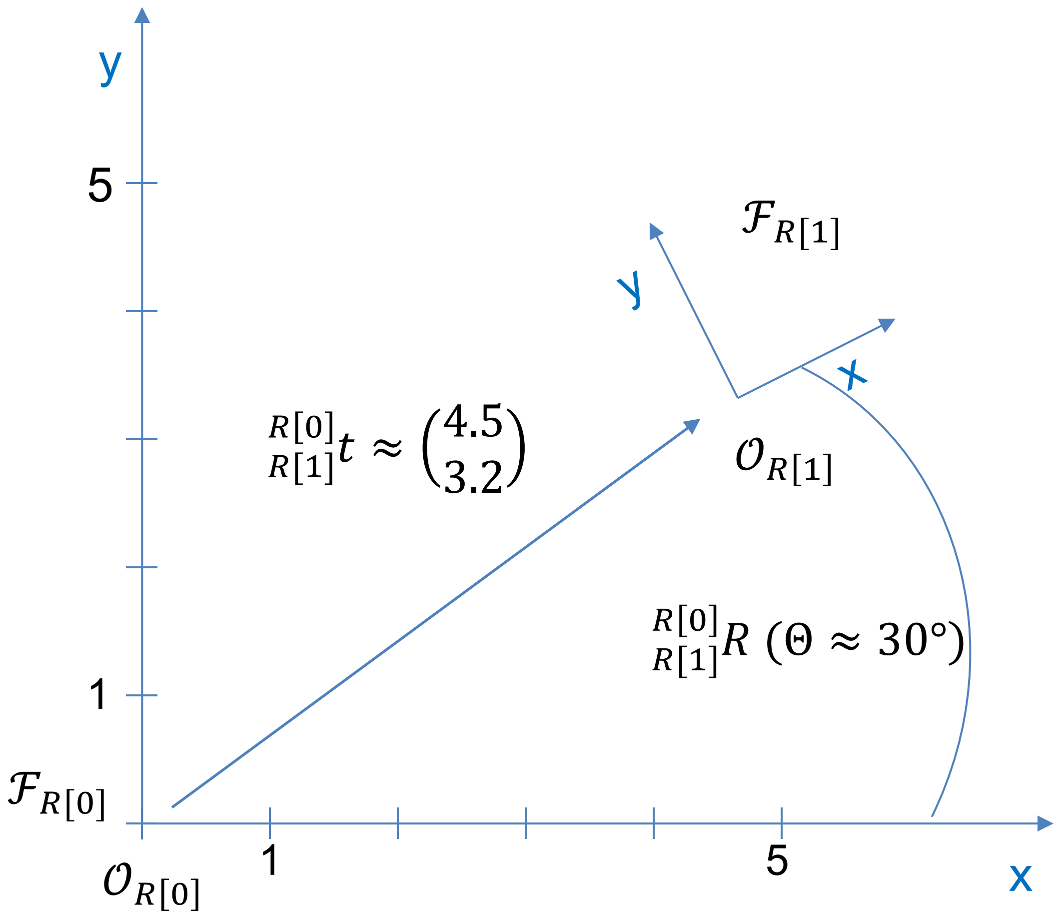
$$\check{\mathbf{q}} = \begin{pmatrix} q_0 & q_x & q_y & q_z \end{pmatrix}^{\mathsf{T}} \equiv \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix}$$

Position, Orientation & Pose



- **Position**:
  - $\binom{x}{y}$ coordinates of any object or point (or another frame)
  - with respect to (wrt.) a specified frame
- **Orientation**:
  - (Θ) angle of any oriented object (or another frame)
  - with respect to (wrt.) a specified frame
- **Pose**:
  - $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$ position and orientation of any oriented object
  - with respect to (wrt.) a specified frame

Translation, Rotation & Transform



- **Translation**:
  - $\begin{pmatrix} x \\ y \end{pmatrix}$ difference, change, motion from one reference frame to another reference frame
- **Rotation**:
  - (Θ) difference in angle, rotation between one reference frame and another reference frame
- **Transform**:
  - $\begin{pmatrix} x \\ y \\ \Theta \end{pmatrix}$ difference, motion between one reference frame and another reference frame

Transform in 3D

Matrix Euler Quaternion

$$ {}^{G}_{A}\mathbf{T} = \begin{bmatrix} {}^{G}_{A}\mathrm{R} & {}^{G}_{A}\mathrm{t} \\ 0_{1x3} & 1 \end{bmatrix} = \begin{pmatrix} {}^{G}_{A}\mathrm{t} \\ {}^{G}_{A}\Theta \end{pmatrix} = \begin{pmatrix} {}^{G}_{A}\mathrm{t} \\ {}^{G}_{A}\check{\mathrm{q}} \end{pmatrix} $$

$$ {}^{G}_{A}\Theta \triangleq \left(\theta_r, \theta_p, \theta_y\right)^T $$

In ROS: Quaternions! (w, x, y, z)
Uses Bullet library for Transforms

Rotation Matrix 3x3

$$ R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} $$

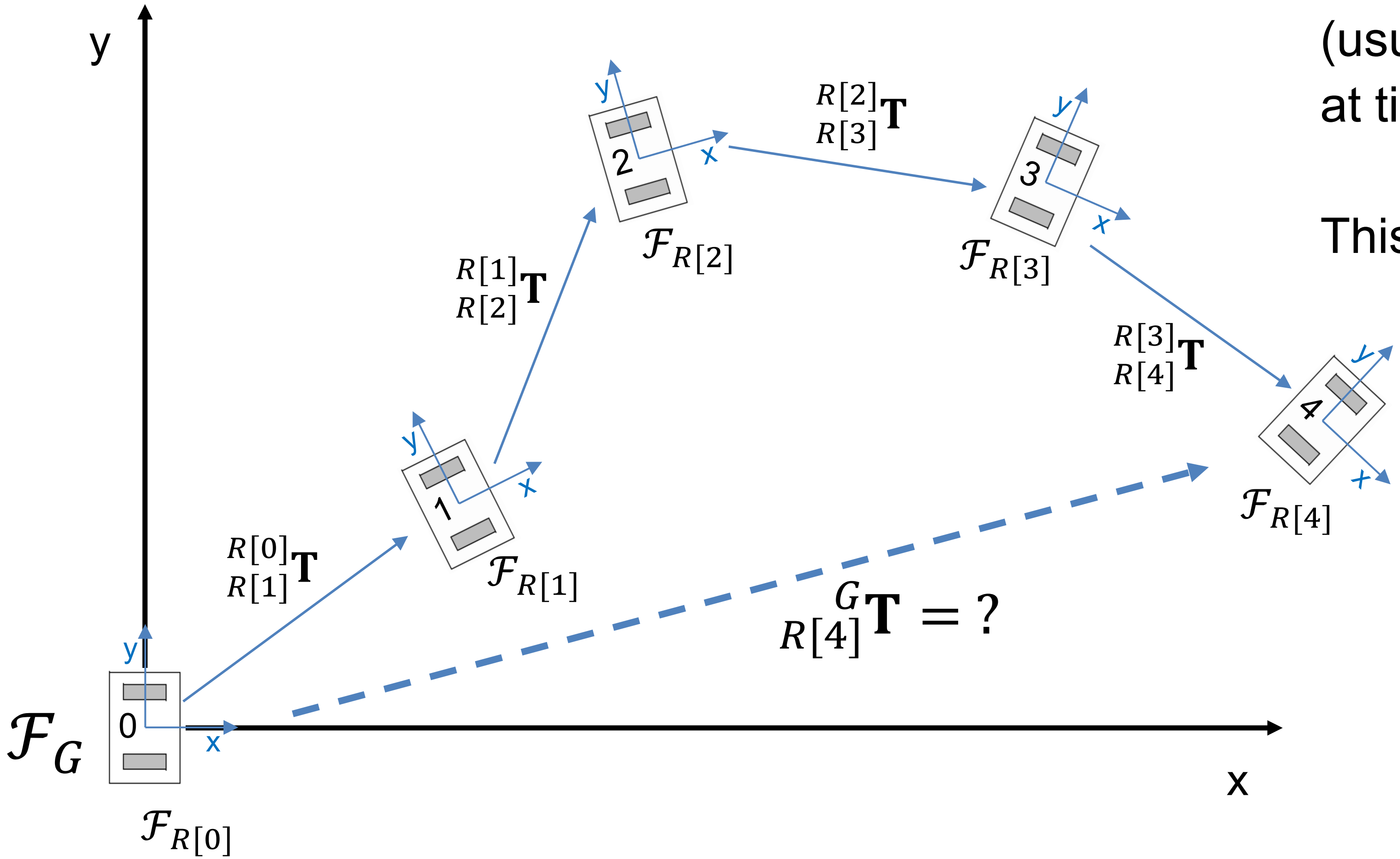
$$ R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} $$

$$ R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} $$

$$ R = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma) $$

yaw = α, pitch = β, roll = γ

Transforms



Where is the Robot now?

The pose of $\mathcal{F}_{R[X]}$ with respect to $\mathcal{F}_G$ (usually = $\mathcal{F}_{R[0]}$) is the pose of the robot at time X.

This is equivalent to ${}^{G}_{R[X]}\mathbf{T}$

Chaining of Transforms

$$ {}^{G}_{R[X+1]}\mathbf{T} = {}^{G}_{R[X]}\mathbf{T} \; {}^{R[X]}_{R[X+1]}\mathbf{T} $$

often: $\mathcal{F}_G \equiv \mathcal{F}_{R[0]} \Rightarrow {}^{G}_{R[0]}\mathbf{T} = id$

In ROS

- First Message at time 97 : G
- Message at time 103 : X
- Next Message at time 107 : X+1

$$ {}^{G}_{R[X+1]}\mathbf{T} = {}^{G}_{R[X]}\mathbf{T} \; {}^{R[X]}_{R[X+1]}\mathbf{T} $$

$$ {}^{R[X]}_{R[X+1]}t_x $$

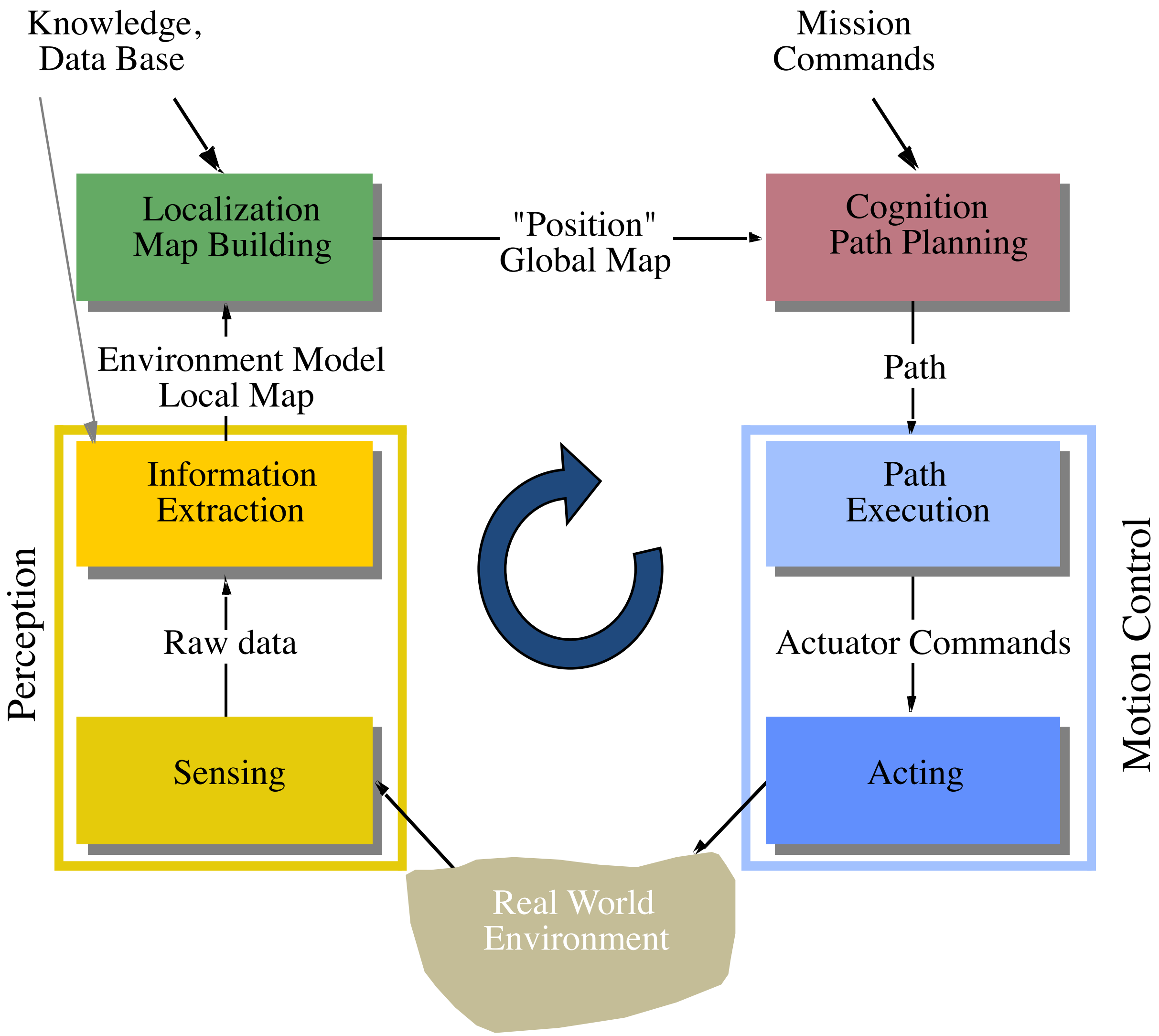
$$ {}^{R[X]}_{R[X+1]}t_y $$

$$ {}^{R[X]}_{R[X+1]}\Theta $$

```
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
geometry_msgs/Pose2D pose2D
  float64 x
  float64 y
  float64 theta
```

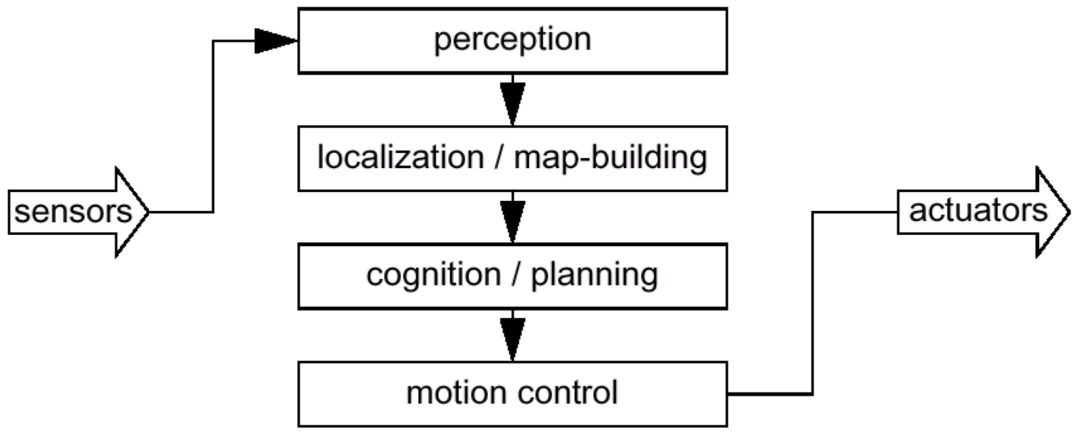
- Take a look at the other related Pose or Transform messages in ROS!

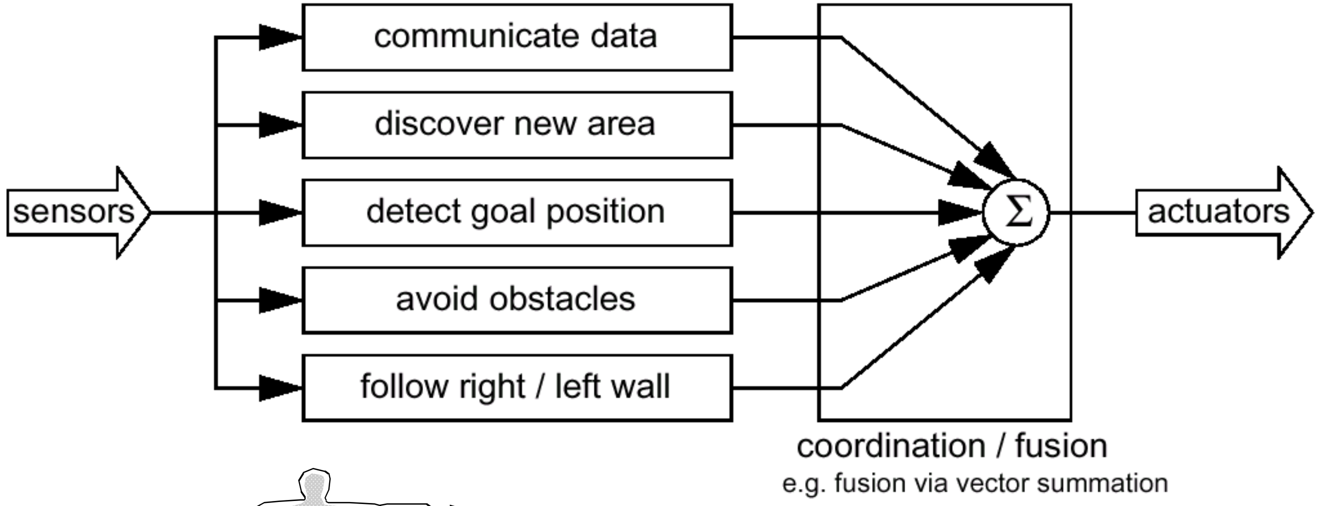
General Control Scheme for Mobile Robot Systems

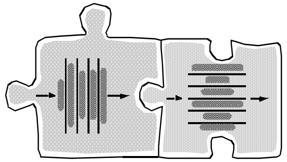


Two Approaches

- Classical AI
  (model based navigation)
  - complete modeling
  - function based
  - horizontal decomposition
- New AI
  (behavior based navigation)
  - sparse or no modeling
  - behavior based
  - vertical decomposition
  - bottom up
- Possible Solution
  - Combine Approaches

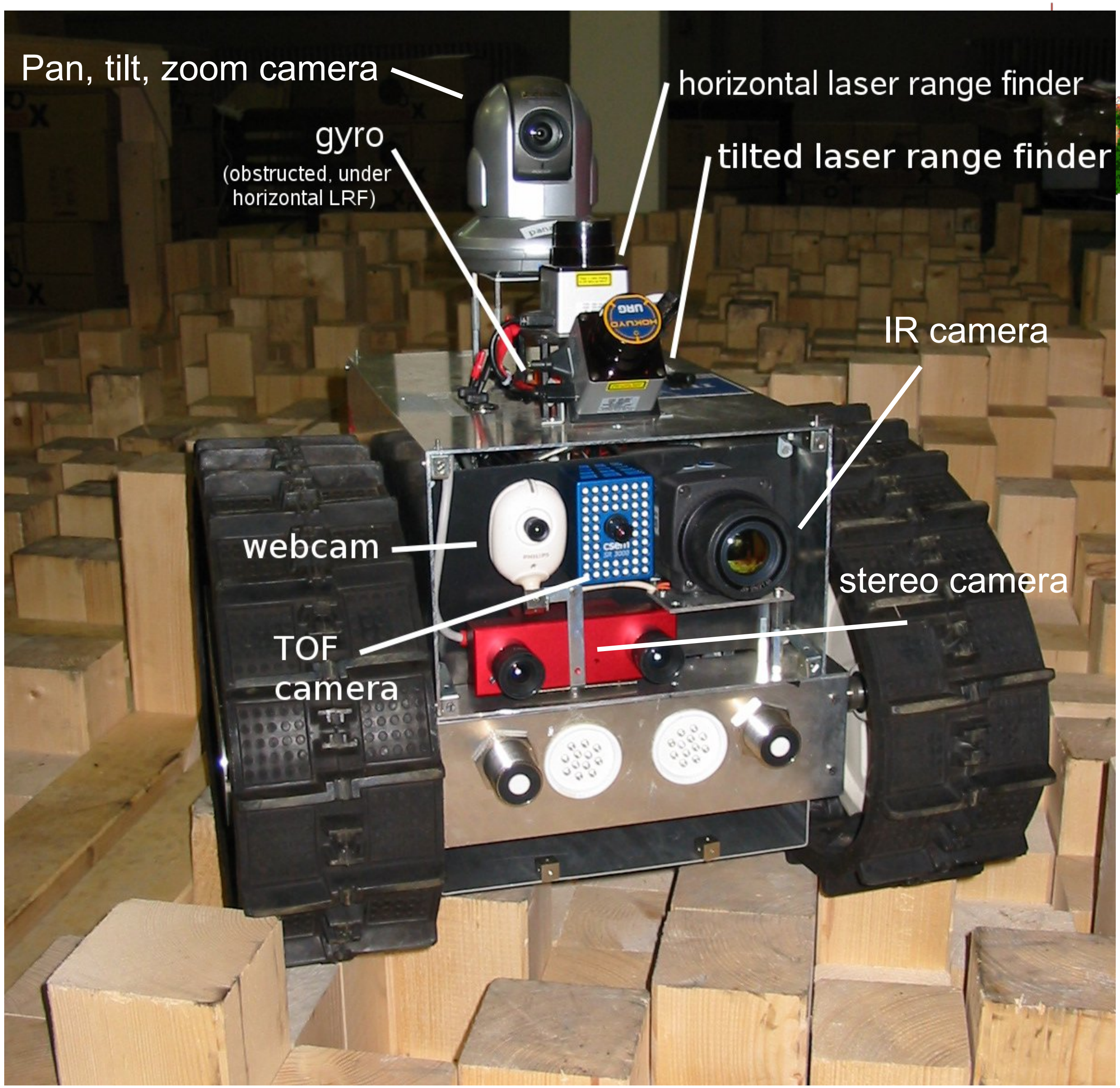






Sensors: outline

- Optical encoders
- Heading sensors
  - Compass
  - Gyroscopes
- Accelerometer
- IMU
- GPS
- Range sensors
  - Sonar
  - Laser
  - Structured light
- Vision



Classification of Sensors

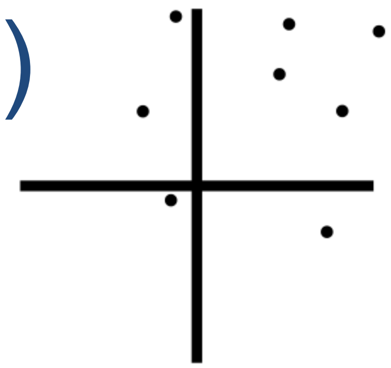
- What:
  - Proprioceptive sensors
    - measure values internally to the system (robot),
    - e.g. motor speed, wheel load, heading of the robot, battery status
  - Exteroceptive sensors
    - information from the robots environment
    - distances to objects, intensity of the ambient light, unique features.
- How:
  - Passive sensors
    - Measure energy coming from the environment
  - Active sensors
    - emit their proper energy and measure the reaction
    - better performance, but some influence on environment

In Situ Sensor Performance (2)

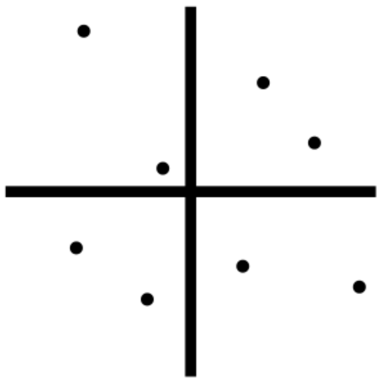
- Error / Accuracy
  - How close to true value
- Precision
  - Reproducibility

- Systematic error -> deterministic errors
  - caused by factors that can (in theory) be modeled -> prediction
  - e.g. calibration of a laser sensor or of the distortion cause by the optic of a camera

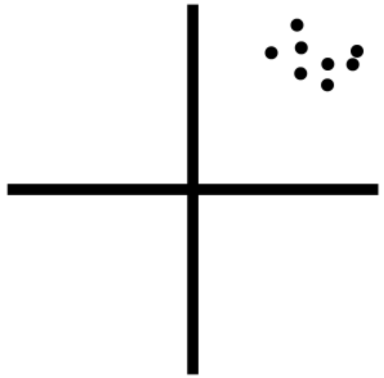
- Random error -> non-deterministic
  - no prediction possible
  - however, they can be described probabilistically
  - e.g. Hue instability of camera, black level noise of camera ..



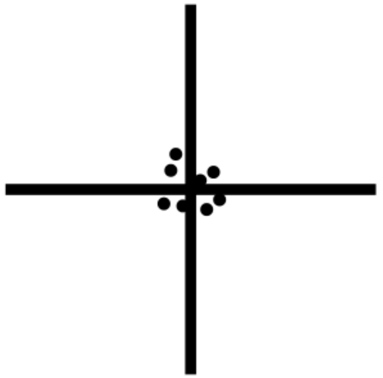
(a) Low precision and low accuracy



(b) Low precision and high accuracy



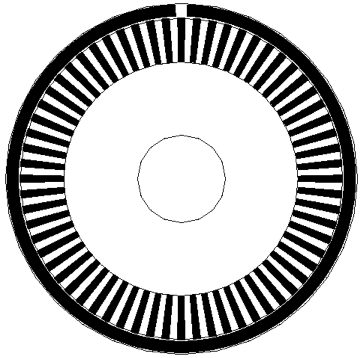
(c) High precision and low accuracy

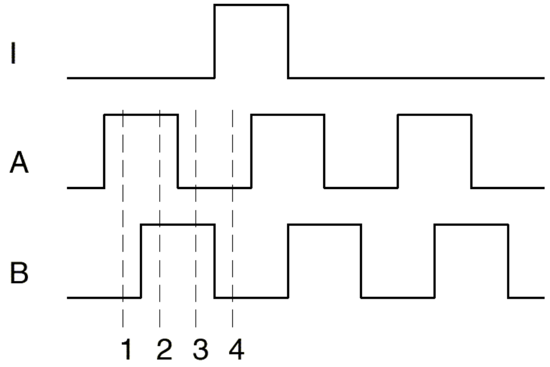


(d) High precision and high accuracy

Wheel / Motor Encoders

- measure position or speed of the wheels or steering
- integrate wheel movements to get an estimate of the position -> odometry
- optical encoders are proprioceptive sensors
- typical resolutions: 64 - 2048 increments per revolution.
  - for high resolution: interpolation
- optical encoders
  - regular: counts the number of transitions but cannot tell the direction of motion
  - quadrature: uses two sensors in quadrature-phase shift. The ordering of which wave produces a rising edge first tells the direction of motion. Additionally, resolution is 4 times bigger
  - a single slot in the outer track generates a reference pulse per revolution

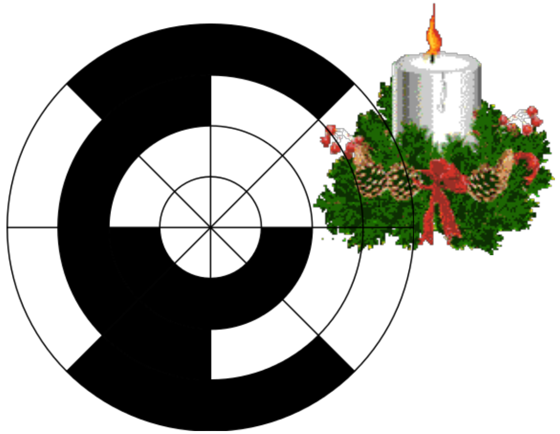




State	Ch A	Ch B
S_1	High	Low
S_2	High	High
S_3	Low	High
S_4	Low	Low

Gray Encoder

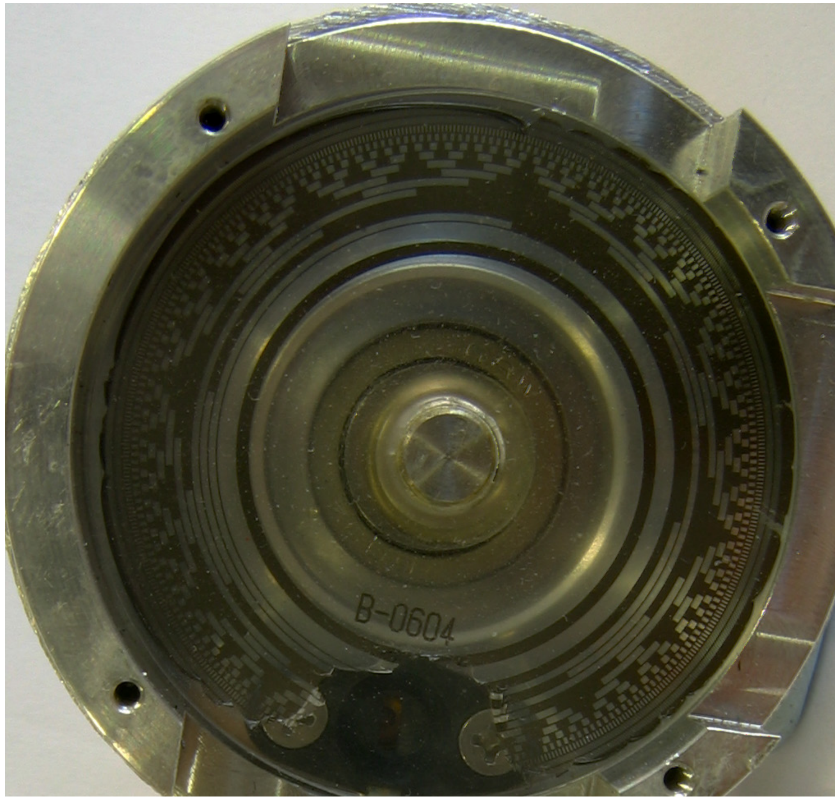
http://en.wikipedia.org/wiki/Gray_code



- Aka: reflected binary code, Gray Code
  - Binary numeral system where two successive values differ in only one bit
  - Also used for error correction in digital communications

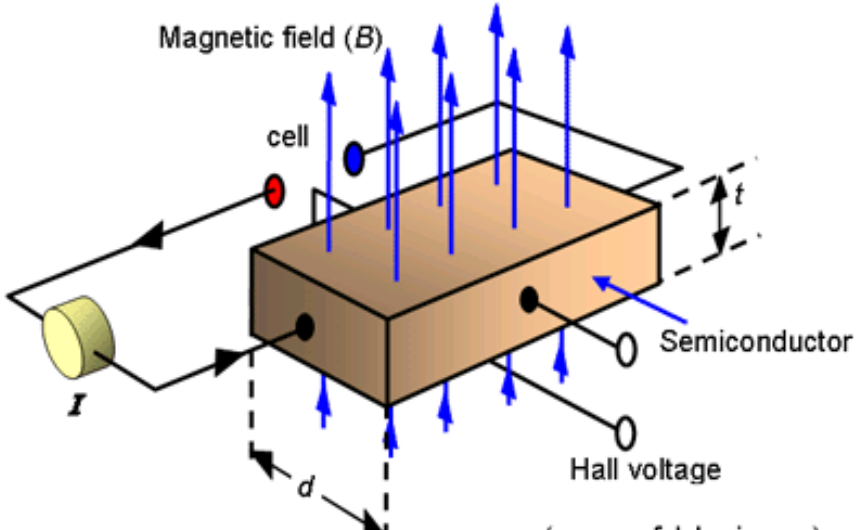
- Absolute position encoder
  - Normal binary => change from 011 to 100
  - 2 bits change – NEVER simultaneously =>
  - 011 -> 111 -> 101 -> 100 or
  - 011 -> 010 -> 110 -> 100 ….
  - => wrong encoder positions might be read
  - Gray encoding: only one bit change!

Dec	Gray	Binary
0	000	000
1	001	001
2	011	010
3	010	011
4	110	100
5	111	101
6	101	110
7	100	111



Compass

- Since over 2000 B.C.
  - China: suspended a piece of naturally magnetite from a silk thread to guide a chariot over land.
- Magnetic field on earth
  - absolute measure for orientation (even birds use it for migrations (2001 discovery))
- Large variety of solutions to measure the earth magnetic field
  - mechanical magnetic compass
  - direct measure of the magnetic field (Hall-effect, magneto-resistive sensors)
- Major drawback
  - weakness of the earth field (30 μTesla)
  - easily disturbed by magnetic objects or other sources
  - bandwidth limitations (0.5 Hz) and susceptible to vibrations
  - not feasible for indoor environments for absolute orientation
  - useful indoor (only locally)

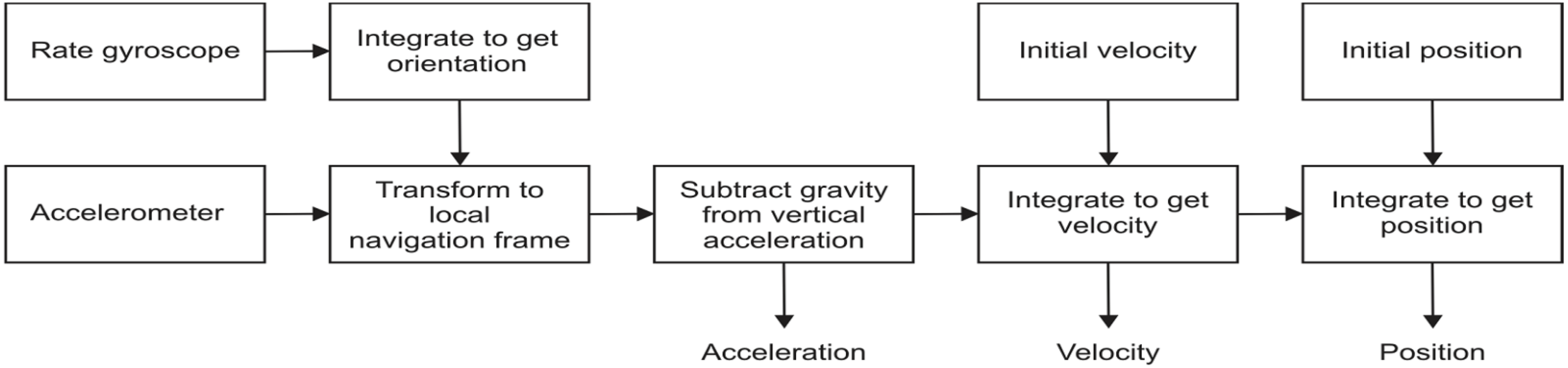


(resourcefulphysics.org)

Inertial Measurement Unit (IMU)

- Device combining different measurement systems:
  - Gyroscopes, Accelerometers, Compass
- Estimate relative position (x, y, z), orientation (roll, pitch, yaw), velocity, and acceleration
- Gravity vector is subtracted to estimate motion
  - Initial velocity has to be known

Xsens MTI



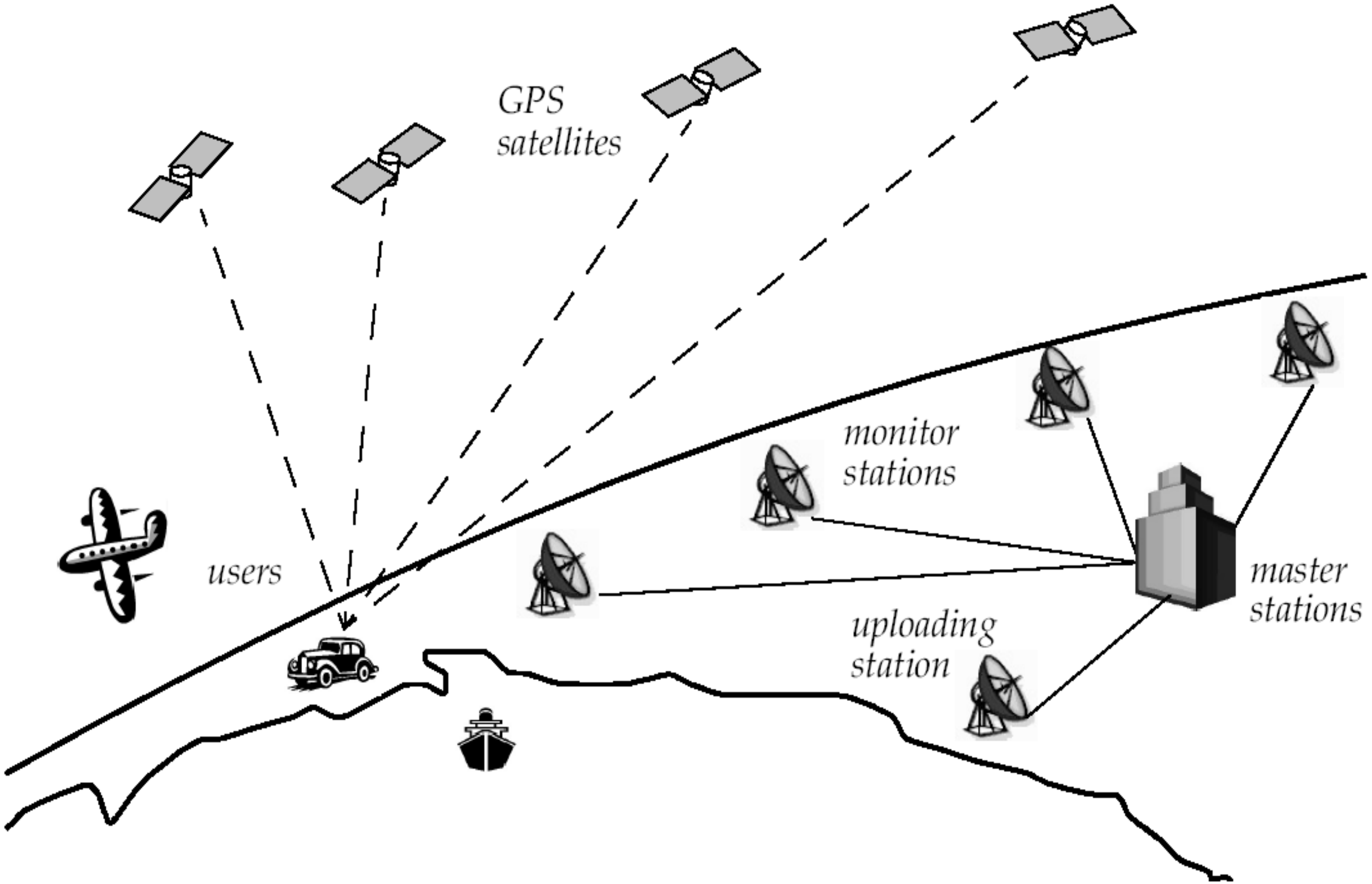
IMU Error and Drift

- Extremely sensitive to measurement errors in gyroscopes and accelerometers:
  - drift in the gyroscope unavoidably =>
  - error in orientation relative to gravity =>
  - incorrect cancellation of the gravity vector.

- Accelerometer data is integrated twice to obtain the position => gravity vector error leads to quadratic error in position.

- All IMUs drift after some time
  - Use of external reference for correction:
  - compass, GPS, cameras, localization

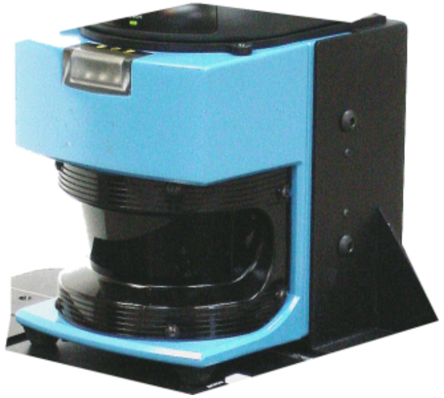
Global Positioning System (GPS) (2)



Range sensors

- Sonar -->

- Laser range finder --->



- Time of Flight Camera -->

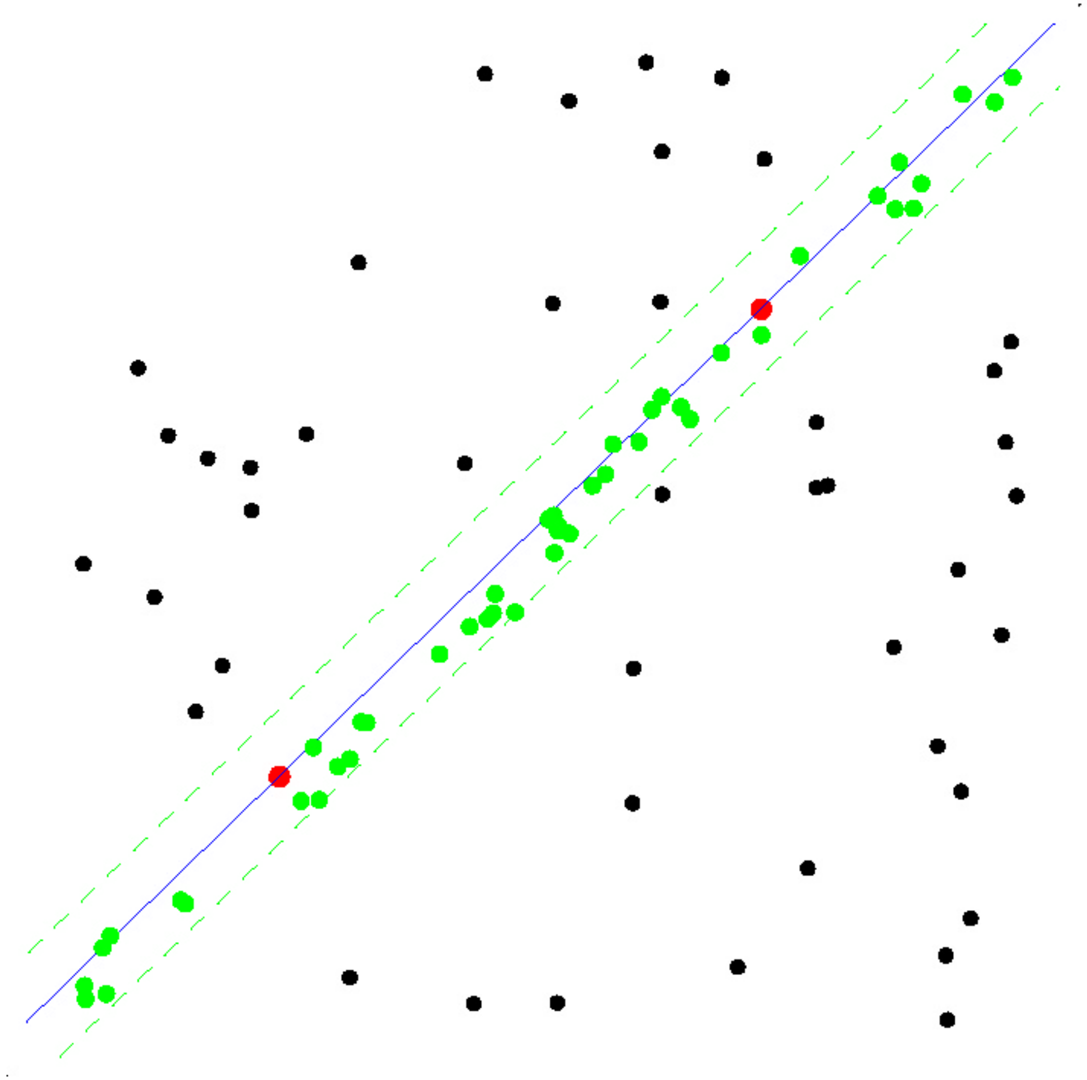
- Structured light ----------->

LINE EXTRACTION

Split and merge
Linear regression
RANSAC
Hough-Transform

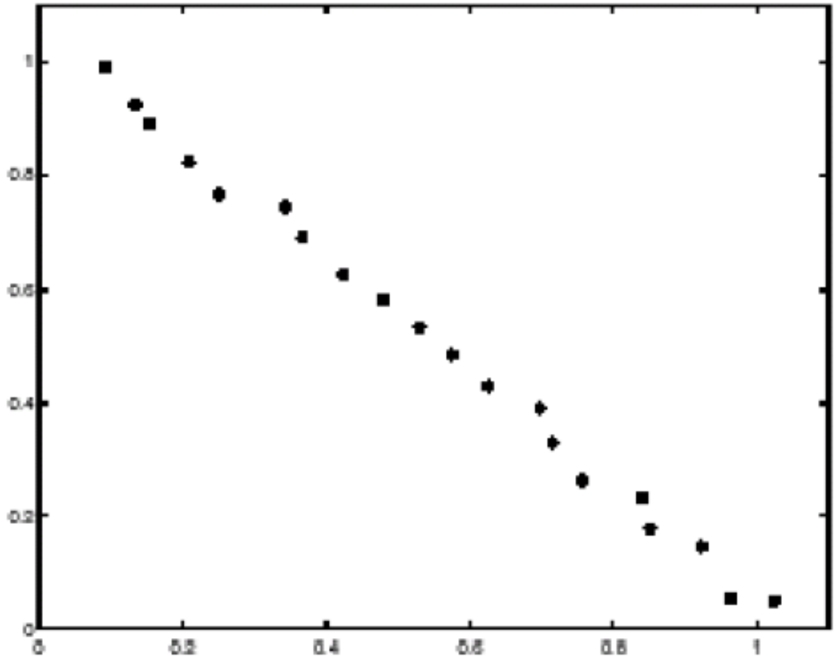
Algorithm 3: RANSAC

ALL-INLIER SAMPLE

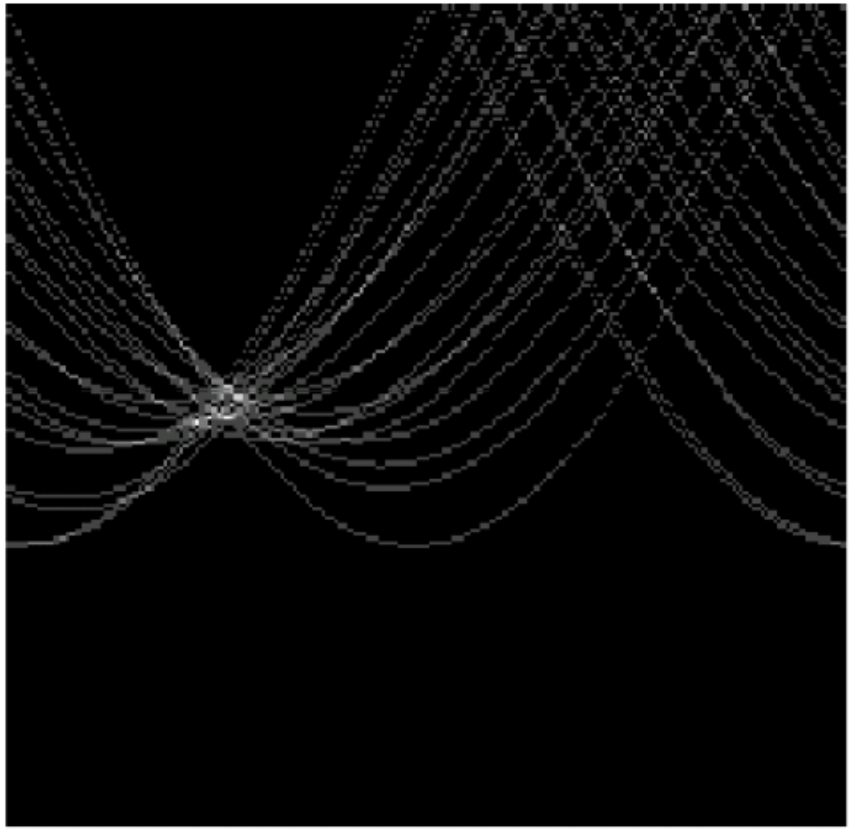


Algorithm 4: Hough-Transform

Effect of Noise



features

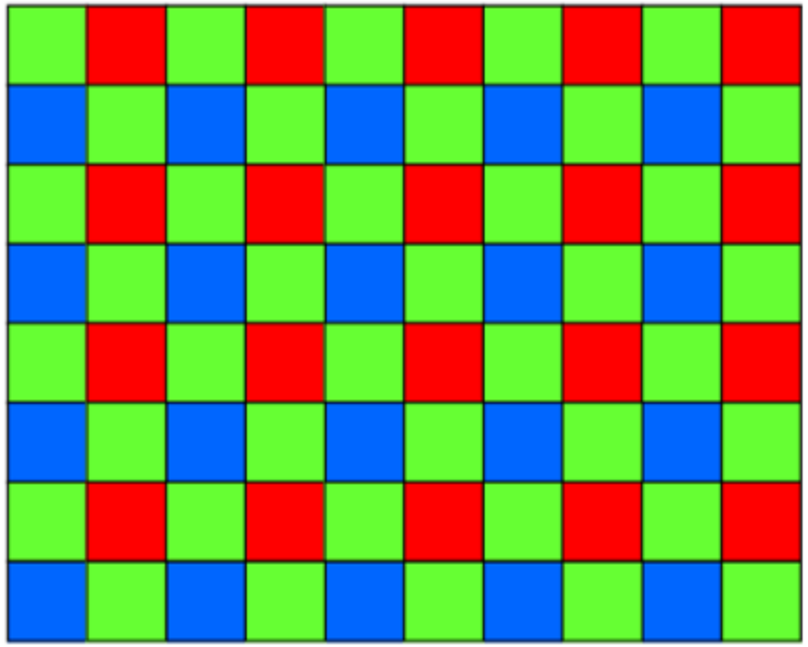


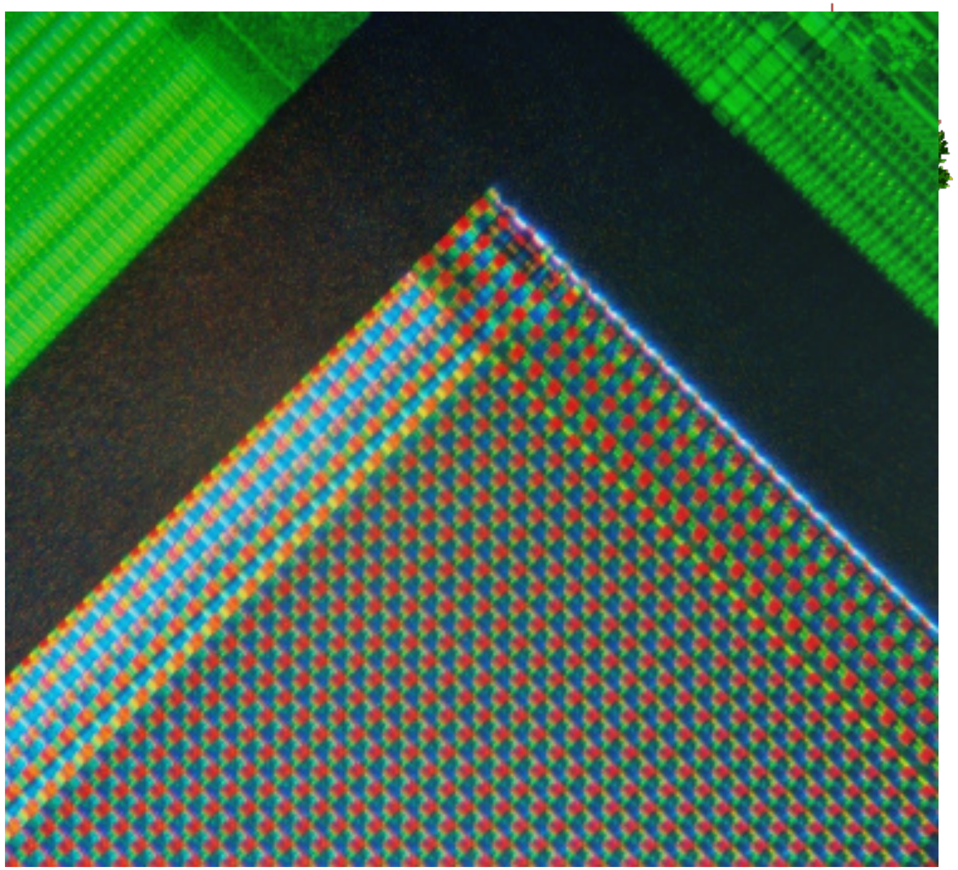
votes

- Peak gets fuzzy and hard to locate

Digital Color Camera

- Bayer Pattern:
  - 50% green, 25% red and 25% blue =>
  - RGBG or GRGB or RGGB.
  - 1 Byte per square
  - 4 squared per 1 pixel
  - More green: eyes are more sensitive to green (nature!)





A micrograph of the corner of the photosensor array of a ‘webcam’ digital camera. (Wikimedia)

Computer Vision: Perspective Projection onto the image plane

- To project a 3D scene point P = (x,y,z) [meters] onto the camera image plane p=(u,v) [pixels] we need to consider:
  - Pixelization: size of the pixel and position of the CCD with respect to the optical center
  - Rigid body transformation between camera and scene
- u = v = 0: where z-Axis passes trhough center of lens – z-Azis prependicular to lens (coincident with optical axis)

$$u = \frac{f}{z} \cdot x$$

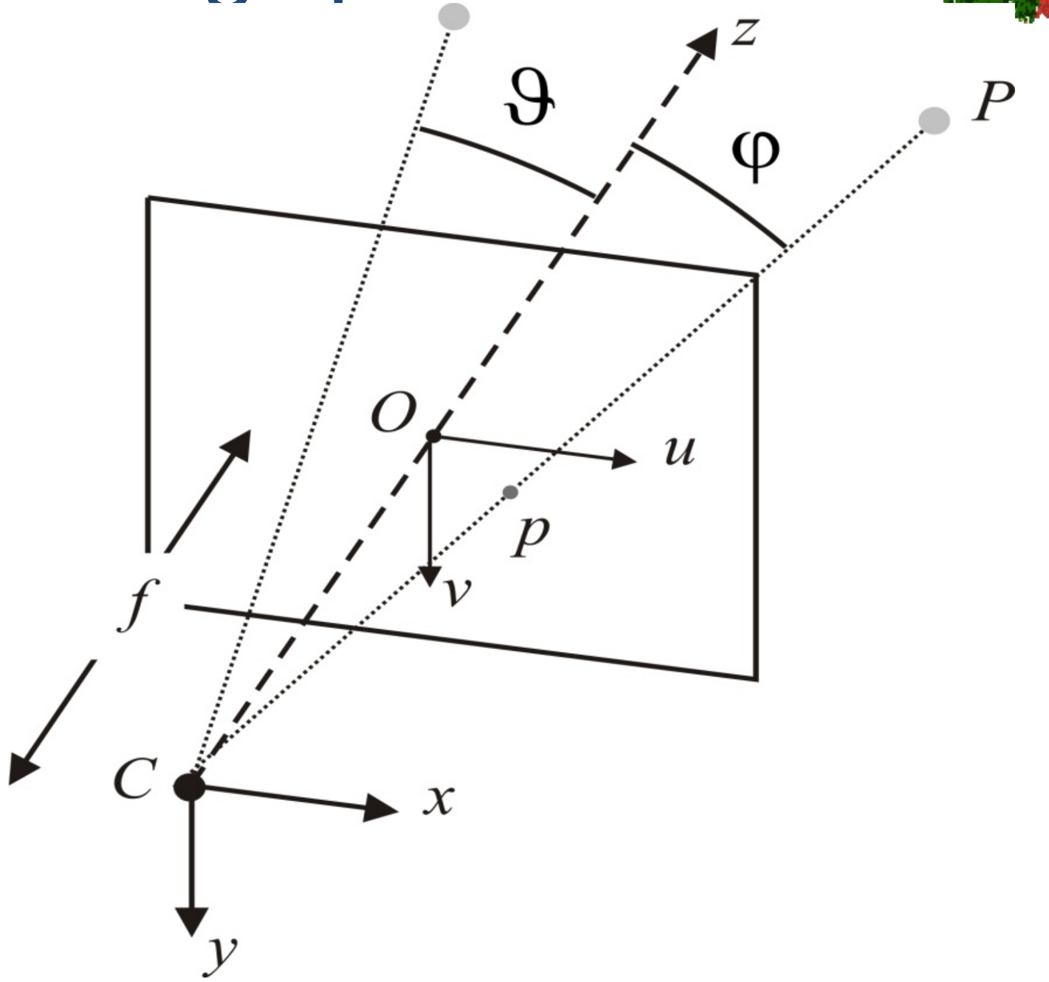
$$v = \frac{f}{z} \cdot y$$

Simple case (without pixelization)

$$u = k_u \frac{f}{z} \cdot x + u_0$$

$$v = k_v \frac{f}{z} \cdot y + v_0$$

With pixelization
u_0, v_0 are the coordinates of the optical center
Ku and Kv are in [pxl/m]



Camera Calibration

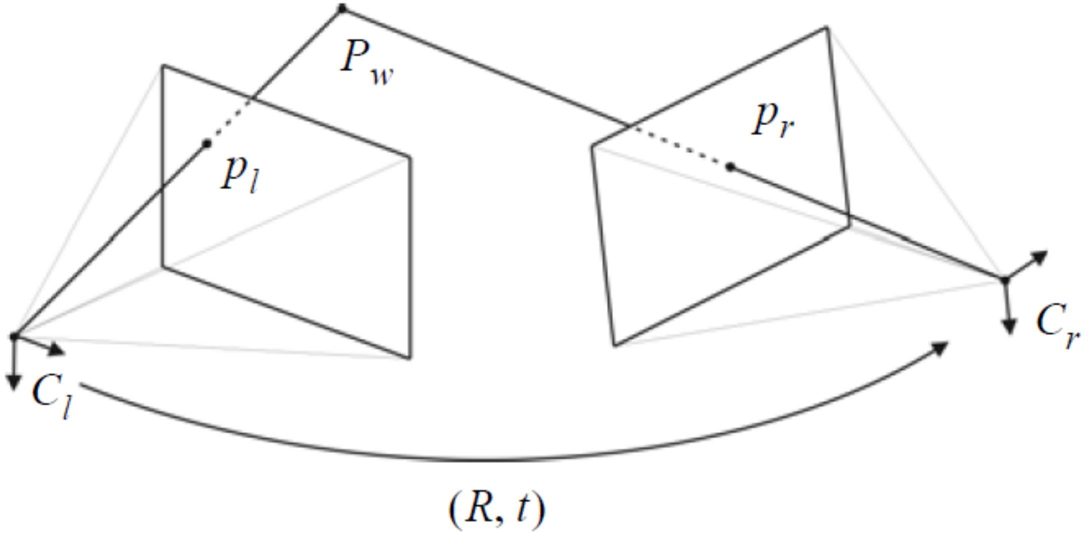
- How many parameters do we need to model a camera?

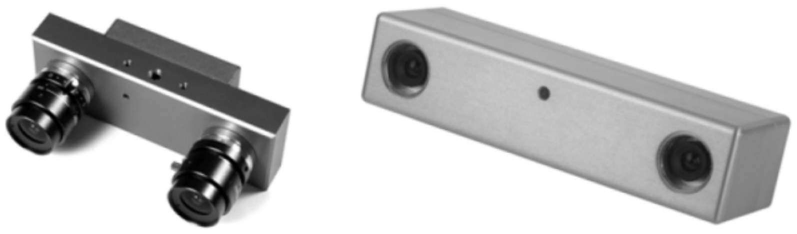
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T \qquad \begin{bmatrix} u_d \\ v_d \end{bmatrix} = (1 + k_1 \rho^2) \cdot \begin{bmatrix} u \\ v \end{bmatrix}$$

- 5 “intrinsic” parameters: α_u, α_v, u_0, v_0, k_1
- Camera pose?
- 6 “extrinsic” parameters (or 0 if the world and the camera frames coincide)

Stereo Vision – the general case

- Two identical cameras do not exist in nature!
- Aligning both cameras on a horizontal axis is very hard, also with the most expensive stereo cameras!



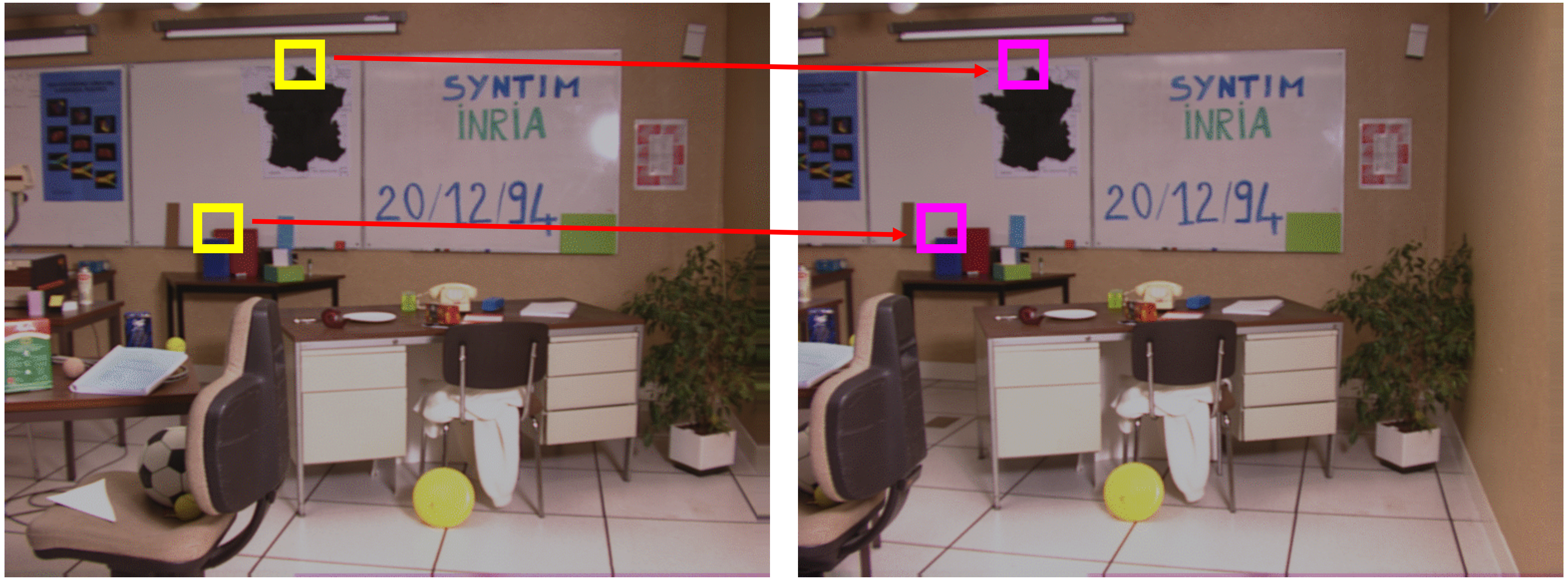


- In order to be able to use a stereo camera, we need first to estimate the relative pose between the cameras, that is, **Rotation** and **Translation**
- However, as the two cameras are not identical, we need to estimate:

focal length, **image center**, **radial distortion**

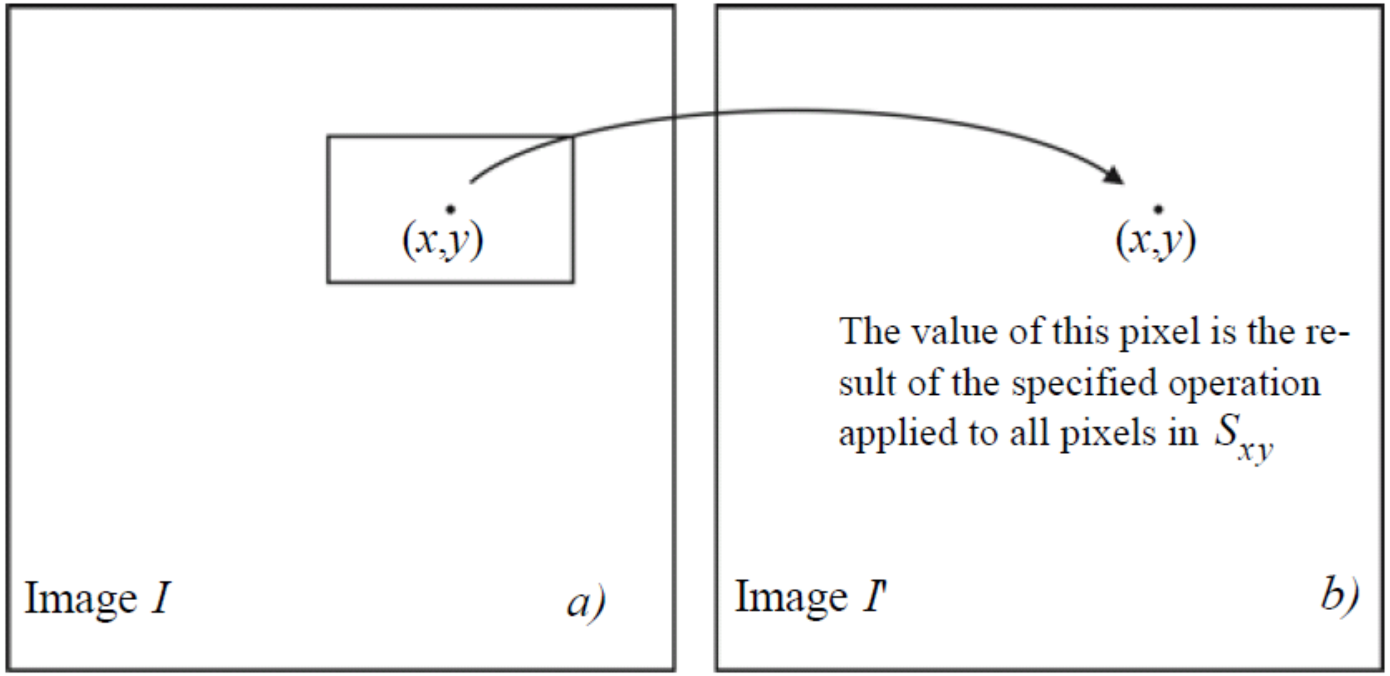
Stereo Vision: Correspondence Problem

- Matching between points in the two images which are projection of the same 3D real point
- Correspondence search could be done by comparing the observed points with all other points in the other image. Typical similarity measures are the Correlation and image Difference.
- This image search can be computationally very expensive! Is there a way to make the correspondence search 1 dimensional?



Spatial filters

- Let *Sxy* denote the set of coordinates of a neighborhood centered on an arbitrary point (x,y) in an image *I*
- Spatial filtering generates a corresponding pixel at the same coordinates in an output image *I'* where the value of that pixel is determined by a specified operation on the pixels in *Sxy*



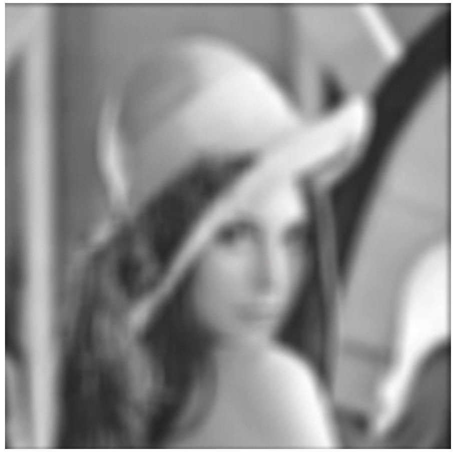
- For example, an averaging filter is: $$I' = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} I(r,c)$$

Smoothing filters (1)

- A constant averaging filter yields the standard average of all the pixels in the mask. For a 3x3 mask this writes:

$$w = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- where notice that all the coefficients sum to 1. This normalization is important to keep the same value as the original image if the region by which the filter is multiplied is uniform.



This example was generated with a 21x21 mask

Smoothing filters (2)

- A Gaussian averaging write as

$$G_\sigma(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

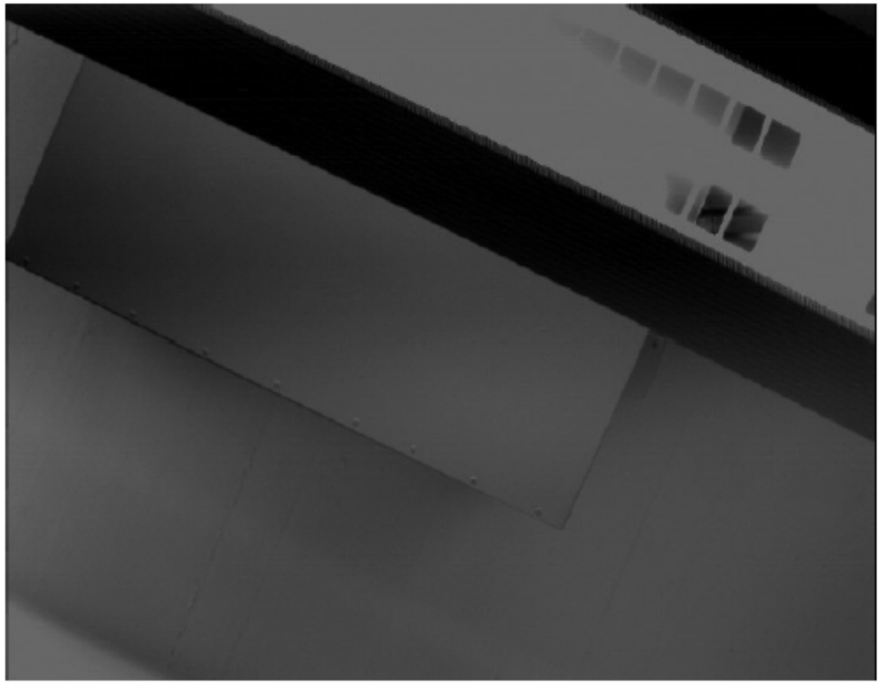
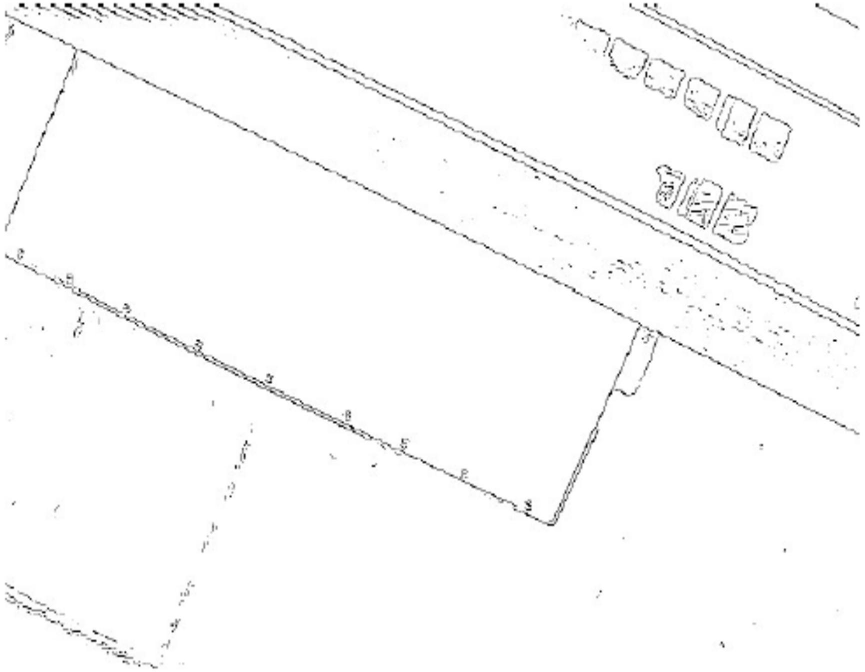
- To generate, say, a 3x3 filter mask from this function, we sample it about its center. For example, with σ=0.85, we get

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

- Very popular: Such low-pass filters effectively removes high-frequency noise =>
- First derivative and especially the second derivative of intensity far more stable
- Gradients and derivatives very important in image processing =>
- Gaussian smoothing preprocessing popular first step in computer vision algorithms

Edge Detection

- Ultimate goal of edge detection
  - an idealized line drawing.
- Edge contours in the image correspond to important scene contours.

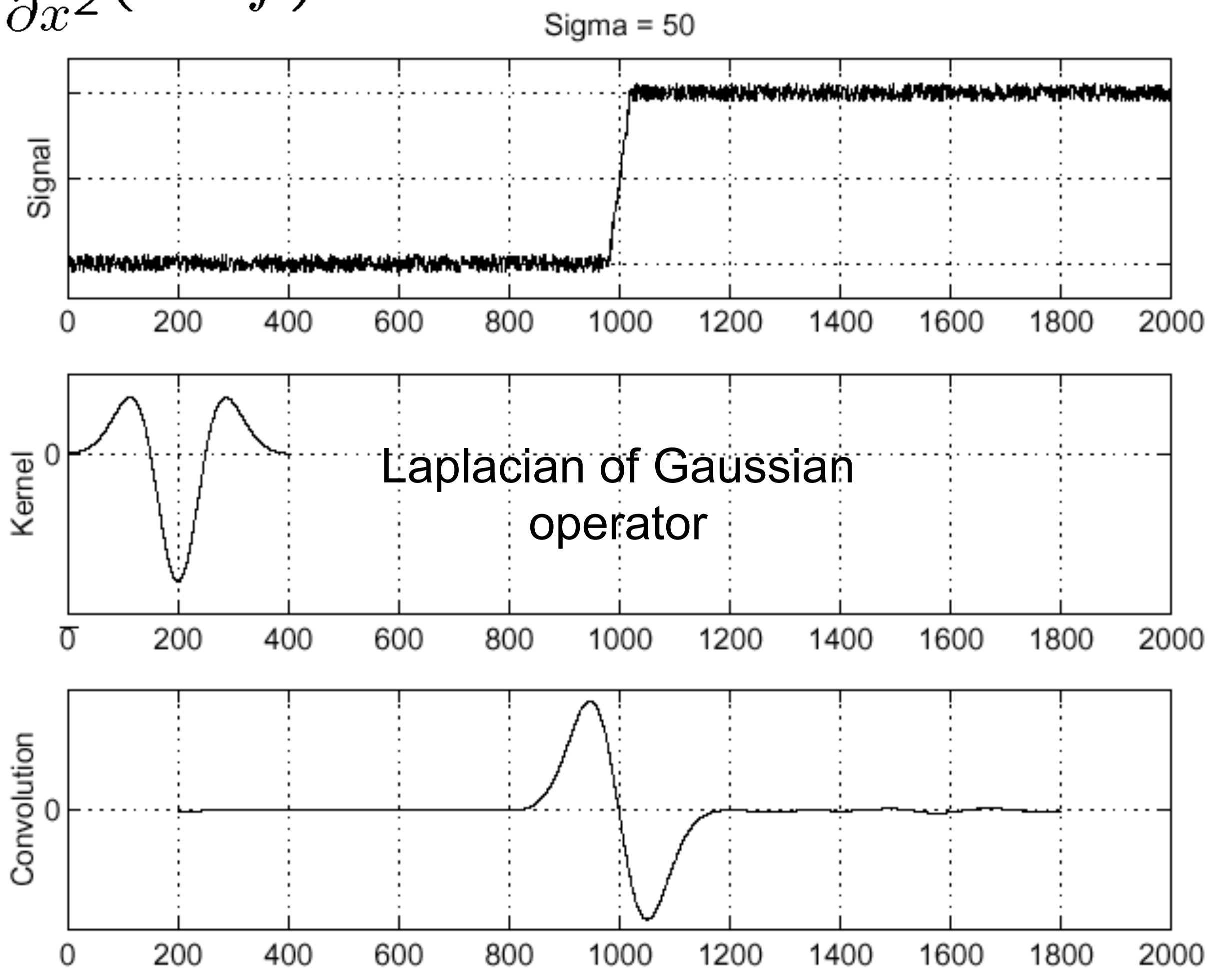
The Canny Edge Detector

- Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$



- Where is the edge?
- Zero-crossings of bottom graph

The Sobel edge detector

thinning

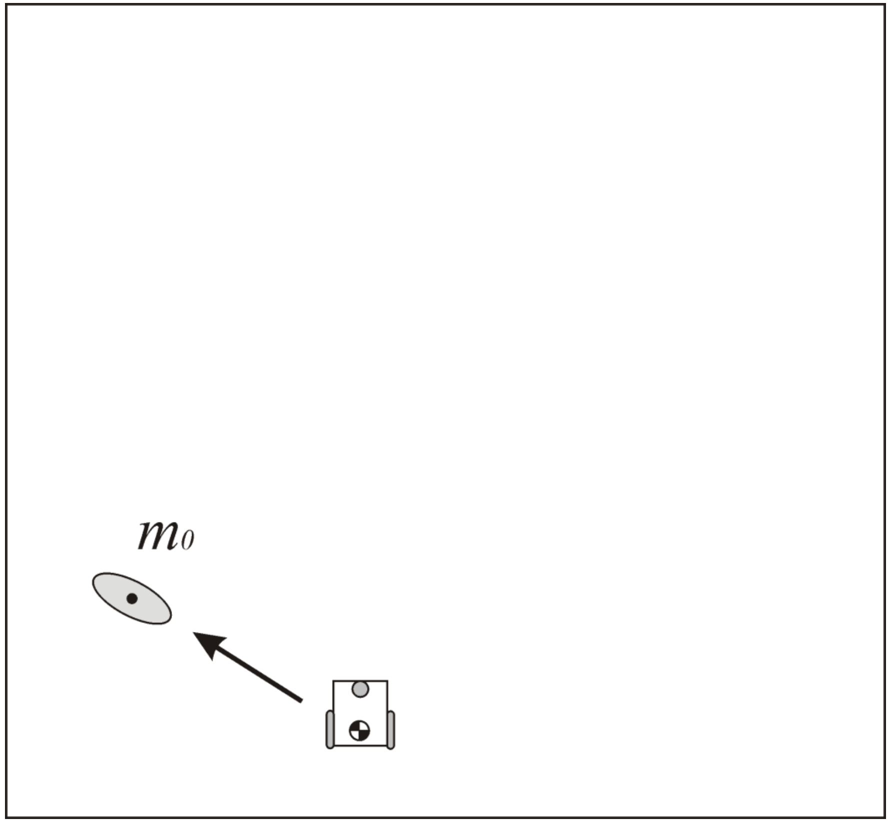
(non-maxima suppression)

IMAGE FEATURES

- Lines
- Points
  - Harris
  - SIFT

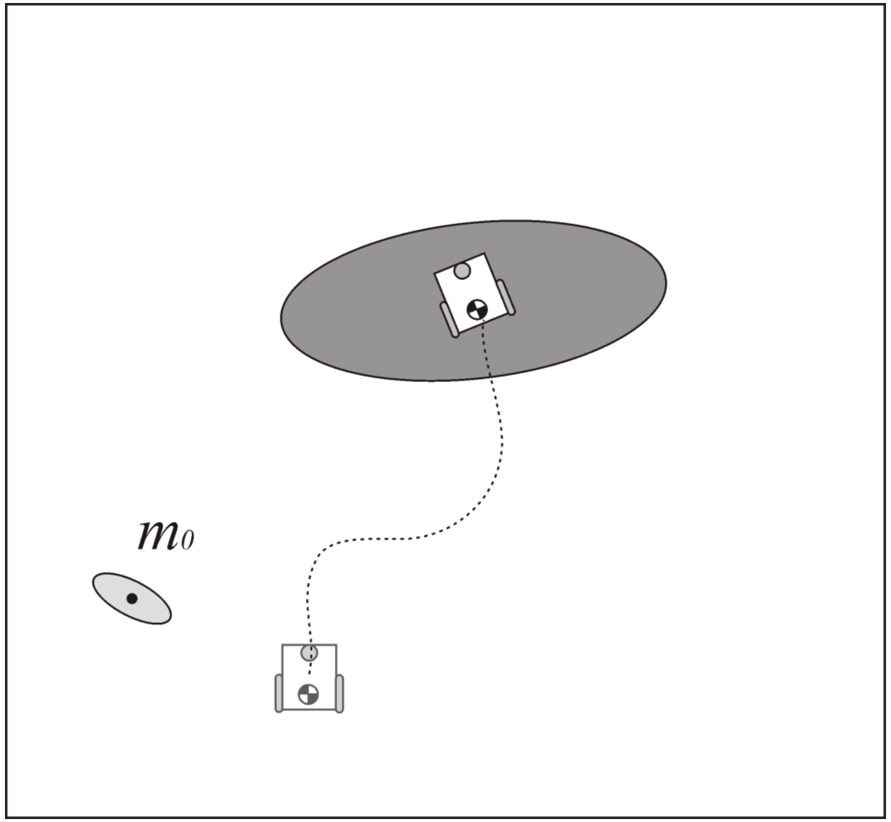
SLAM overview

- Let us assume that the robot uncertainty at its initial location is zero.
- From this position, the robot observes a feature which is mapped with an uncertainty related to the exteroceptive sensor error model



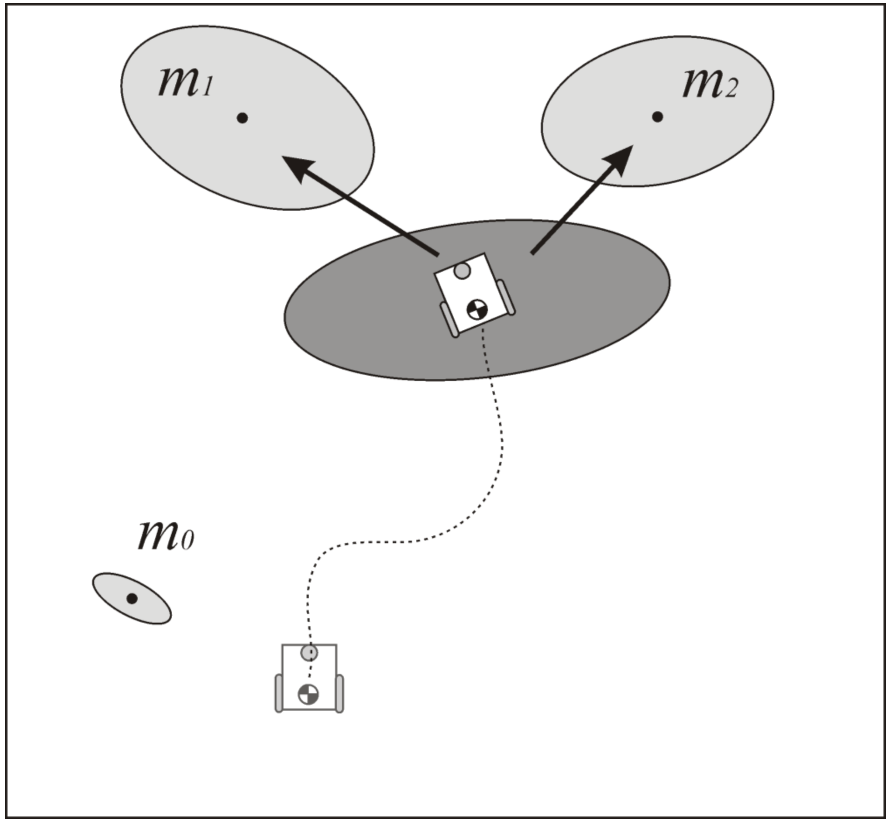
SLAM overview

- As the robot moves, its pose uncertainty increases under the effect of the errors introduced by the odometry



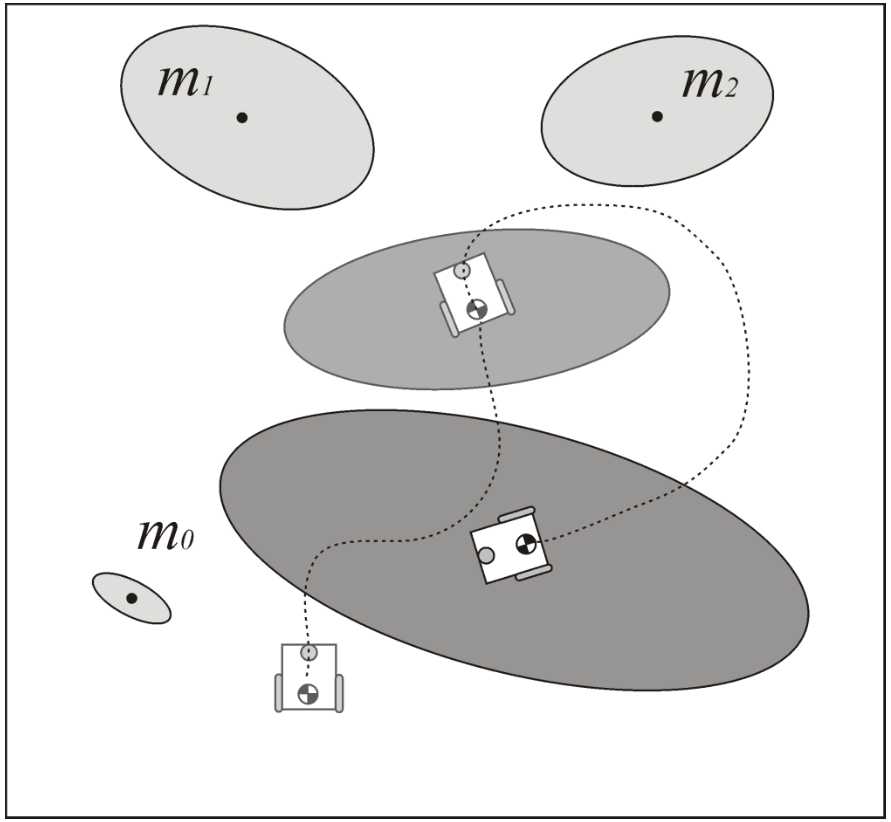
SLAM overview

- At this point, the robot observes two features and maps them with an uncertainty which results from the combination of the measurement error with the robot pose uncertainty
- From this, we can notice that the map becomes correlated with the robot position estimate. Similarly, if the robot updates its position based on an observation of an imprecisely known feature in the map, the resulting position estimate becomes correlated with the feature location estimate.



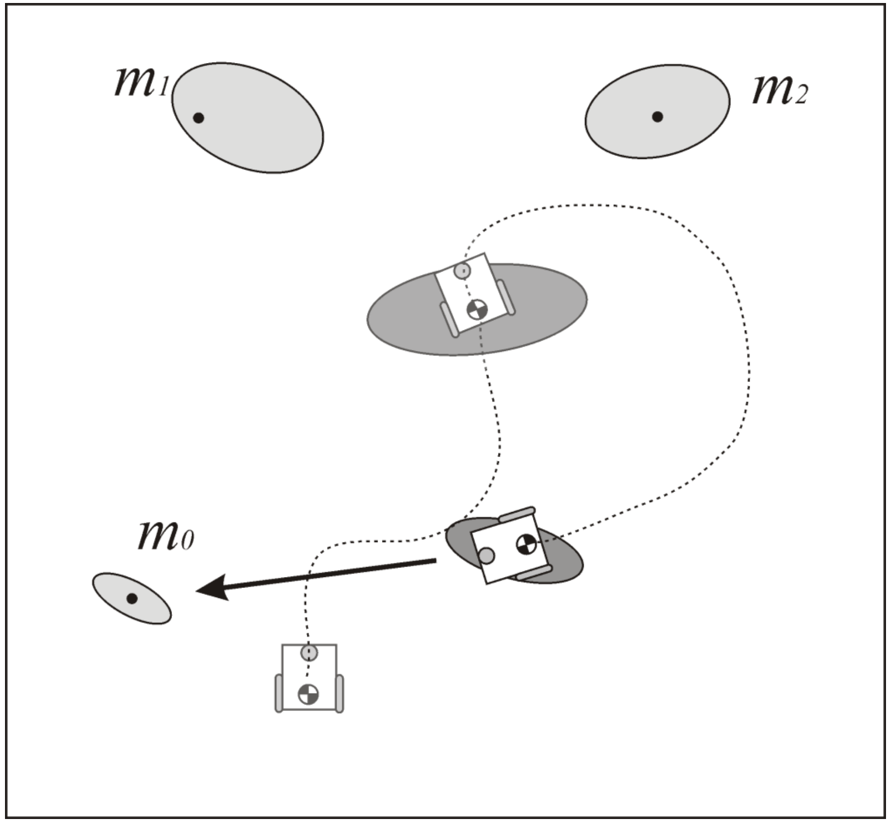
SLAM overview

- The robot moves again and its uncertainty increases under the effect of the errors introduced by the odometry



SLAM overview

- In order to reduce its uncertainty, the robot must observe features whose location is relatively well known. These features can for instance be landmarks that the robot has already observed before.
- In this case, the observation is called *loop closure detection.*
- When a loop closure is detected, the robot pose uncertainty shrinks.
- At the same time, the map is updated and the uncertainty of other observed features and all previous robot poses also reduce

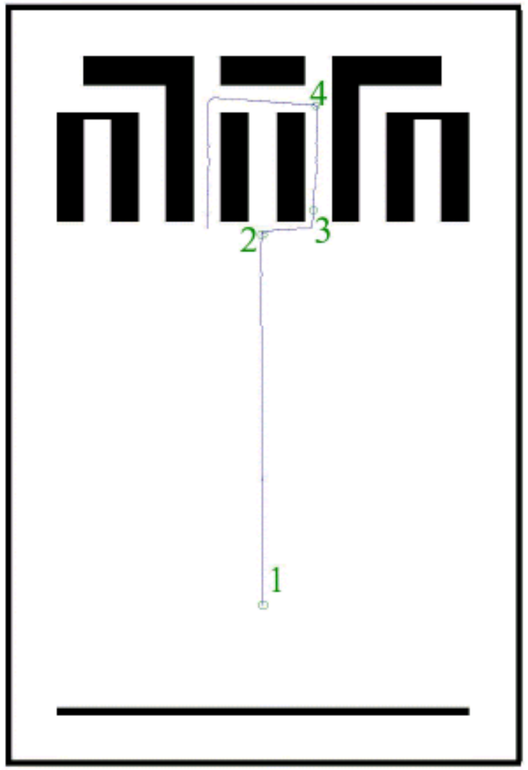


The Three SLAM paradigms

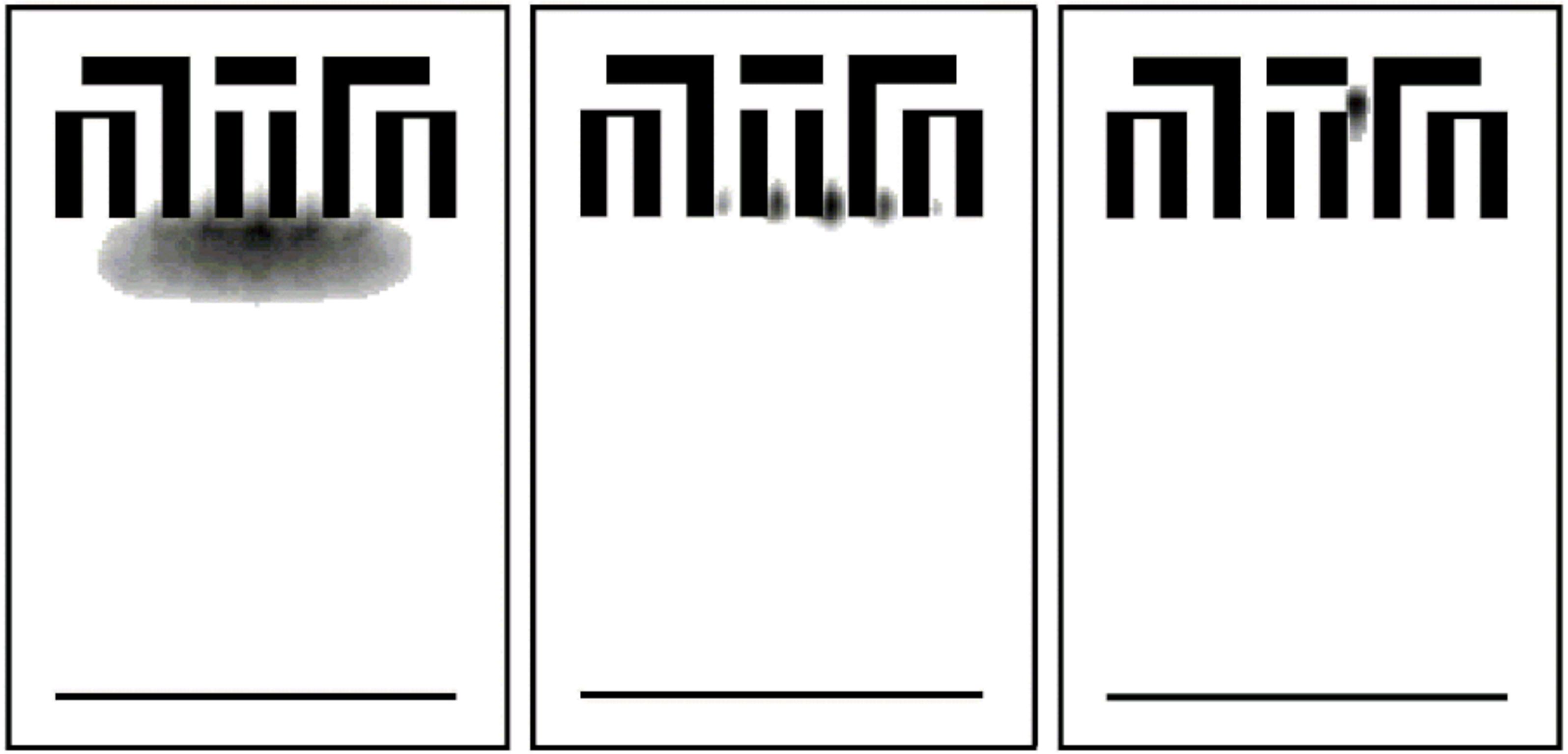
- Most of the SLAM algorithms are based on the following three different approaches:
  - Extended Kalman Filter SLAM: (called EKF SLAM)
  - Particle Filter SLAM: (called FAST SLAM)
  - Graph-Based SLAM

Grid-based Representation - Multi Hypothesis

Courtesy of W. Burgard



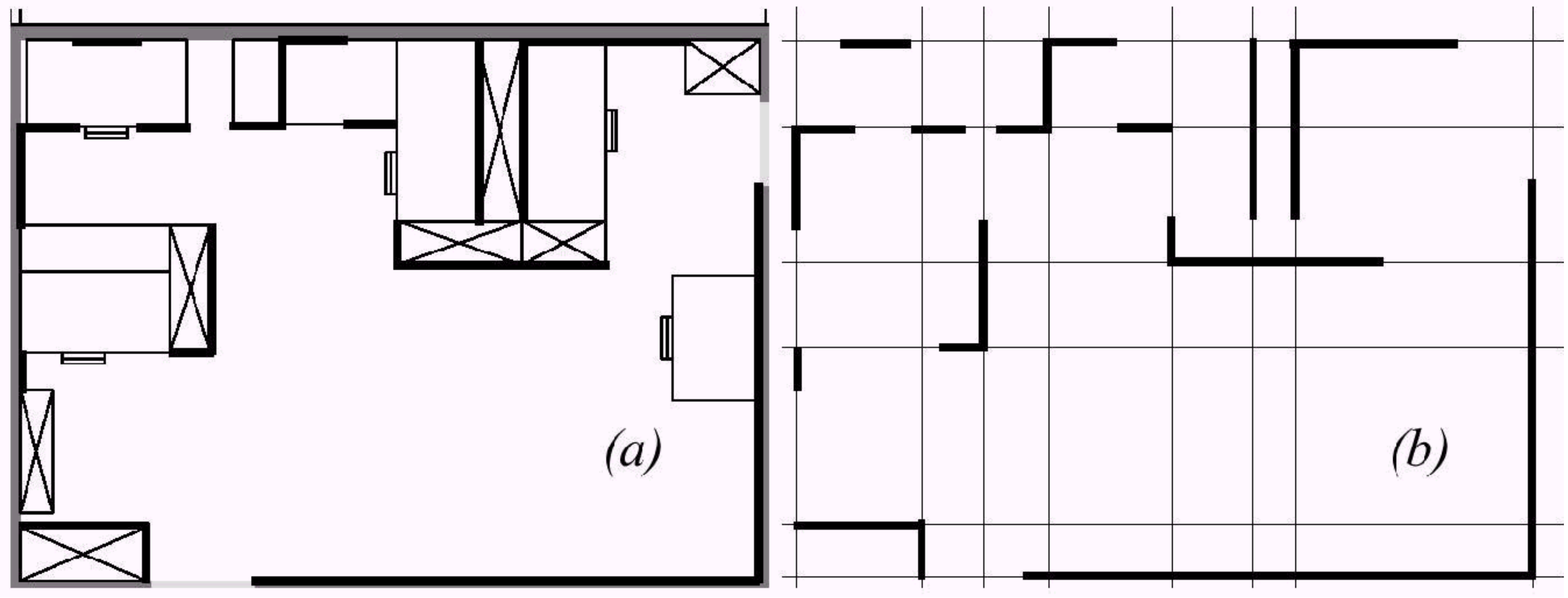
Path of the robot



Belief states at positions 2, 3 and 4

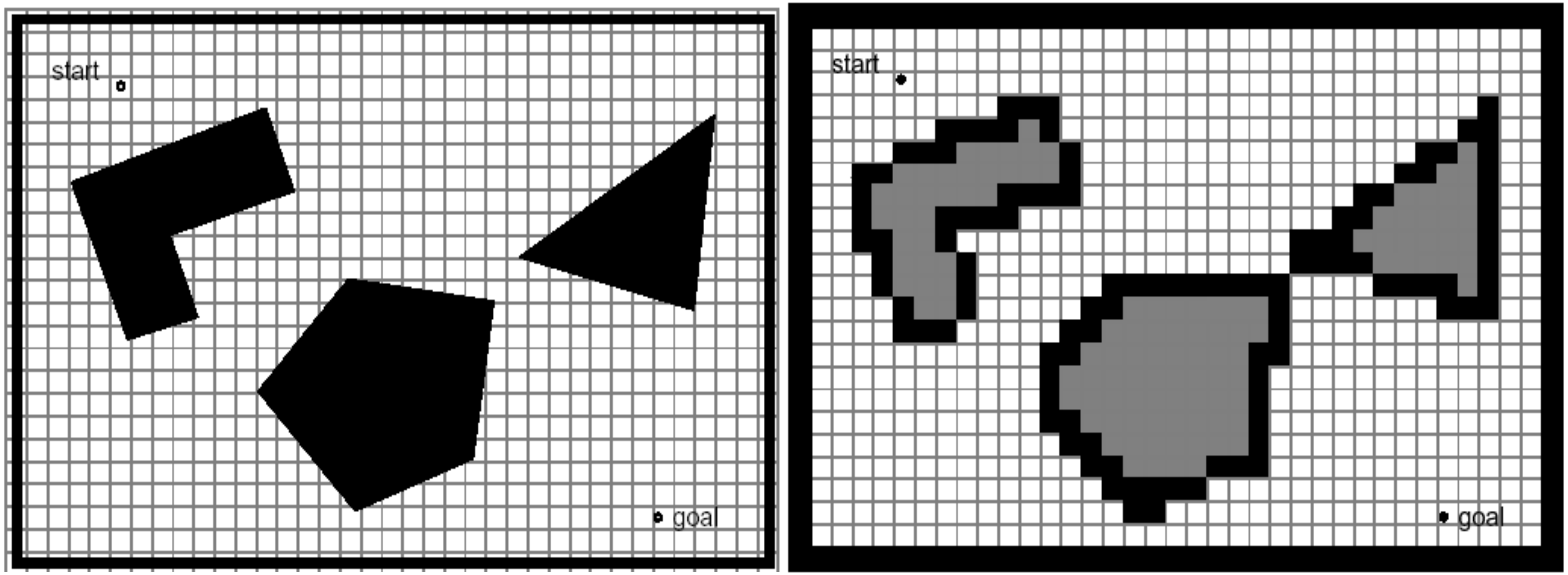
Map Representation: Continuous Line-Based

a) Architecture map
b) Representation with set of finite or infinite lines



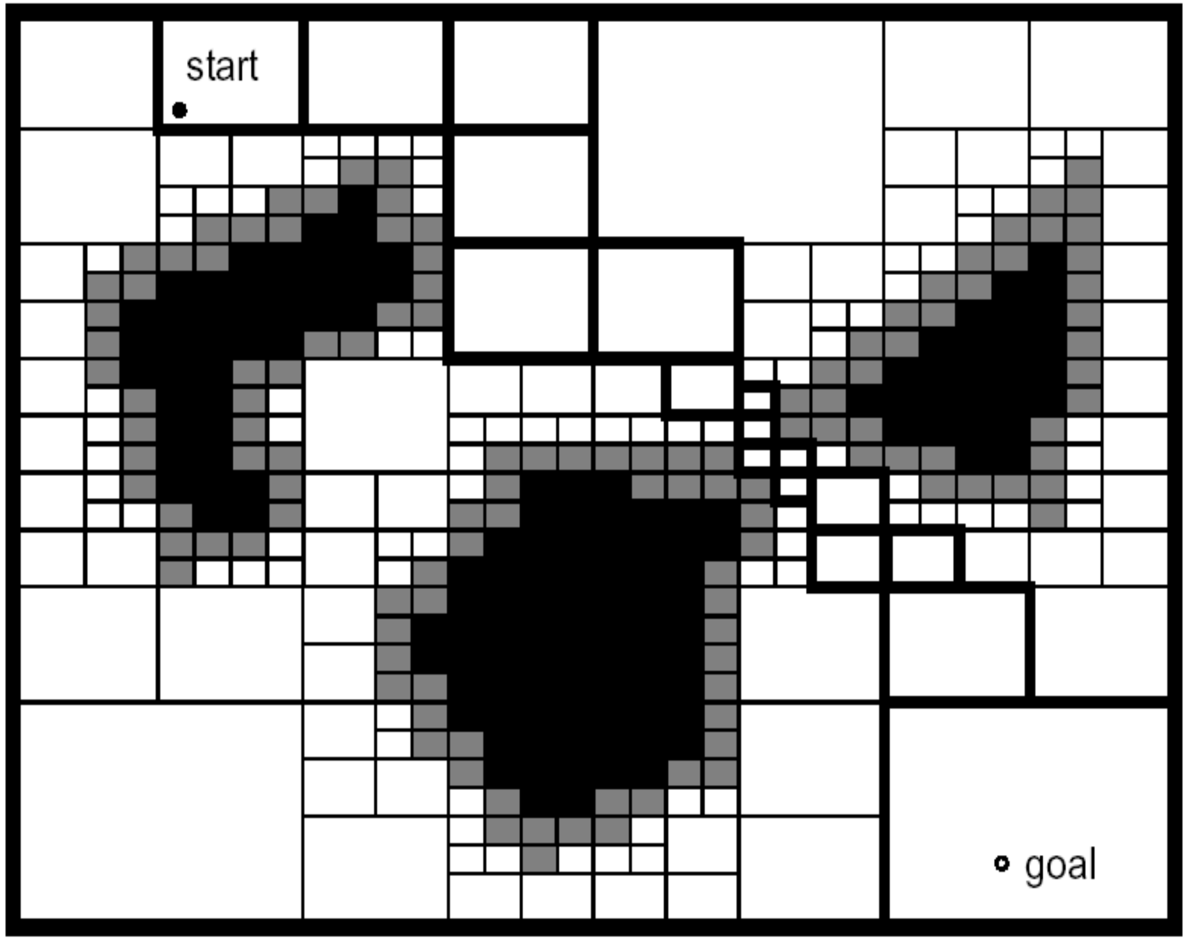
Map Representation: Approximate cell decomposition (1)

- Fixed cell decomposition
  - Narrow passages disappear



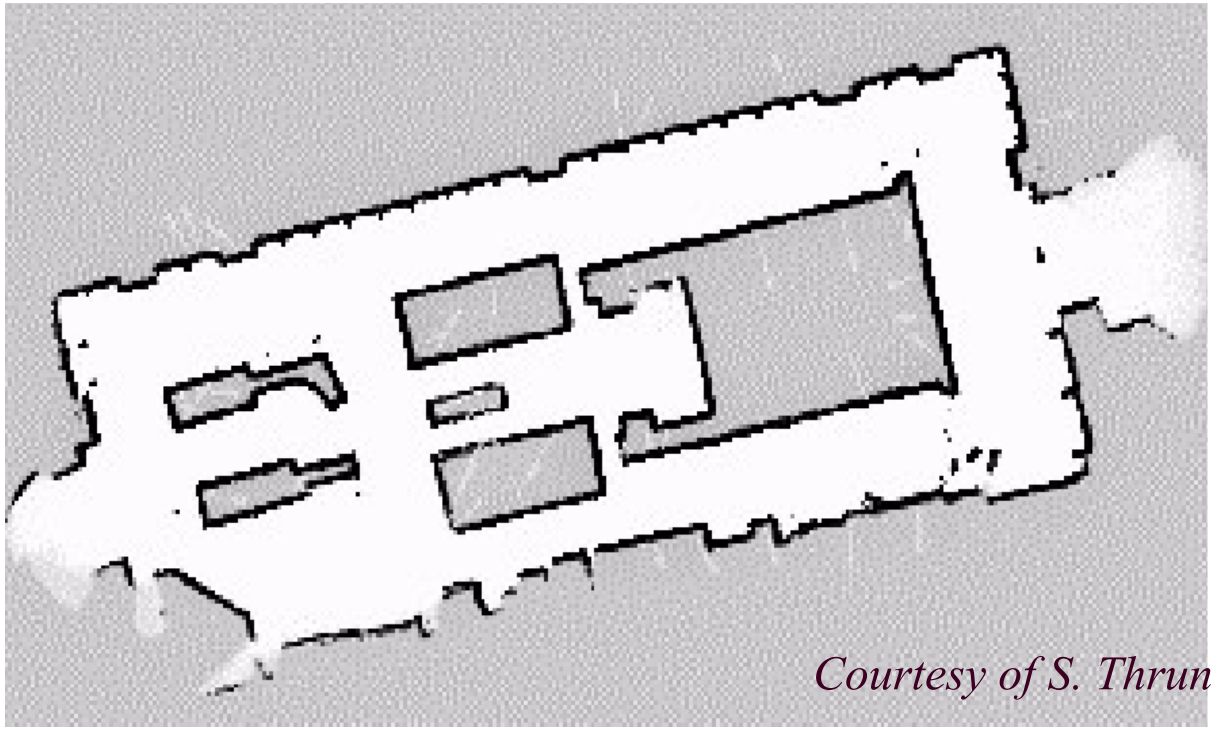
Map Representation: Adaptive cell decomposition (2)

- For example: Quadtree



Map Representation: Occupancy grid

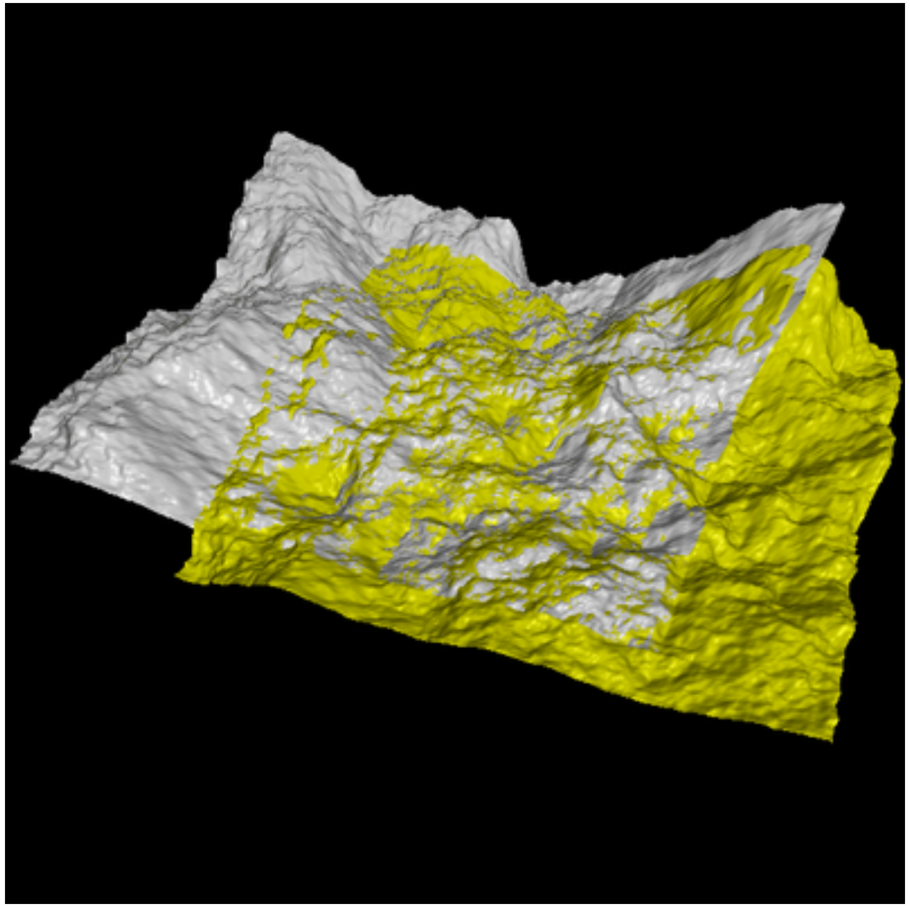
- Fixed cell decomposition: occupancy grid example
  - In occupancy grids, each cell may have a counter where 0 indicates that the cell has not been hit by any ranging measurements and therefore it is likely free-space. As the number of ranging strikes increases, the cell value is incremented and, above a certain threshold, the cell is deemed to be an obstacle
  - The values of the cells are discounted when a ranging strike travels through the cell. This allows us to represent “transient” (dynamic) obstacles



Courtesy of S. Thrun

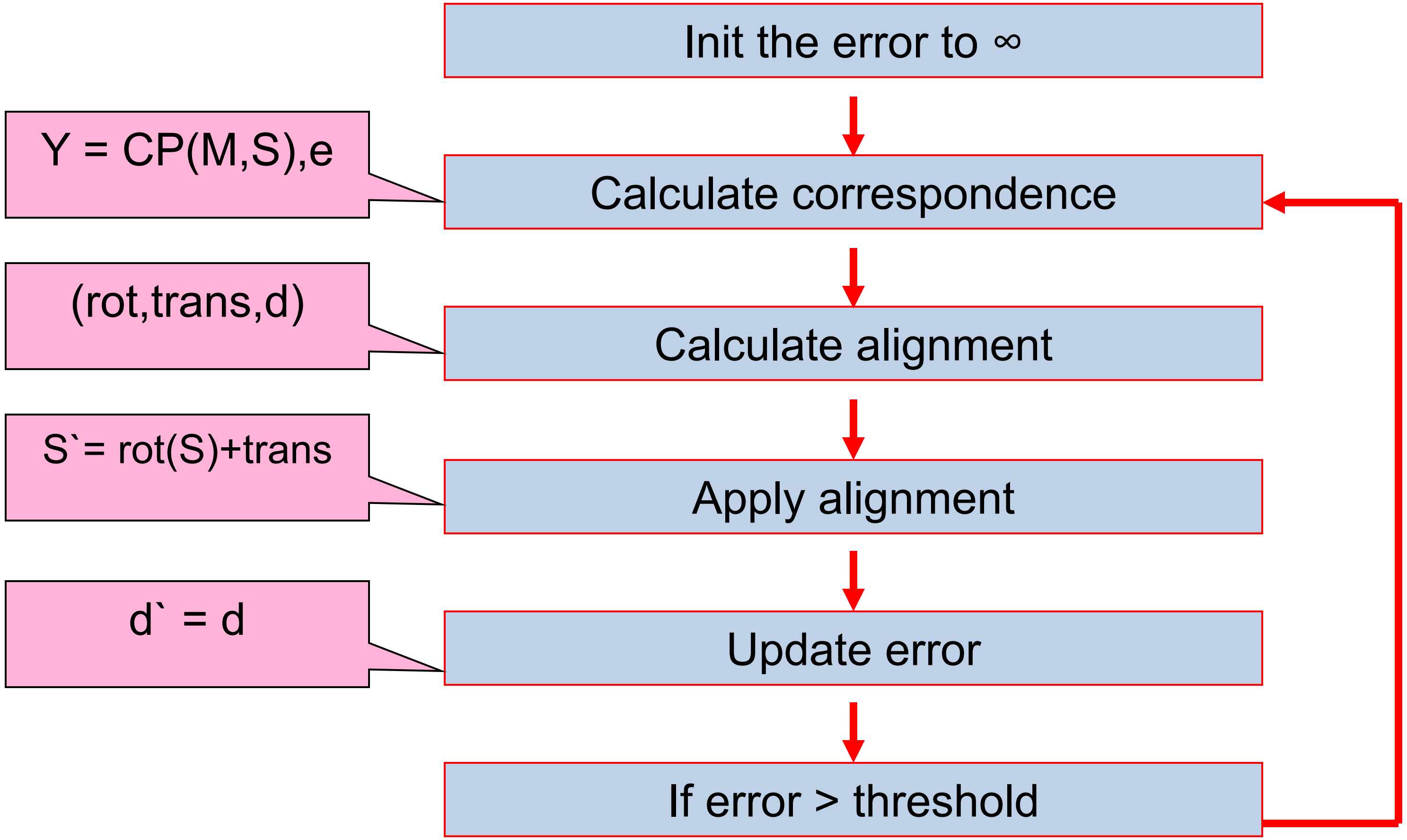
ICP: Iterative Closest Points Algorithm

- Align two partially-overlapping point sets (2D or 3D)
- Given initial guess for relative transform



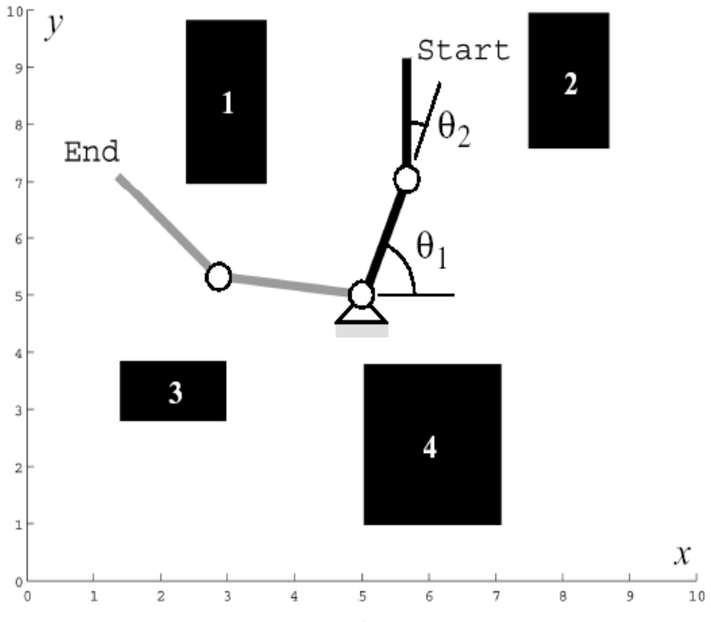
Material derived from Ronen Gvili : *www.cs.tau.ac.il/~dcor/Graphics/adv-slides/**ICP**.ppt*

The Algorithm



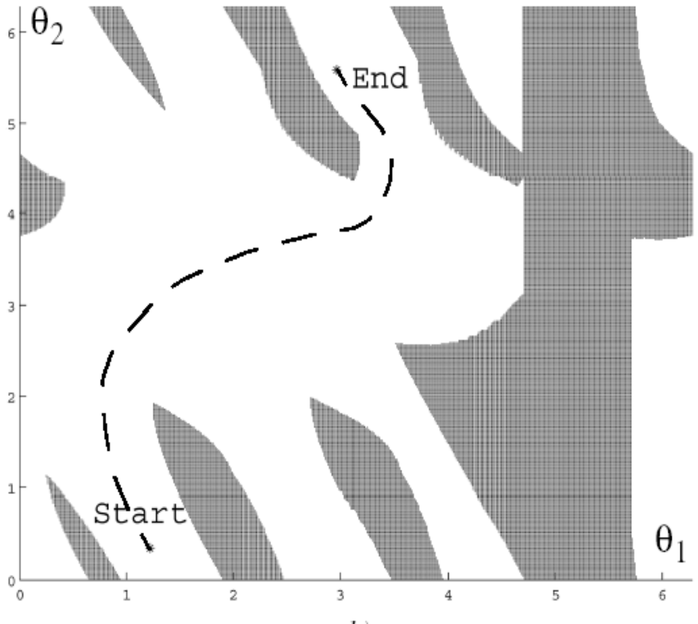
Work Space (Map) → Configuration Space

- State or configuration q can be described with k values q_i



a)

Work Space

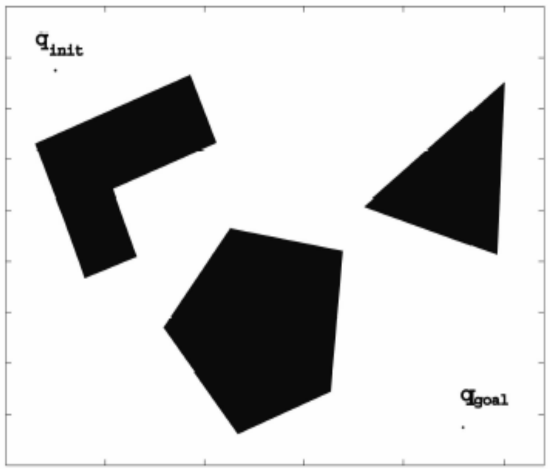


b)

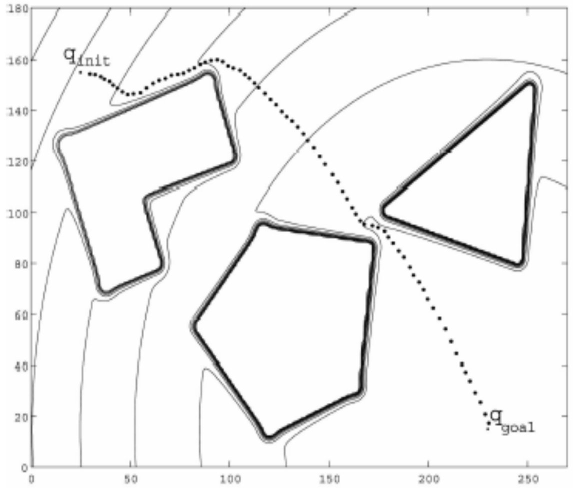
Configuration Space:
the dimension of this
space is equal to the Degrees of Freedom (DoF)
of the robot

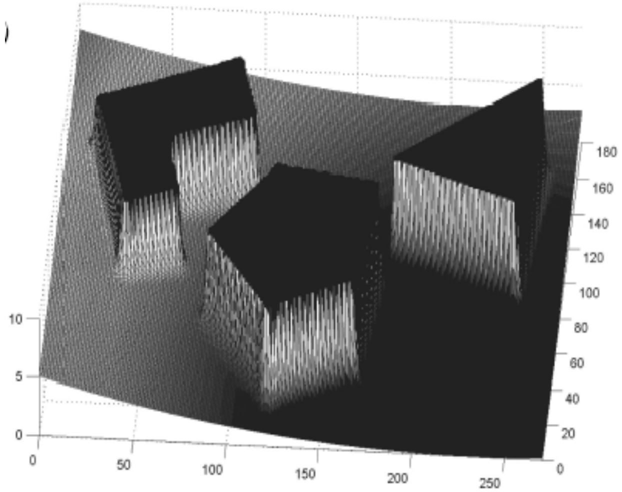
- What is the configuration space of a mobile robot?

Potential Field Path Planning Strategies



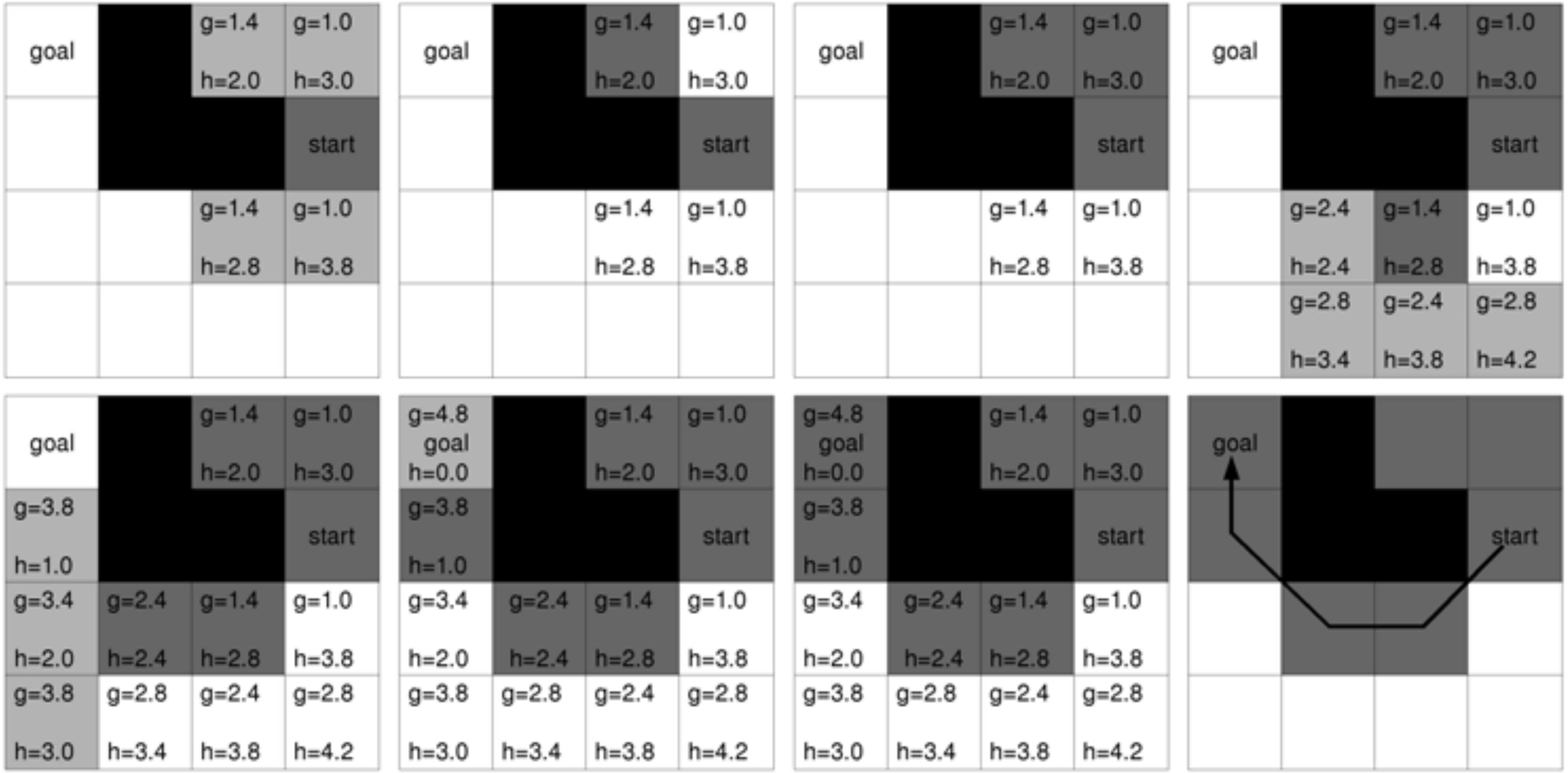
- Robot is treated as a *point under the influence* of an artificial potential field.
- Operates in the continuum
  - Generated robot movement is similar to a ball rolling down the hill
  - Goal generates attractive force
  - Obstacle are repulsive forces





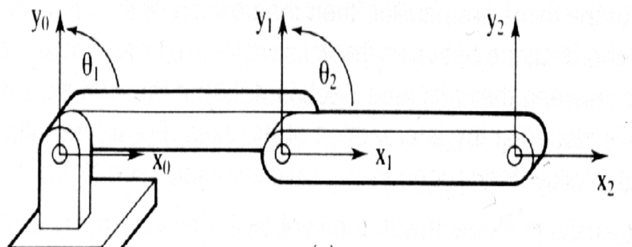
Graph Search Strategies: A* Search

- Similar to Dijkstra‘s algorithm, except that it uses a heuristic function h(n)
- f(n) = g(n) + ε h(n)



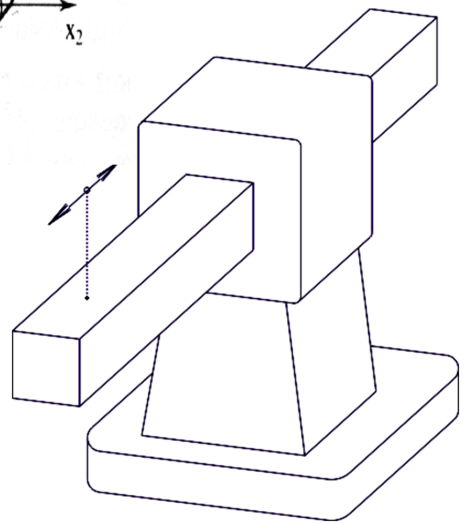
Robot Arm: Joints

- Revolute Joint: 1DOF

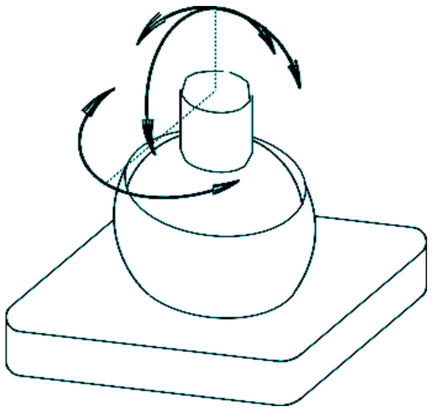


(a)

- Prismatic Joint/ Linear Joint: 1DOF

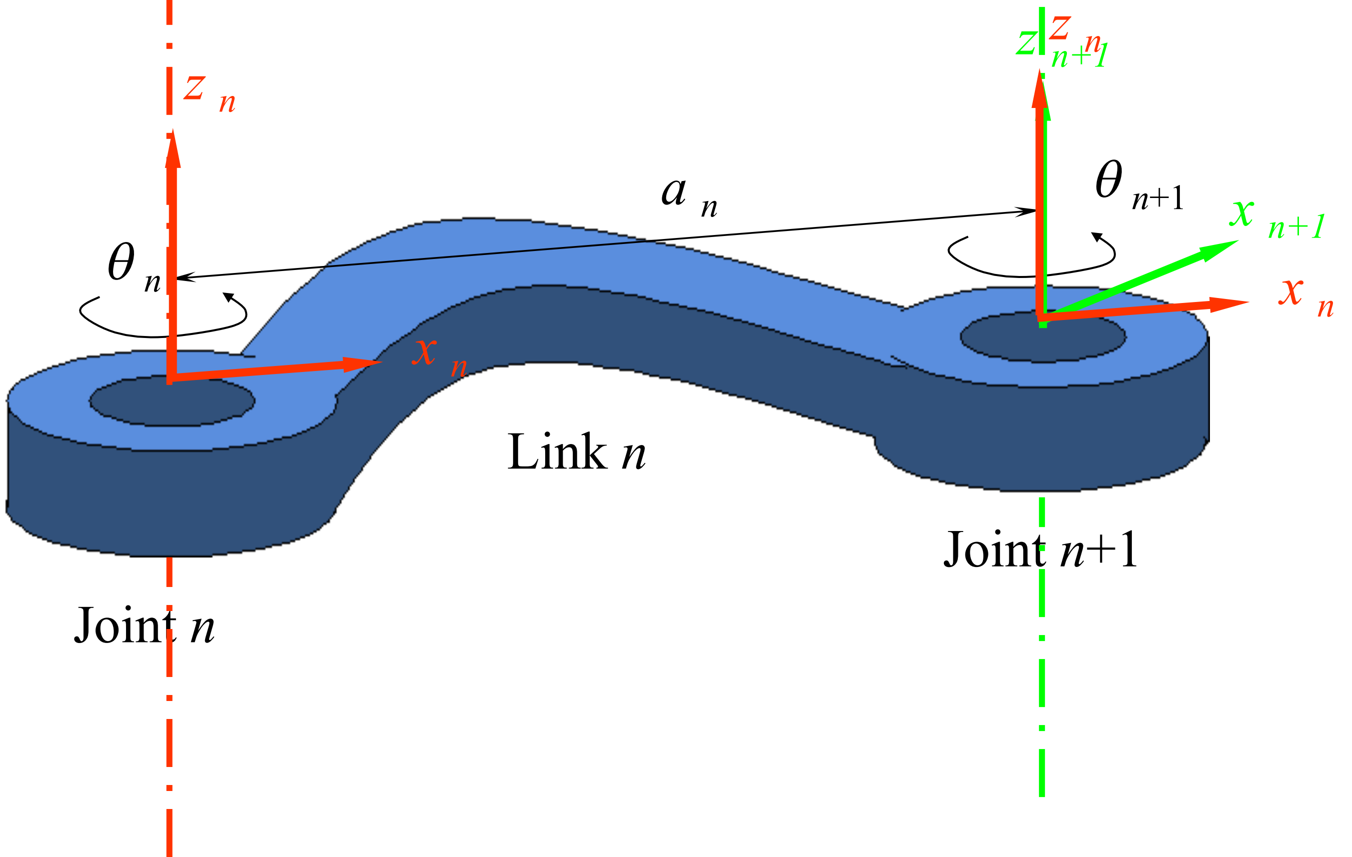


- Spherical Joint: 3DOF



Robot Arm: Link

- A link is considered as a rigid body which defines the relationship between two neighboring joint axes of a manipulator.



Link and Joint Parameters

4 parameters are associated with each link. You can align the two axis using these parameters.

- Link parameters:

a_n the length of the link.

α_n the twist angle between the joint axes.

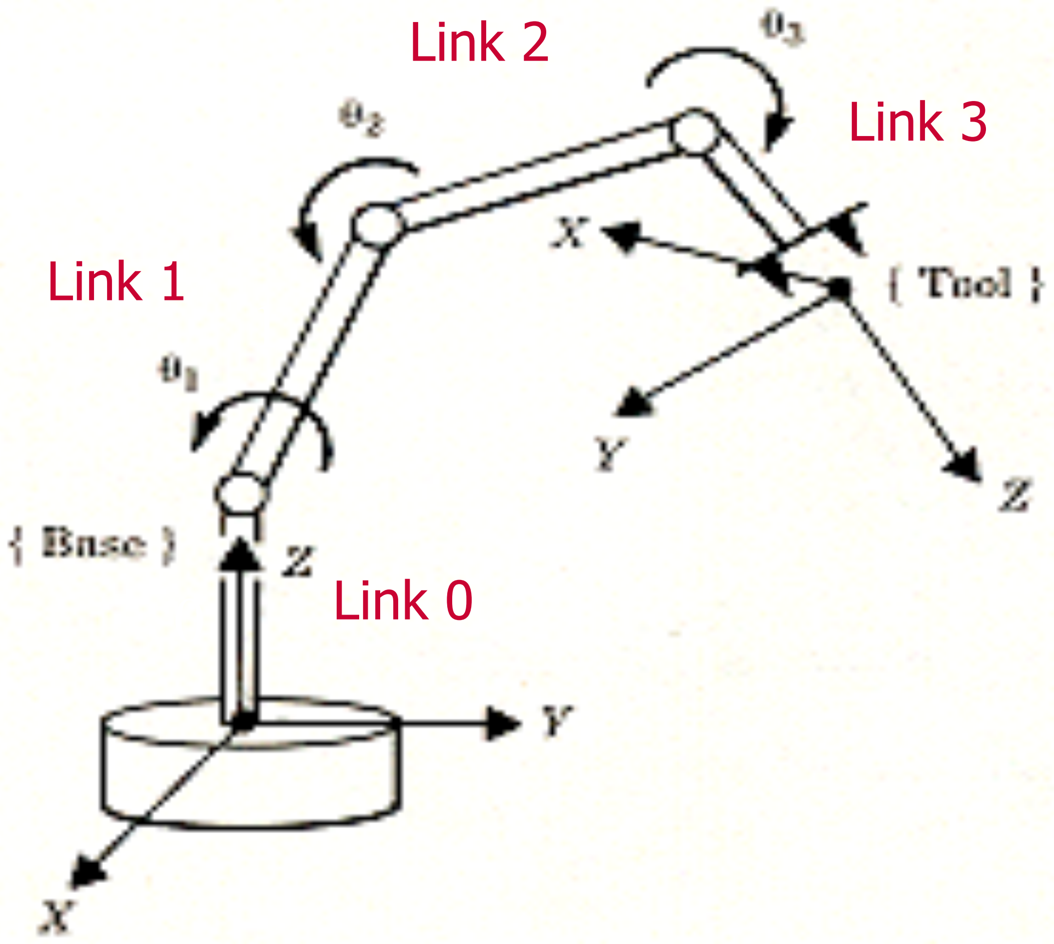
- Joint parameters:

θ_n the angle between the links.

d_n the distance between the links

Links Numbering Convention

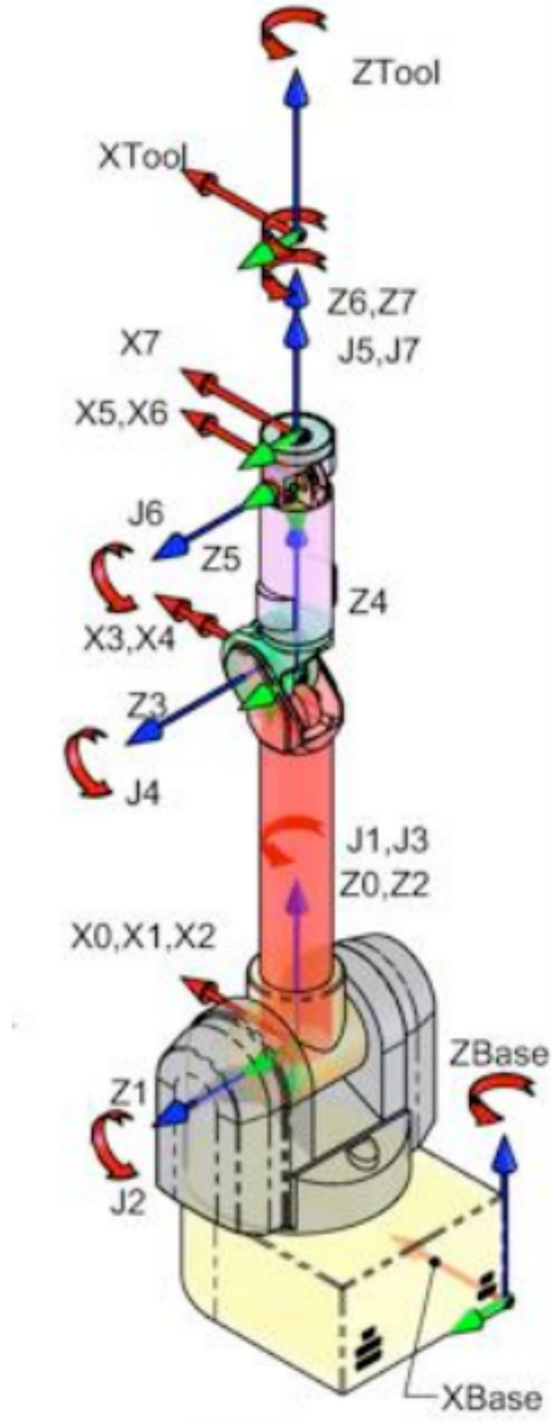
Base of the arm:	**Link-0**
1st moving link:	**Link-1**
.	**.**
.	**.**
.	**.**
Last moving link:	**Link-n**



A 3-DOF Manipulator Arm

Frames

- Choose the base and tool coordinate frame
  - Make your life easy!
- Several conventions
  - Denavit Hartenberg (DH), modified DH, Hayati, etc.



Kinematics

Forward Kinematics (angles to position)

(it is straight-forward -> easy)

What you are given: The length of each link
The angle of each joint

What you can find: The position of any point (i.e. it's (x, y, z) coordinates)

Inverse Kinematics (position to angles)

(more difficult)

What you are given: The length of each link
The position of some point on the robot

What you can find: The angles of each joint needed to obtain that position

Kinematics

Cartesian Space

Tool Frame (E)
(aka End-Effector)
Base Frame (B)

$$ {}^{B}_{E}T = \begin{Bmatrix} {}^{B}_{E}t \\ {}^{B}_{E}R \end{Bmatrix} $$

Forward Kinematics

$$ {}^{B}_{E}T = f(q) $$

Inverse Kinematics

$$ q = f^{-1}({}^{B}_{E}T) $$

Joint Space

Joint 1 = q_1
Joint 2 = q_2
Joint 3 = q_3
...
Joint n = q_n

Rigid body transformation
Between coordinate frames

Linear algebra

Inverse Kinematics (IK)

- Given end effector position, compute required joint angles
- In simple case, analytic solution exists
  - Use trig, geometry, and algebra to solve
- Generally (more DOF) difficult
  - Use Newton's method
  - Often more than one solution exist!

Iterative IK Solutions

- Frequently analytic solution is infeasible
- Use Jacobian
- Derivative of function output relative to each of its inputs
- If y is function of three inputs and one output

$$y = f(x_1, x_2, x_3)$$

$$\delta y = \frac{\delta f}{\partial x_1} \cdot \delta x_1 + \frac{\delta f}{\partial x_2} \cdot \delta x_2 + \frac{\delta f}{\partial x_3} \cdot \delta x_3$$

- Represent Jacobian J(X) as a 1x3 matrix of partial derivatives

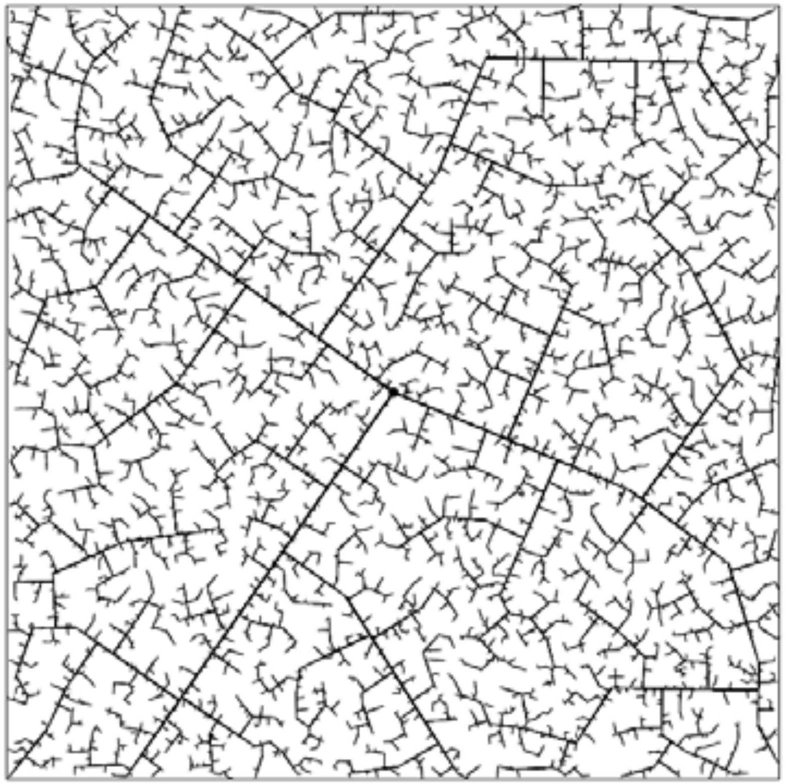
Kinematic Problems for Manipulation

- Reliably position the tip - go from one position to another position
- Don’t hit anything, avoid obstacles
- Make smooth motions
  - at reasonable speeds and
  - at reasonable accelerations
- Adjust to changing conditions -
  - i.e. when something is picked up *respond to the change in weight*

Graph Search Strategies: Randomized Search

- Most popular version is the rapidly exploring random tree (RRT)
  - Well suited for high-dimensional search spaces
  - Often produces highly suboptimal solutions

45 iterations

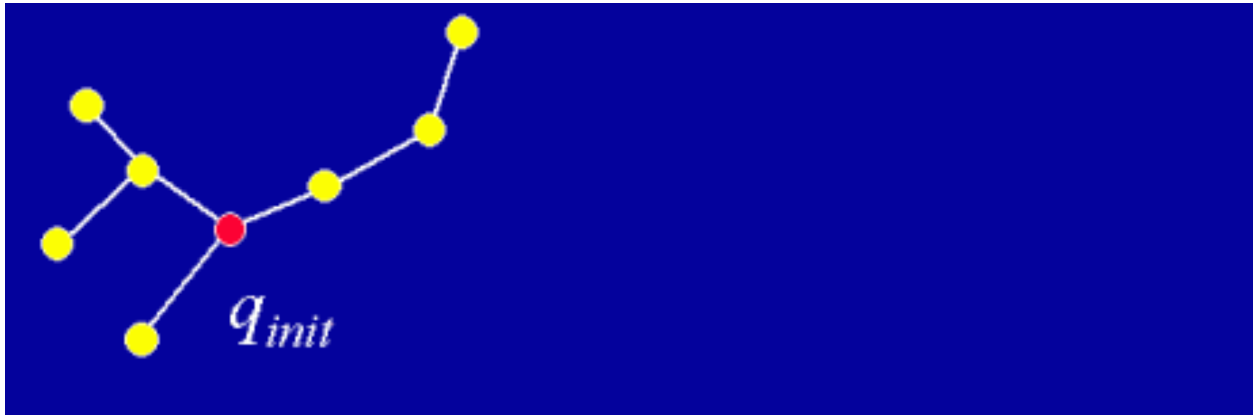


2345 iterations

RRT

BUILD_RRT(q_{init})
$\mathcal{T}$.init(q_{init});
for $k = 1$ **to** K **do**
    $q_{rand} \leftarrow$ RANDOM_CONFIG();
    EXTEND($\mathcal{T}, q_{rand}$);
Return $\mathcal{T}$

EXTEND($\mathcal{T}, q$)
$q_{near} \leftarrow$ NEAREST_NEIGHBOR($q, \mathcal{T}$);
if NEW_CONFIG(q, q_{near}, q_{new}) **then**
    $\mathcal{T}$.add_vertex(q_{new});
    $\mathcal{T}$.add_edge(q_{near}, q_{new});
    if $q_{new} = q$ **then**
        Return *Reached*;
    else
        Return *Advanced*;
Return *Trapped*;



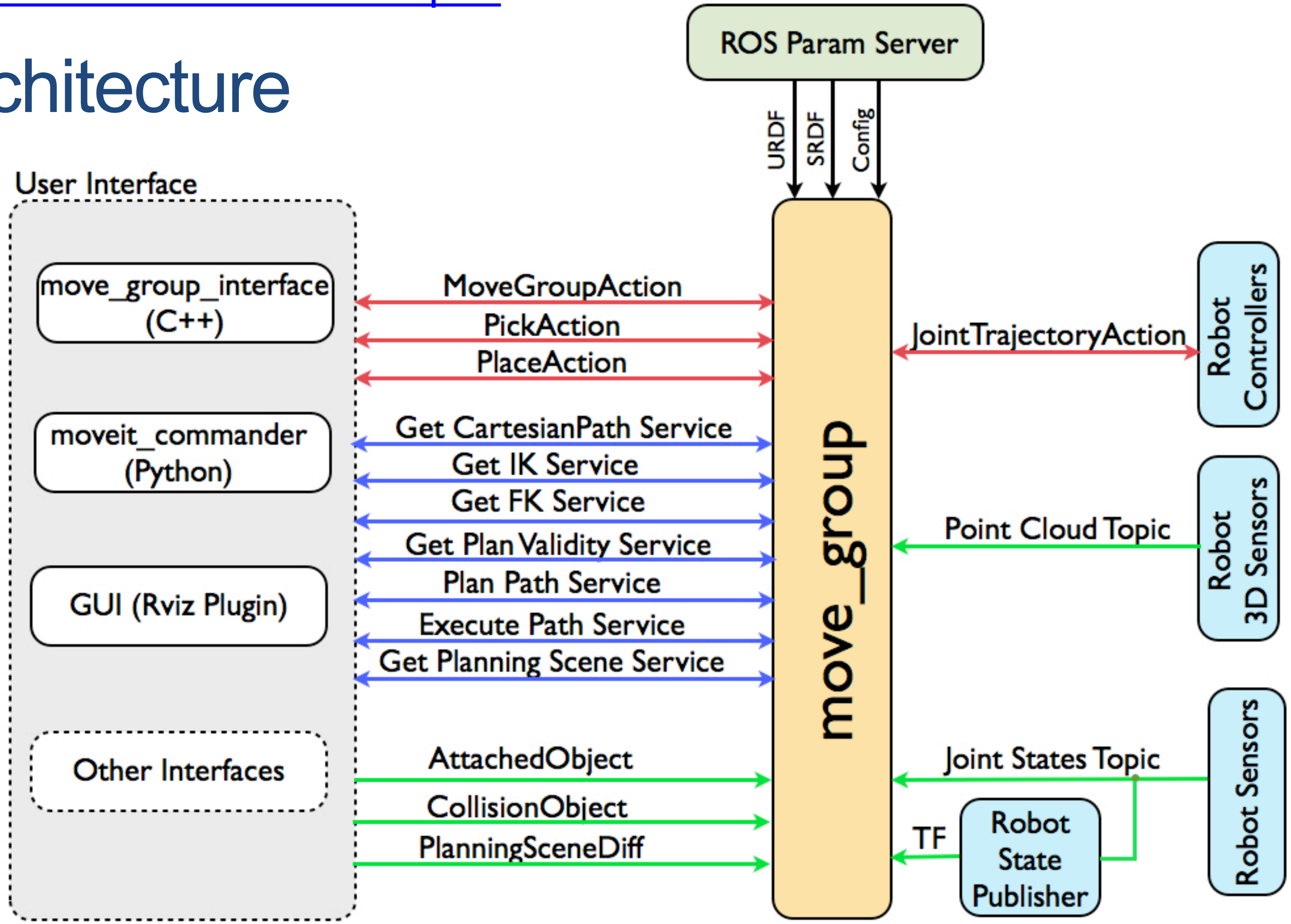
ROS Basics

- Different components, modules, algorithms run in different processes: **nodes**
- Nodes communicate using **messages** (and **services** …)
- Nodes **publish** and **subscribe** to **messages** by using names (**topics**)
- **Messages** are often passed around as shared pointers which are
  - "write protected" using the const keyword
  - The shared pointers take the message type as template argument
  - Shared pointers can be accessed like normal pointers

- Talker/ Listener example

http://moveit.ros.org/documentation/concepts/

System Architecture



QUESTIONS ? ☺