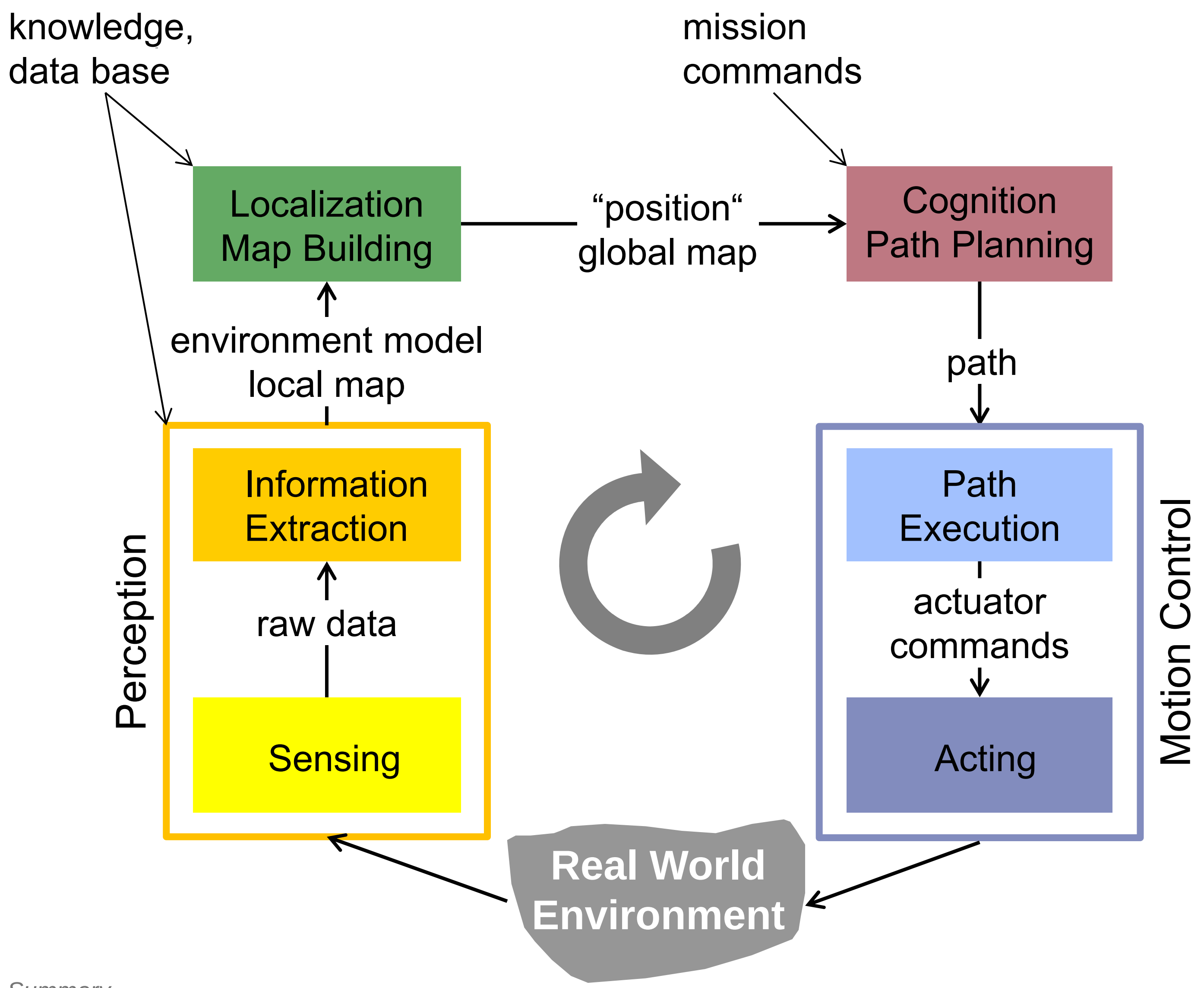
Roland Siegwart
Margarita Chli
Martin Rufli
Davide Scaramuzza

ETH Master Course: 151-0854-00L

Autonomous Mobile Robots

Summary

Lecture Overview
Mobile Robot Control Scheme



Roland Siegwart

Margarita Chli

Martin Rufli

Davide Scaramuzza

ETH Master Course: 151-0854-00L

Autonomous Mobile Robots

Kinematics

Kinematics
Definitions

- Kinematics
  - Origin: *kinein* (Greek) – *to move*
  - The subfield of Mechanics dealing with motions of bodies

- Forward kinematics
  - Given is a set of actuator positions
  - Determine corresponding reference pose

- Inverse kinematics
  - Given is a desired reference pose
  - Determine corresponding actuator positions

Kinematics
Inverse Kinematics for Kin. Chains

- From forward kinematics we know
  $\left[x_g \; y_g \; z_g\right]^T = h(\theta_1, \cdots, \theta_n)$

- *h* is often not easily invertible in closed form

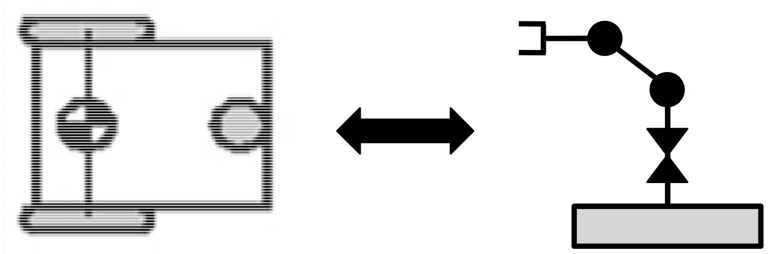
- Approach: *iteratively* perform the following steps
  1. Start from a known forward-kinematic solution (e.g., via sampling) $\left[x_i \; y_i \; z_i\right]^T = h(\theta_1, \cdots, \theta_n)$
  2. Linearize h around $(\theta_1, \cdots, \theta_n)$, resulting in the *Jacobian*

$$J = \left.\begin{bmatrix} \frac{\partial h_1}{\partial \theta_1} & \cdots & \frac{\partial h_1}{\partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial \theta_1} & \cdots & \frac{\partial h_m}{\partial \theta_n} \end{bmatrix}\right|_{\theta=\theta_i}$$

  3. Invert the Jacobian to obtain $\left[\Delta\theta_1 \cdots \Delta\theta_2\right]^T = J^{-1}\left[\Delta x_i \; \Delta y_i \; \Delta z_i\right]^T$
  4. Move by Δ in direction $\left[x_g - x_i \; y_g - y_i \; z_g - z_i\right]^T$

6 Kinematics
(Wheeled) Non-Holonomic Systems

- Wheels
  - Are often subject to motion constraints
  - Often do not allow to compute kinematics directly



- Consequently, for most wheeled robots, actuator positions do not map to unique reference poses
  - There is *no direct* (*i.e.*, instantaneous) *way to measure a robot's position*
  - *Position must be integrated over time*, depends on the path taken

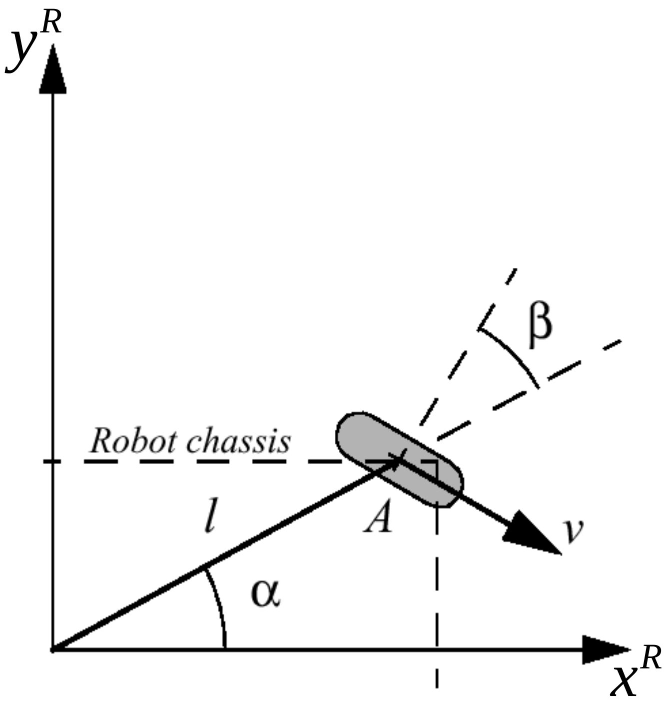
- Understanding mobile robotic motion requires an understanding of wheel constraints placed on the robot's mobility

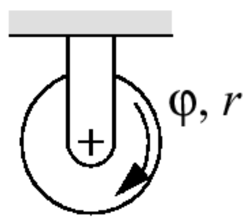
7 Kinematics
Forward Kinematics for Wheels

- Fixed standard wheel

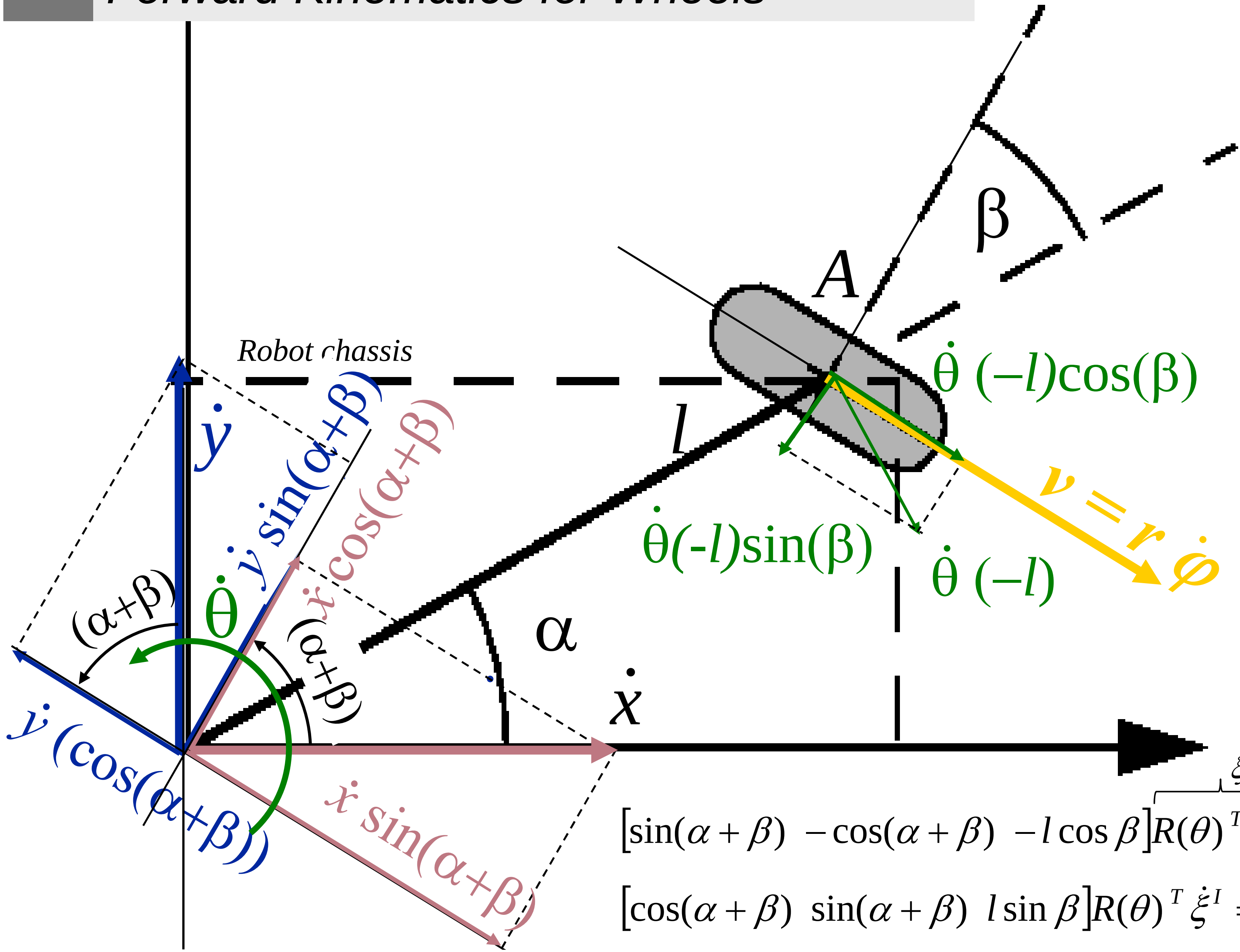
$$\left[\sin(\alpha+\beta)\ \ -\cos(\alpha+\beta)\ \ -l\cos\beta\right]\overbrace{R(\theta)^{T}\dot{\xi}^{I}}^{\dot{\xi}^{R}=\left[\dot{x}^{1}\ \dot{y}^{1}\ \dot{\theta}^{1}\right]^{T}}-r\dot{\varphi}=0$$

$$\left[\cos(\alpha+\beta)\ \ \sin(\alpha+\beta)\ \ l\sin\beta\right]R(\theta)^{T}\dot{\xi}^{I}=0$$





Kinematics
Forward Kinematics for Wheels



$$\left[\sin(\alpha+\beta)\;\; -\cos(\alpha+\beta)\;\; -l\cos\beta\right]\overbrace{R(\theta)^T\,\dot{\xi}^I}^{\dot{\xi}^R=\left[\dot{x}^1\;\dot{y}^1\;\dot{\theta}^1\right]^T} - r\dot{\varphi} = 0$$

$$\left[\cos(\alpha+\beta)\;\; \sin(\alpha+\beta)\;\; l\sin\beta\right]R(\theta)^T\,\dot{\xi}^I = 0$$

Differential Forward Kinematics
Concatenation of Constraints

- Given a wheeled robot
  - Each wheel imposes ≥ 0 constraints on its motion
  - Only fixed and steerable *standard wheels impose no-sliding constraints*

- Suppose the robot has $N_f + N_s$ standard wheels of radius r_i, then the individual wheel constraints can be concatenated in matrix form
  - Rolling constraints

$$J_1(\beta_s)R(\theta)\dot{\xi}^I + J_2\dot{\varphi} = 0 \qquad \underbrace{\varphi(t) = \begin{bmatrix} \varphi_f(t) \\ \varphi_s(t) \end{bmatrix}}_{(N_f+N_s)\times 1} \quad \underbrace{J_1(\beta_s) = \begin{bmatrix} J_{1f} \\ J_{1s}(\beta_s) \end{bmatrix}}_{(N_f+N_s)\times 3} \quad J_2 = diag(r_1 \cdots r_N)$$

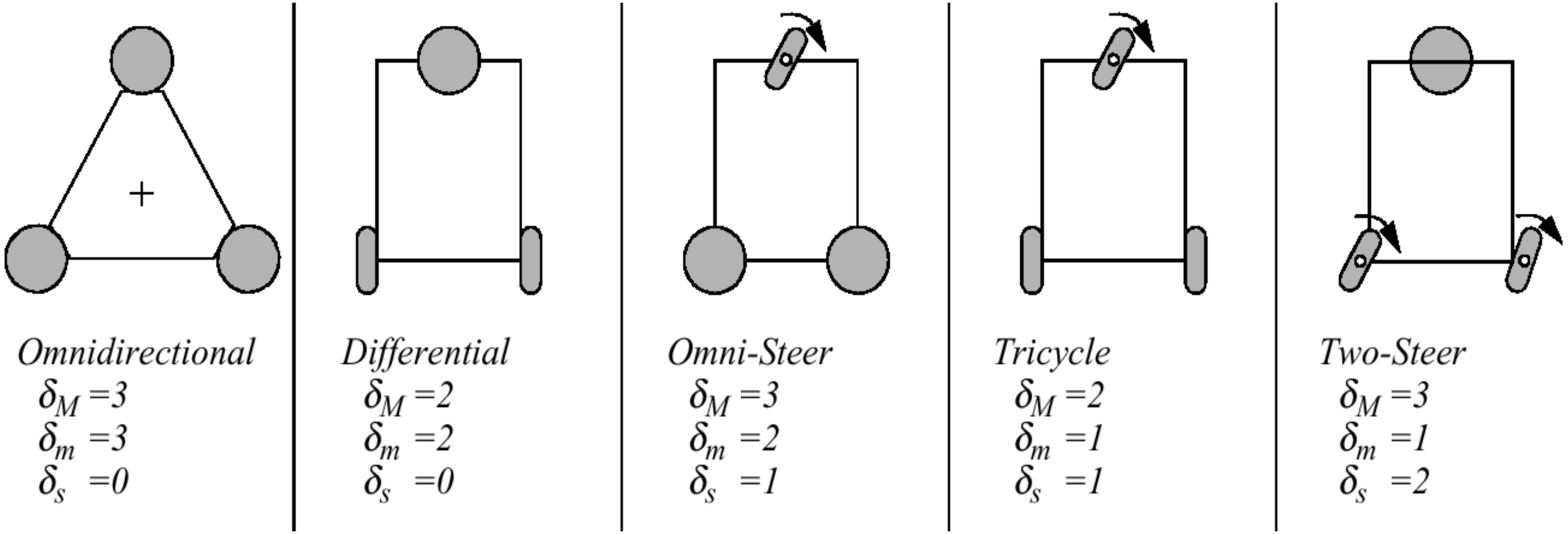
  - No-sliding constraints

$$C_1(\beta_s)R(\theta)\dot{\xi}^I = 0 \qquad \underbrace{C_1(\beta_s) = \begin{bmatrix} C_{1f} \\ C_{1s}(\beta_s) \end{bmatrix}}_{(N_f+N_s)\times 3}$$

- Solving for $\dot{\xi}^I$ results in an expression for *differential forward kinematics*

Five Basic Types of Three-Wheel Configurations

- Degree of mobility δ_m = 3 - *Number of independent wheel constraints*
- Degree of steerability δ_s
- Robots maneuverability $\delta_M = \delta_m + \delta_s$



Roland Siegwart
Margarita Chli
Martin Rufli
Davide Scaramuzza

ETH Master Course: 151-0854-00L

Autonomous Mobile Robots

Perception: Sensors Overview

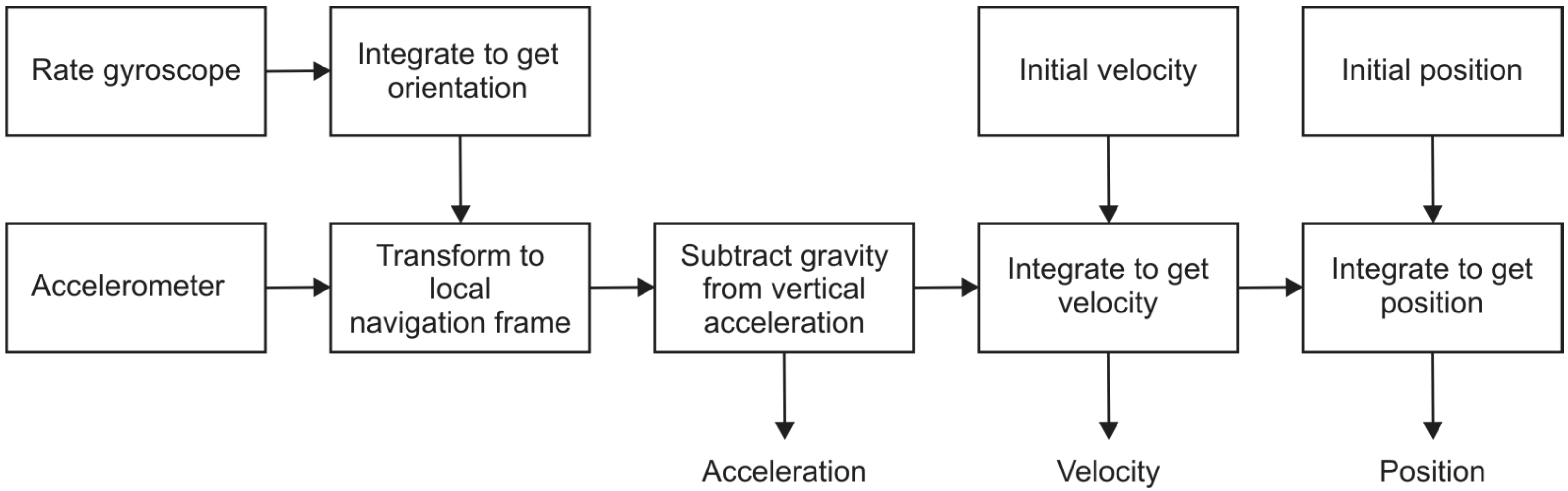
Sensors Overview
Sensor Outline

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

Sensors Overview
Inertial Measurement Unit (IMU)

- **Definition**
  - **An inertial measurement unit (IMU) is a device that uses** measurement systems such as **gyroscopes and accelerometers** to estimate the relative position (x, y, z), orientation (roll, pitch, yaw), velocity, and acceleration of a moving vehicle with respect to an inertial frame

- In order to estimate motion, **the gravity vector must be subtracted**. Furthermore, **initial velocity has to be known** (this is done by starting moving from a rest position)



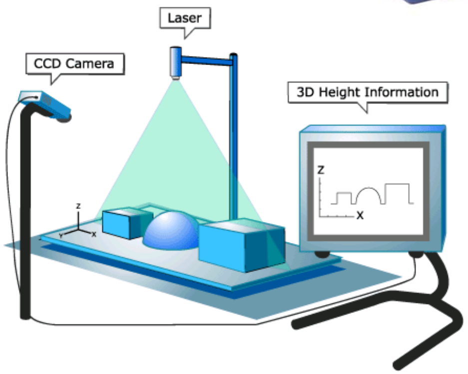
Sensors Overview
Range sensors

- Sonar

- Laser range finder

- Time of Flight Camera

- Structured light



Roland Siegwart

Margarita Chli

Martin Rufli

Davide Scaramuzza

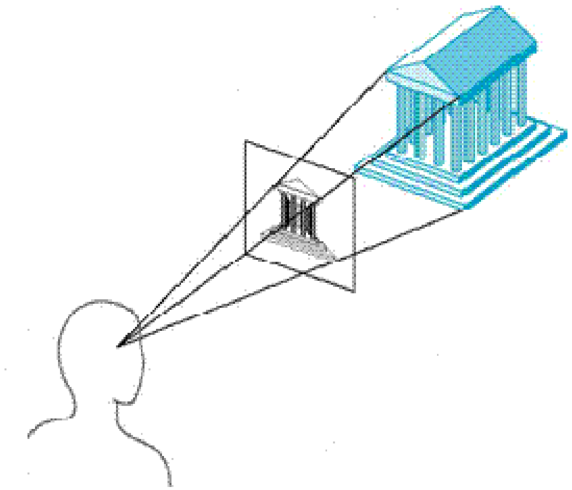
ETH Master Course: 151-0854-00L

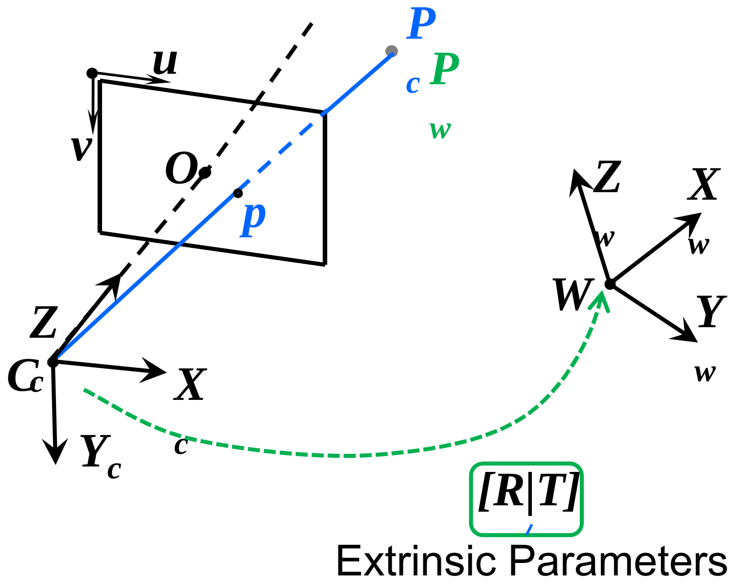
Autonomous Mobile Robots

Robot Vision

Robot Vision
From World to Pixel coordinates

- Mapping of world points to pixels in the image

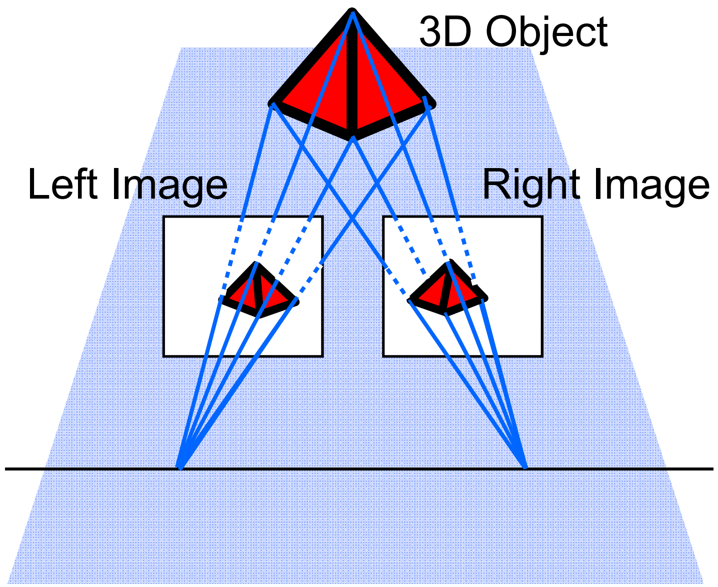




- Perspective projection
  - Convert a 3D point in camera coordinates P_c to pixel coordinates
  - Generalize the projection for any 3D point P_w in world coordinates
  - Use projection equations for *camera calibration*

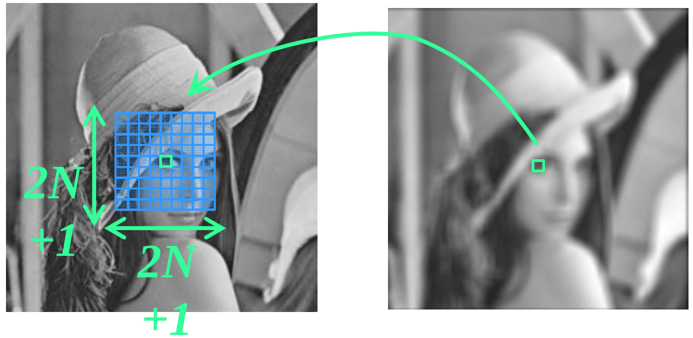
Robot Vision
Stereo Vision - Summary

- Summary
  - Stereo imaging can give us *scale*
  - *"Triangulation"*: with known stereo-camera configuration, we can compute the 3D coordinates of a point seen in both images
  - *Epipolar constraint* for efficient & robustified search for correspondences
  - Use stereo processing for 3D scene reconstruction or computation of disparity maps
  - Generalize stereo processing to multiple cameras for *structure from motion*

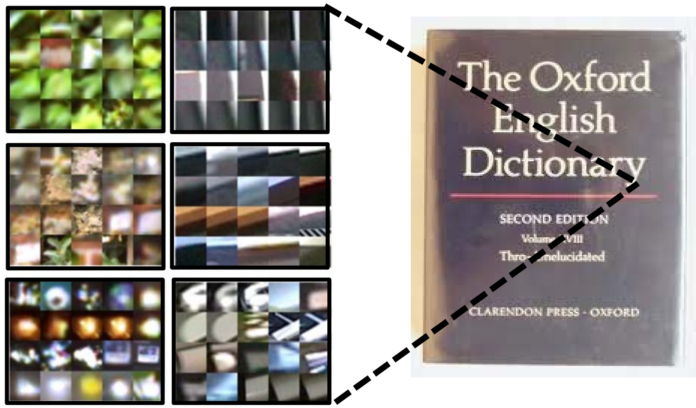


Robot Vision
Salient Image Regions

- Correlation vs. Convolution
  - Use in template matching, smoothing and taking the derivate of an image
- Image filtering for *edge detection*
- Point Features: *Haris*, *Sift*, FAST, BRIEF, BRISK and their characteristics e.g. scale/rotation invariance, computational time
- Building and using the visual vocabulary for place recognition



Examples of Visual Words

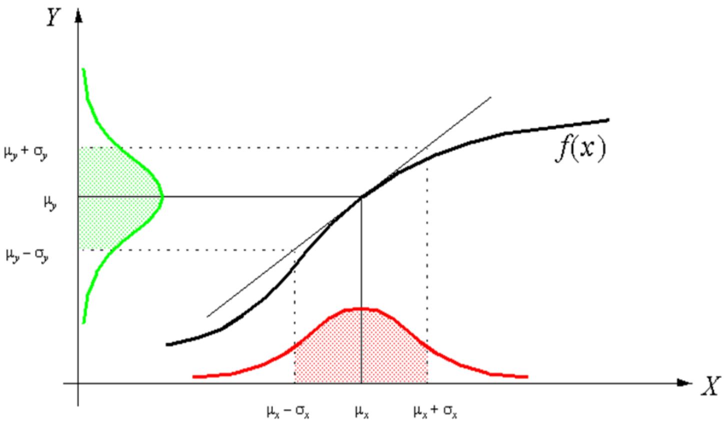


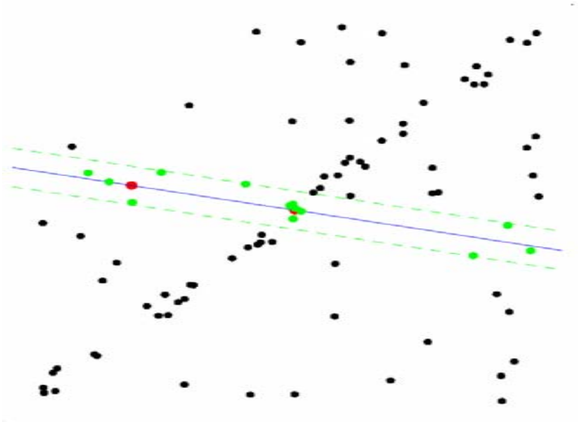
19

Robot Vision
Error Propagation and Line Fitting

- Representing uncertainty for real-world data
- The *error propagation law* and its significance

- Line Fitting algorithms for image/laser point clouds:
  - Split-and-merge, Line regression, RANSAC, Hough Transform, …
  - How they work and their characteristics and applications





Roland Siegwart
Margarita Chli
Martin Rufli
Davide Scaramuzza

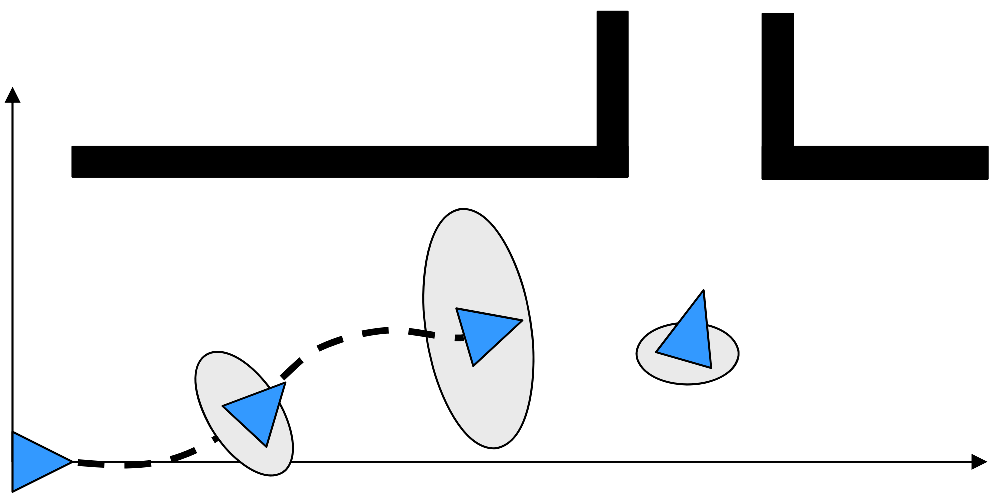
ETH Master Course: 151-0854-00L

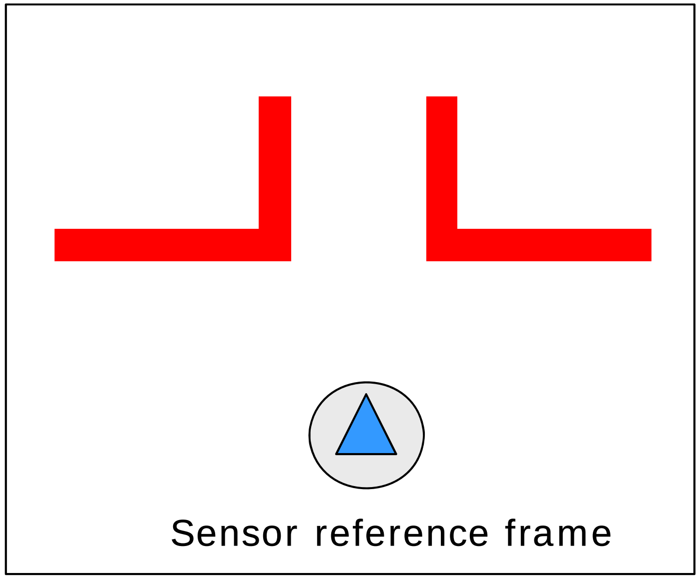
Autonomous Mobile Robots

Localization

Localization
The probabilistic localization problem

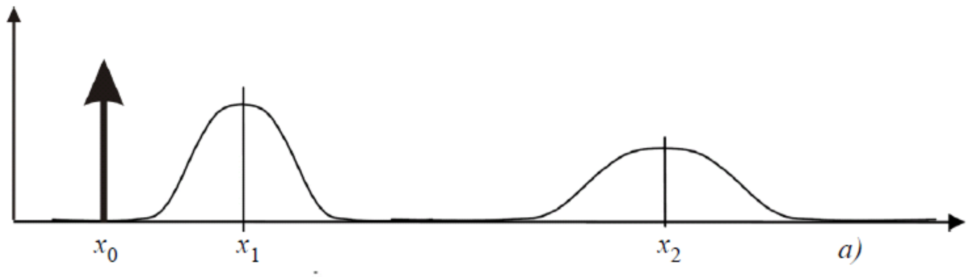
- Consider a mobile robot moving in a known environment.
- As it starts to move, say from a precisely known location, it can keep track of its motion using odometry.
- Due to odometry uncertainty, after some movement the robot will become very uncertain about its position.
- To keep position uncertainty from growing unbounded, the robot must localize itself in relation to its environment map. To localize, the robot uses its on-board exteroceptive sensors (e.g. ultrasonic, laser, vision sensors) to make observations of its environment
- The robot updates its position based on the observation. Its uncertainty shrinks.



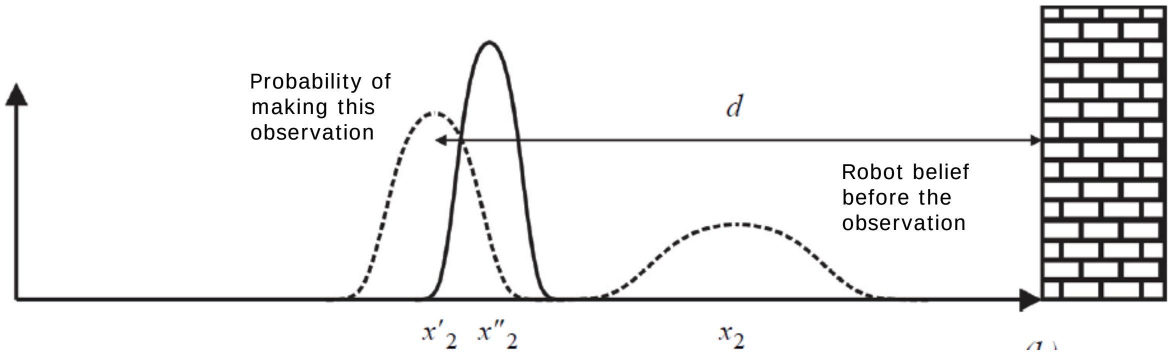


Localization
Action and Perception updates

- In robot localization, we distinguish two update steps:
  1. ACTION (or prediction) update:
     - the robot moves and estimates its position through its **proprioceptive** sensors. During this step, the robot uncertainty grows.



  2. PERCEPTION (or meausurement) update:
     - the robot makes an observation using its **exteroceptive** sensors and correct its position by opportunely combining its belief before the observation with the probability of making exactly that observation.
       During this step, the robot uncertainty shrinks.



Localization
Solution to the probabilistic localization problem

How do we solve the Action and Perception updates?

- Action update uses the Theorem of Total probability

$$\overline{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$$

- Perception update uses the Bayes rule

$$bel(x_t) = \eta \cdot p(z_t \mid x_t) \overline{bel}(x_t)$$

(because of the use of the Bayes rule, probabilistic localization is also called *Bayesian localization*)

Localization
Markov versus Kalman

Two approaches exist to represent the probability distribution and to compute the Total Probability and Bayes Rule during the Action and Perception phases

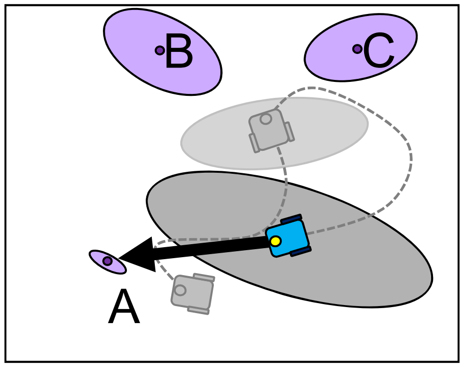
Markov	Kalman
• The configuration space is divided into many cells. The configuration space of a robot moving on a plane is 3D dimensional (x,y,θ). Each cell contains the probability of the robot to be in that cell. • The probability distribution of the sensors model is also discrete. • During Action and Perception, all the cells are updated. Therefore, the computational cost is very high	• The probability distribution of both the robot configuration and the sensor model is assumed to be continuous and **Gaussian!** • Since a Gaussian distribution is only described by its mean value μ and covariance Σ, we need only to update μ and Σ. Therefore the computational cost is very low!

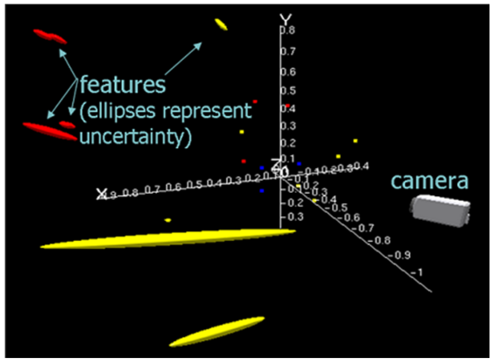
Localization
Markov versus Kalman

Markov	Kalman
PROS ▪ localization starting from any unknown position ▪ recovers from ambiguous situation	**PROS** ▪ Tracks the robot and is inherently very precise and efficient
CONS ▪ However, to update the probability of all positions within the whole state space at any time requires a discrete representation of the space (grid). The required memory and calculation power can thus become very important if a fine grid is used.	**CONS** ▪ If the uncertainty of the robot becomes too large (e.g. collision with an object) the Kalman filter will fail and the position is definitively lost

Simultaneous Localization and Mapping
SLAM - Summary

- What is SLAM and how does it work?
- Graphical representation of SLAM and approaches to solve it
  - *Full graph optimization*
  - *Filtering*
  - *Keyframe-based* approaches
- Popular techniques, working principles and relative merits
  - *EKF SLAM* (via the MonoSLAM system)
  - *Particle Filtering SLAM*
  - *GraphSLAM*





Roland Siegwart

Margarita Chli

Martin Rufli

Davide Scaramuzza

ETH Master Course: 151-0854-00L

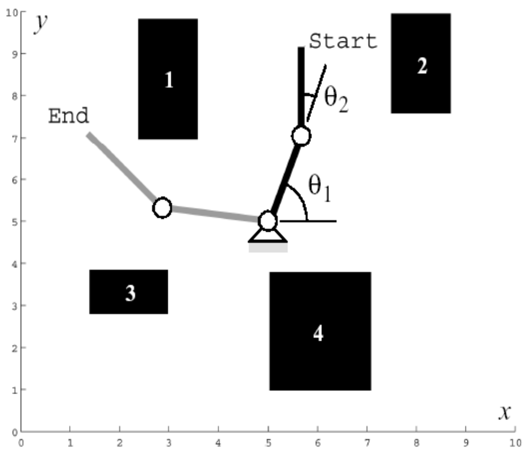
Autonomous Mobile Robots

Motion Planning

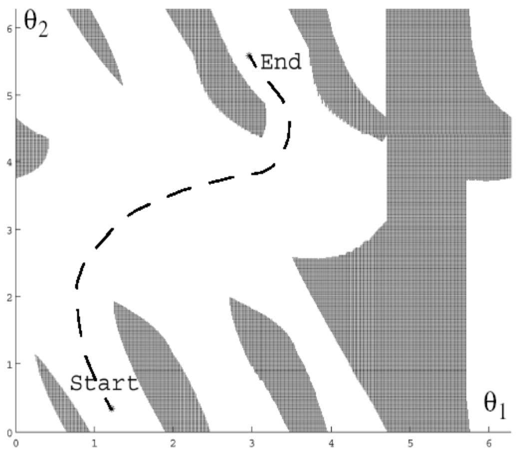
Motion Planning
Work & Configuration Spaces

- Work space
  - The *physical space* we live and operate in
  - Usually 3D on flat ground and 6D for flying robots
- State space (Configuration space)
  - The *space of configurations* an agent operates in
  - One DoF for each non-redundant actuator

- Application: manipulator arm



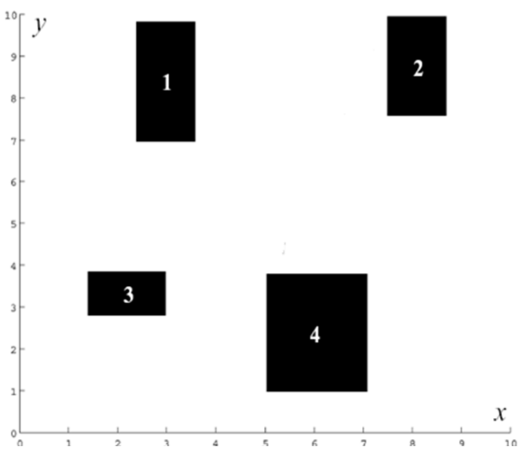
Work Space



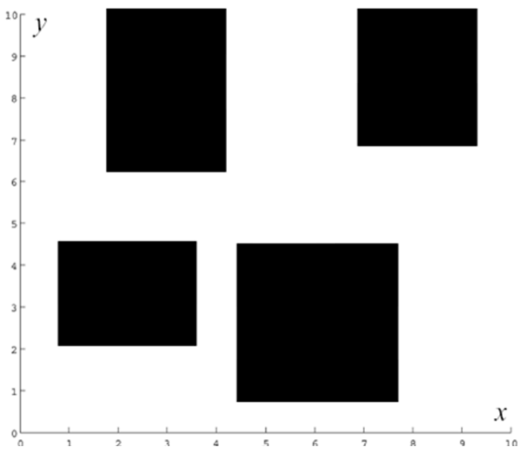
Configuration Space

Motion Planning
Work & Configuration Spaces

- Application: wheeled robot
  - Operation on ground results in a 3D config space: (x, y, θ)
  - In practice / for simplicity often a *holonomic point-mass* is assumed
  - Obstacles are *inflated* by the robot radius



Work Space

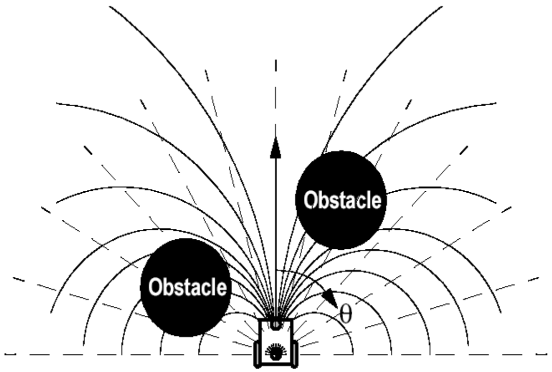


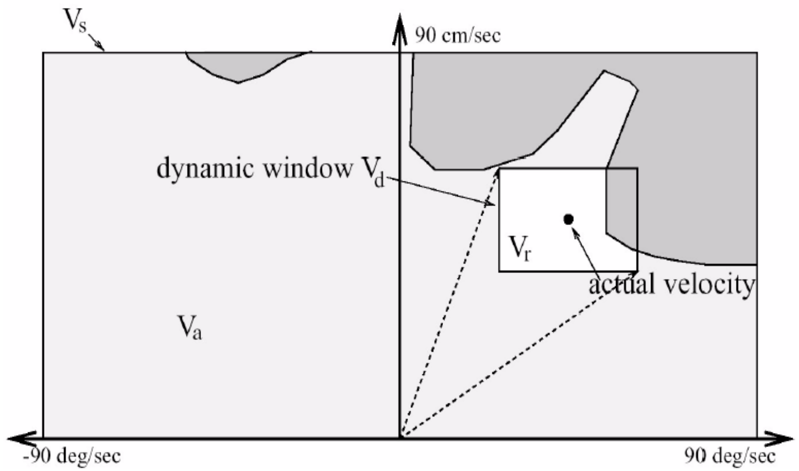
Configuration Space

Motion Planning
Dynamic Window Approach (DWA)

- Working Principle
  - 2D evidence grid transformed into (v,ω) space based on inevitable collision states
  - Circular arcs and acceleration window V_d *account for vehicle kinematics*
  - Selection of (v,ω)-pair within $V_r = V_s \cap V_a \cap V_d$ that maximizes objective function $G(v,\omega) = \sigma\left(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot velocity(v,\omega)\right)$

- Properties
  - Prone to local minima





ETH Zürich

31

Motion Planning
Graph Search Algorithms

- Overview
  - Graph structure is a *discrete representation G(N,E)*
  - Solves a least cost problem between two states on a (directed) graph

- Limitations
  - State space is discretized, hence completeness is at stake
  - *Feasibility of paths* is often *not inherently encoded*

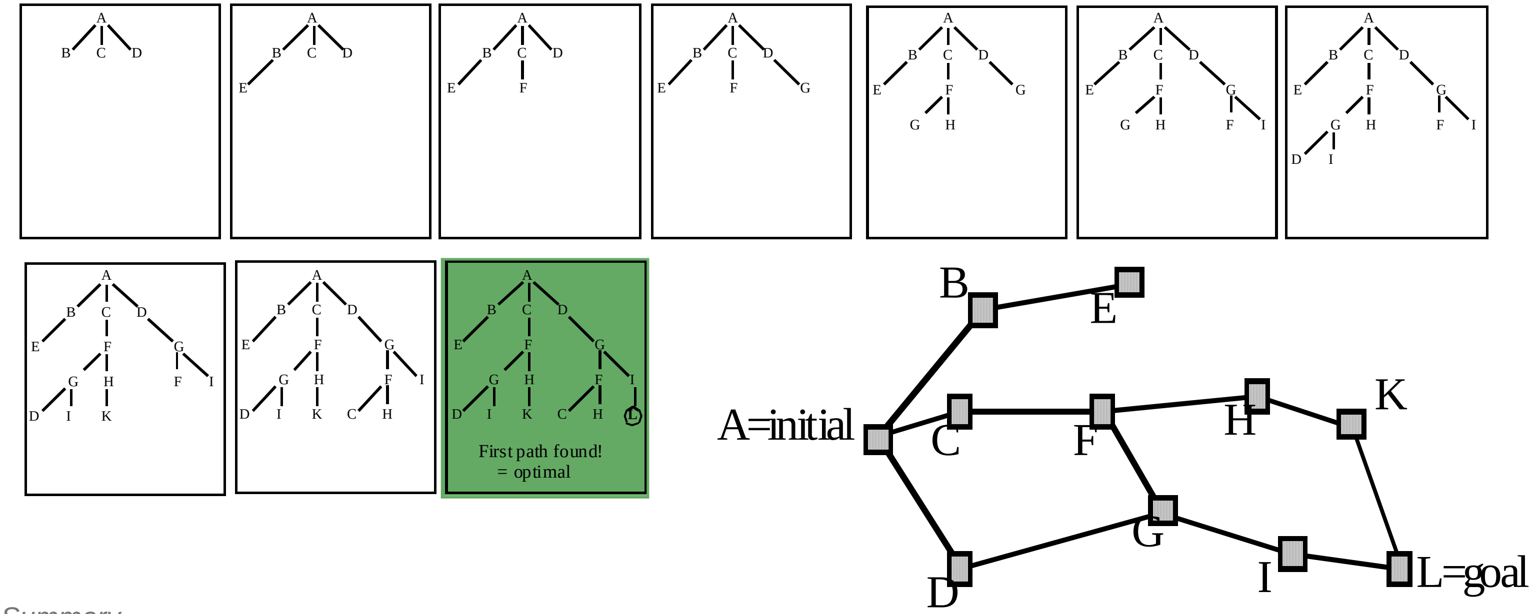
- Algorithms Covered
  - Breadth first
  - Depth first
  - Dijkstra
  - A* and variants
  - (D* and variants)

Courtesy wikipedia.org

ETH Zürich

Global Planning
Breadth First Graph Search

- Working principle
  - Operates on a *FIFO queue* and a *“closed” list*
  - Backtracks optimal solution starting from goal state
- Properties
  - First goal state encountered is optimal, iff edge costs are identical
  - Optimal for arbitrary edge costs, iff expansion is continued until FIFO queue is empty



Discussion:

Exam

- Oral, 30 minutes

- 3 Questions selected drawn by dices
  - Application
  - Basics 1
  - Basics 2

- Questions given beforehand (Webpage, sent to all participants)
  - http://www.asl.ethz.ch/education/master/mobile_robotics/Questions2013.pdf

- Example:

All terrain demining in unstructured environments		3.2.3 Wheel kinematic constraints of the 5 wheel types, pro/cons of wheel types		5.2.4 Odometric position estimation and error model for a differential drive robot and their use in Markov and EKF localization