Outline

1. Introduction
2. History
3. Design of Systems
4. Kinematics
5. Sensing
6. Features
7. Mapping and Localisation
8. Planning and Navigation
9. Lab System
10. Summary
Objective:
- Introduce the overall parts of a mobile robotic systems
- Basic notation and mathematics
- Have a sense of current practice
- Enough basic knowledge to play with simple design

Disclaimer:
- Normally my introduction is 600+ slides, so this is a condensed version for 3+ hours!
- At places the material is sketchy
Outline

1. Introduction
2. History
3. Design of Systems
4. Kinematics
5. Sensing
6. Features
7. Mapping and Localisation
8. Planning and Navigation
9. Lab System
10. Summary
Robotics dates back to Greek mythology - Pygmalion etc., da Vinci had drawings of humanoids in his notes. The work robota means labour in Czech, and is used as inspiration to Capek in his 1920 RUR Play (Rossum’s Universal Robots), the first real literature description of robotics.

Walters (Essex) designed one of the first “autonomous” robots around 1948.

Robotics was introduced and made popular by Issac Asimov.

Industrial robotics emerged around 1956 in a patent application.

Industrial Robotics was initially by Unimation (Joseph Engelberger)
History

- Industrial robotics has sold about 900,000 units with another 80,000 being installed each year.
- The growth rate is around 8% per year.
- The main business sectors are car manufacturing (welding and painting) and packaging.
- The new emerging sector is the food industry. The drive is to avoid human contact with raw food items.
The service robot industry is gradually becoming a major sector.

The growth rate is 400% per year.

The main applications have been vacuum cleaners and lawn movers.

Toys such as the Sony AIBO is another growing application area.

The professional markets is expected to be a new sector.
Service Motivation

- The number of elderly people will grow by 50% over the next 20 years
- The dependency ratio will change from 3 to 2 which will challenge the economy.
- The “AgeQuake” provides a view of how it will change our society
- This is the drive behind Japanese robotics and more recently Korean efforts (100M Euro/year).
Manufacturing Robots
Domestic Robots

Robots for Quality of Life

- Tennisball-collector (D)
- Auto Mower, Husqvarna (S)
- Trilobite® 2.0, Electrolux (S)
- Assisten Roboter Manipulator "ARM", Exact Dynamics, www.exactdynamics.nl (NL)
- Pool Cleaner, Weda (S)
- Window cleaners Quirl, IPA (D)
Entertainment

Entertainment-Robots

SONY QRIO (J)
Sanyo Fiatthru (J) auf der ROBODEX, 2003
Sony AIBO, ERS-7M2 (J)
Partner-type Personal Robot (PaPeRo), NEC System Technologies, Ltd. (J)

TOYOTA MOTOR CORPORATION (TMC), (J)
Mona, Oskar, Opel Rüsselsheim (D)
“Banru”, tmsuk Co., LTD. and Sanyo Electric (J)
Wakamaru, MHI (J)
RoboX, Blueotics S.A. (CH)
3. Design of Systems

4. Kinematics

5. Sensing

6. Features

7. Mapping and Localisation

8. Planning and Navigation

9. Lab System

10. Summary
Design of Systems - Outline

- Localisation & Map Building
  - Position/Map
  - Local map
- Perception
- Task/Mission
  - "Cognition" & Path Planning
  - Path
- Motion Control
- Environment

Overview:
- Introduction
- History
- Design of Systems
- Kinematics
- Sensing
- Features
- Mapping and Localisation
- Planning and Navigation
- Lab System
- Summary
Architectural Approaches

- Early research on robotics was almost based on a sense-plan-act model
- By \( \approx 1985 \) Brooks proposed a behaviour based / subsumption based approach to robotics. A radical change in partition of system design
- Subsumption also has its limitations
- The hybrid deliberative approach took over by \( \approx 1990. \)
Temporal decomposition of architecture

- Division according to temporal requirements
- Provides a coarse division of control
- Layering can be loosely synchronous
Traditionally the sense-plan-act has been a well known method.

More recent systems have used a parallel decomposition often referred to as a reactive/behaviour based system.

Behaviours are situated control modules with well defined context and control specifications.
The Hybrid Architecture Interleaves Deliberation (planning) with reactive control to have flexible handling of mission goals and unexpected events – obstacles. By far the most common architecture today.
Important to recognize that the mobile architectures are entirely devoted to navigation

For cognitive systems this is only part of the puzzle!

- The exact integration is subject to research and part of the CoSy project (WP1)
- Aaron Sloman is happy to provide an elaborate lecture on the cognitive side of the architecture discussion.
Pieces of the puzzle

- Modelling the interaction with the environment
  - From sensing and control to motion in the world
- Acquiring sensory information about the world (for navigation primarily mapping (local/global))
- Design of control strategies to achieve task completion.
- Integration into an overall structure
Outline

1. Introduction
2. History
3. Design of Systems
4. Kinematics
5. Sensing
6. Features
7. Mapping and Localisation
8. Planning and Navigation
9. Lab System
10. Summary
Kinematics – Models of the Physical System

- We will only consider wheeled systems
- Locomotion could also be based on legs, tracks, etc. beyond the present lecture.
- In addition we will only consider simple wheeled systems
Systems with wheels

- Wheels is often a good solution – in particular indoor
- Three wheels enough to guarantee stability
- More than three wheels requires suspension
- Wheel configuration and type depends upon the application
Types of wheels

- There are four types of wheels
- Standard wheel: two degrees of freedom – rotation around motorized axle and the contact point
- Castor wheel: three degrees of freedom: wheel axle, contact point and castor axle
Types of wheels – II

- **Swedish wheel**: three degrees of freedom - motorized wheel axles, rollers, and contact point (Video)

- **Ball or spherical wheel**: suspension not yet technically solved
Synchro Drive

- All wheels are driven synchronously by one motor
  - Defines speed
- All wheels are steered synchronously by second motor
  - Define direction of motion
- orientation of inertial frame remains the same
Differential drive setup

- Two wheeled or possible two wheels and a castor
- Control of each wheel independently
Bicycle drive

- Two wheeled with one wheel control of direction
- Only dynamically stable
Motion Control

Requirements

- Kinematic / dynamic model of the robot
- Model of ground/wheel interaction
- Definition of required motion → velocity / position control
- Design of control law to satisfy constraints
Mobile Robot Kinematics

- Model of mechanical behaviour of robot for design and control
- Models can be used both for mobile systems and manipulators
- Manipulators allow “direct” estimation of position, which is not always true for mobile systems
- Position to be derived from integration over time
- Motion is not free. There are constraints to be considered in the design and control generation.
Reference frames

- Inertial reference frame (I)
- Robot references frame (R)
- Robot pose

\[ \xi_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \]
The relation between the references frame is through the standard orthogonal rotation transformation:

\[
R(\theta) = \begin{bmatrix}
\cos(\theta) & \sin(\theta) & 0 \\
-\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]

Enable handling of motion between frames

\[
\dot{\xi}_R = R(\theta)\dot{\xi}_I
\]
Simple 90° rotation

- Now with $\dot{\xi}_R = R(\theta)\dot{\xi}_I$
- $\dot{\xi}_R = R(\frac{\pi}{2})\dot{\xi}_I$

$$R\left(\frac{\pi}{2}\right) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Implies

$$\dot{\xi}_R = R\left(\frac{\pi}{2}\right)\dot{\xi}_I = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{y} \\ -\dot{x} \\ \dot{\theta} \end{bmatrix}$$
Goal:

- Determine the robot speed $\dot{\xi} = [\dot{x} \ \dot{y} \ \dot{\theta}]^T$ as a function of wheel speed $\phi$, steering angle $\beta$, steering speed $\dot{\beta}$ and the geometric parameters of the robot.

Forward kinematics

$$\dot{\xi} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(\phi_1, \ldots, \phi_n, \beta_1, \ldots, \beta_m, \dot{\beta}_1, \ldots, \dot{\beta}_m)$$

Inverse kinematics

$$\begin{bmatrix} \phi_1 & \ldots & \phi_n & \beta_1 & \ldots & \beta_m & \dot{\beta}_1 & \ldots & \dot{\beta}_m \end{bmatrix}^T = f(\dot{x}, \dot{y}, \dot{\theta})$$

Why not

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = f(\phi_1, \ldots, \phi_n, \beta_1, \ldots, \beta_m)$$

the relation is not straight forward.
Forward kinematic model – differential drive

- Assume a set up with two drive wheels. Wheels have radius \( r \), and are placed at a distance \( l \) from the center.
- Wheels rotate at speeds \( \dot{\varphi}_1 \) and \( \dot{\varphi}_2 \).
- Prediction of the motion of the robot motion in the global frame

\[
\dot{\xi}_I = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f(l, r, \theta, \dot{\varphi}_1, \dot{\varphi}_2)
\]
Differential drive model

- Given $\dot{\xi}_l = R(\theta)^{-1} \dot{\xi}_R$
- Speed of each wheel is $r \dot{\varphi}_i$, the translational speed is the average velocity
  \[ \dot{x}_R = r \frac{\dot{\varphi}_1 + \dot{\varphi}_2}{2} \]
- The instantaneous rotation of $P$ for one wheel is
  \[ \omega_1 = \frac{r \dot{\varphi}_1}{2l} \]
- The total rotation is then
  \[ \dot{\theta} = \frac{r}{2l} (\dot{\varphi}_1 - \dot{\varphi}_2) \]
Differential drive model

- Given $\dot{\xi}_I = R(\theta)^{-1}\dot{\xi}_R$
- The full model is then:

\[
\dot{\xi}_I = R(\theta)^{-1} \frac{r}{2} \begin{bmatrix}
\dot{\varphi}_1 + \dot{\varphi}_2 \\
0 \\
\frac{\dot{\varphi}_1 - \dot{\varphi}_2}{l}
\end{bmatrix}
\]

- The rotation matrix is trivial to invert

\[
R(\theta)^{-1} = \begin{bmatrix}
\cos(\theta) & -\sin(\theta) & 0 \\
\sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 1
\end{bmatrix}
\]
Kinematic control is following of a pre-specified trajectory described in terms of positions and velocities.

Often the trajectory is divided into trajectory segments.

Simple controllers use a combination of arcs and line segments (as done on American roads). Others use clothoids in which the curvature changes linearly with time, as done on European roads.

An entire field of robotics is devoted to path planning. See http://msl.cs.uiuc.edu/planning for comprehensive / free book on the topic.
A more appropriate strategy is a trajectory feedback controller that uses the path specification as “control” points to drive the robot system.
Problem statement

- In the robot reference frame the error is
  \[ e = [x, y, \theta]^T_R \]

- The task is now to design a control matrix \( K \)
  \[
  K = \begin{bmatrix}
  k_{11} & k_{12} & k_{13} \\
  k_{21} & k_{22} & k_{23}
  \end{bmatrix}
  \quad k_{ij} = k(t, e)
  \]

- Such that
  \[
  \begin{bmatrix}
  v(t) \\
  \omega(t)
  \end{bmatrix} = Ke = K \begin{bmatrix}
  x \\
  y \\
  \theta
  \end{bmatrix}_R
  \]

  drives the error to zero
  \[ \lim_{t \to \infty} e(t) = 0 \]
The basic setup for control
Kinematic model

Consider a differential drive robot in the inertial frame

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix} =
\begin{bmatrix}
\cos(\theta) & 0 \\
\sin(\theta) & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
v \\
\omega
\end{bmatrix}
\]

In polar coordinates the error is now

\[
\rho = \sqrt{\Delta x^2 + \Delta y^2} \\
\alpha = -\theta + \text{atan2}(\Delta y, \Delta x) \\
\beta = -\theta - \alpha
\]
Kinematic control

Rephrased in polar coordinates:

\[
\begin{bmatrix}
\dot{\rho} \\
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix} =
\begin{bmatrix}
\cos(\alpha) & 0 \\
\frac{\rho}{\sin(\alpha)} & -1 \\
\frac{\rho}{\sin(\alpha)} & 0
\end{bmatrix}
\begin{bmatrix}
v \\
\omega
\end{bmatrix}
\]
The control law

- Control the linear control law:
  \[ v = k_\rho \rho \]
  \[ \omega = k_\alpha \alpha + k_\beta \beta \]

- Which generates a closed loop system of:
\[
\begin{bmatrix}
\dot{\rho} \\
\dot{\alpha} \\
\dot{\beta}
\end{bmatrix}
= \begin{bmatrix}
-k_\rho \rho \cos(\alpha) \\
-k_\rho \rho \sin(\alpha) - k_\alpha \alpha - k_\beta \beta \\
-k_\rho \sin(\alpha)
\end{bmatrix}
\]
Stability requirement

- It can be shown that the system is exponentially stable if:

\[
\begin{align*}
  k_\rho &> 0 \\
  k_\beta &< 0 \\
  k_\alpha - k_\rho &> 0
\end{align*}
\]

- Sketch of proof \((\cos x = 1, \sin x = x)\):

\[
\begin{bmatrix}
  \dot{\rho} \\
  \dot{\alpha} \\
  \dot{\beta}
\end{bmatrix} =
\begin{bmatrix}
  -k_\rho & 0 & 0 \\
  0 & -(k_\alpha - k_\rho) & k_\beta \\
  0 & -k_\rho & 0
\end{bmatrix}
\begin{bmatrix}
  \rho \\
  \alpha \\
  \beta
\end{bmatrix}
\]
Sketch of stability requirement

- if $A$ has all eigenvalues where the real part is negative it is exponentially stable
- Characteristic polynomial:

$$\left(\lambda + k_{\rho}\right)\left(\lambda^2 + \lambda\left(k_{\alpha} - k_{\rho}\right) - k_{\rho}k_{\beta}\right)$$
Control examples
Outline

1. Introduction
2. History
3. Design of Systems
4. Kinematics
5. Sensing
6. Features
7. Mapping and Localisation
8. Planning and Navigation
9. Lab System
10. Summary
Sensing
Robots and Sensors

- iRobot ATRV
- Trimble 212DL DGPS
- 3Com WLAN 802.11b
- SICK LMS 291 laser scanner
- AXIS WebCam or SONY XC777 Stereo cameras
- Polaroid 6500 UltraSound
- UltraSound
- Bumpers
- KVH-100 Compass
- Crossbow DMU-6x (INS)
Sensors

- Uncertainty in the layout of the environment due to lack of models or unknown dynamics
- Execution of commands is uncertain due to imperfect actuation
- Sensors are needed to cope with the uncertainty and provide an estimate of “robot state” and environmental layout.
Sensor classes

- Sensing is divided according to the purpose:
  - **Proprioception** Estimation of the internal state of the robot. Configuration, temperature, current, speed of axis, ...
  - **Exteroception** Estimation of the state of the environment with respect to robot

- Sensors are also divided according to measurement principle
  - **Passive** Uses ambient energy to perform the measurement
  - **Active** Transmits energy into environment to allow measurements
## Sensors in mobile robotics

<table>
<thead>
<tr>
<th>Classification</th>
<th>Sensor Type</th>
<th>PC/EC</th>
<th>A/P</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tactile sensors</strong></td>
<td>Switches/Bumpers</td>
<td>EC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Optical barriers</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Proximity</td>
<td>EC</td>
<td>P</td>
</tr>
<tr>
<td><strong>Haptic sensors</strong></td>
<td>Contact arrays</td>
<td>EC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Force/Torque</td>
<td>EC/PC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Resistive</td>
<td>EC</td>
<td>P</td>
</tr>
<tr>
<td><strong>Motor/Axis sensors</strong></td>
<td>Brush Encoders</td>
<td>PC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Potentiometers</td>
<td>PC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Resolvers</td>
<td>PC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Optical encoders</td>
<td>PC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Magnetic encoders</td>
<td>PC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Inductive encoders</td>
<td>PC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Capacity encoders</td>
<td>EC</td>
<td>A</td>
</tr>
</tbody>
</table>
## Sensors for mobile robots

<table>
<thead>
<tr>
<th>Classification</th>
<th>Sensor Type</th>
<th>PC/EC</th>
<th>A/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heading sensors</td>
<td>Compass</td>
<td>EC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Gyroscopes</td>
<td>PC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Inclinometers</td>
<td>EC</td>
<td>A/P</td>
</tr>
<tr>
<td>Beacon based</td>
<td>GPS</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td>(Postion wrt an inertial frame)</td>
<td>Active Optical</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>RF beacons</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Ultrasound beacon</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Reflective beacons</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td>Ranging</td>
<td>Capacitive sensor</td>
<td>EC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Magnetic sensors</td>
<td>EC</td>
<td>P/A</td>
</tr>
<tr>
<td></td>
<td>Camera</td>
<td>EC</td>
<td>P/A</td>
</tr>
<tr>
<td></td>
<td>Ultra-sound</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Laser range</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Structures light</td>
<td>EC</td>
<td>A</td>
</tr>
</tbody>
</table>
## Sensors for mobile robots

<table>
<thead>
<tr>
<th>Classification</th>
<th>Sensor Type</th>
<th>PC/EC</th>
<th>A/P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed/motion</td>
<td>Doppler radar</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Doppler sound</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Camera</td>
<td>EC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>Accelerometer</td>
<td>EC</td>
<td>P</td>
</tr>
<tr>
<td>Identification</td>
<td>Camera</td>
<td>EC</td>
<td>P</td>
</tr>
<tr>
<td></td>
<td>RFID</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Laser ranging</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Radar</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Ultra-sound</td>
<td>EC</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Sound</td>
<td>EC</td>
<td>P</td>
</tr>
</tbody>
</table>
Orientation and heading

- Compass used as a reference since 2000 B.C.
- Today available in solid state technology
- Sensitive to ferro magnetic materials
- High environmental variation
Gyro/Accelerometers

- The inertia of a spinning wheel provides a reference for orientation.
- Today available in fiber-optic and solid state versions
- Double integration implies high noise sensitivity
Ranging for estimation of position is a very common methodology. Several methods are used:

- Time of Flight: Travel time for a pulse
- Phase Difference: Phase of modulation $\propto$ time of travel
- Triangulation: Simple geometric relations

From range measurements position can be estimated.
Time of Flight – Ranging

- Measures travel time.
- Speed of propagation $c$, distance $d$ implies
  \[ d = c \times t \quad \Rightarrow \quad t = \frac{d}{c} \]
- Travels back and forth so $d = \frac{c}{2}t$
Phase differencing – Ranging

- At a distance $d$ the phase difference is

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

- Estimated using PLL which is “inexpensive” technology
Triangulation – Ranging

- Use simple geometric relations to recover depth
- Example is IR/Laser triangulation

\[ D = f \frac{L}{x} \]

- Depth inversely proportional to \( x \)
One of the most common approaches in robotics
Consider handling of two range readings
Given two range pings $d_1$, $d_2$ and known positions: $(0, 0)$ and $(x_2, y_2)$ the position of intersection is

$$x = \frac{x_2^2 + d_1^2 - d_2^2}{2x_2}$$

$$y^2 = \frac{2x_2^2 d_1^2 + 2d_1^2 d_2^2 + 2x_2^2 d_2^2 - x_2^4 - d_1^4 - d_2^4}{4x_2^2}$$

Trivial to compute intersection point(s)
Unique position estimates

- With 3+ range estimates the intersection point is unique
- Noise might contaminate the measurements
Ultra-sonic ranging

- Widely used in underwater for mapping
- In air the main application has been cameras
- Speed of sound $\approx 343 \frac{m}{s}$ so processing is "slow"
- Pulse based time of flight (49.1 kHz)
- Cheap technology for mass products
Sonar ranging
Sonar sensing – Characteristics
IR Ranging

- Short-range triangulation sensor
- Dependent on surface color/reflectance
- Very inexpensive (easy interfacing)
- Primarily obstacle detection
Laser Scanning - Beacon Based

- Angular distribution / Known landmarks
- Robust to variations
- Easy to install
- Used in factory settings for AGV systems
- More than 15000 units sold
Laser Scanning - TOF

- Rotating mirror
- Pulsed laser
- Range 10-50 m
- Resolution $\approx 1$ cm
- Sampling rate: 37 Hz
Laser scanner - SICK

- Frequent sensor system today (2004)
- Safety classified
- Unusual error distribution (uniform)
- Price: $ 5000
Outline

1 Introduction
2 History
3 Design of Systems
4 Kinematics
5 Sensing
6 Features
7 Mapping and Localisation
8 Planning and Navigation
9 Lab System
10 Summary
Features

- Features introduced to capture primary sensory information
- Two purposes:
  - Compression of data to remove redundant information
  - (Geometric) Invariance to simplify matching/recognition
- Simplified uncertainty model for processing
Feature representations

- Gridmaps: a tessellation of the world
- Scans: (world) aligned sensory data
- Feature maps: discrete collections of features
  - Points, lines, planes, ...
- Topological: graph of places/transitions
Gridmaps

- Easy to understand
- $O(env^2)$ in size
- Easy to update (might be slow)
Bayes Grid

- Assume \( p_{ij}(t = 0) = 0.5 \)
- Update:

\[
p_{ij}(t+1) = \frac{P(R|S_{ij} = O)p_{ij}(t)}{P(R|S_{ij} = O)p_{ij}(t) + [1 - P(R|S_{ij} = O)][1 - p_{ij}(t)]}
\]

where \( R \) = reading and \( S \) is the sensor reading. \( P(R|S) \) is the sensor model.
Scans

- Direct alignment of sensor data
- Easy to model

- Generalizes poorly
- One sensor system
- ScanStudio by Gutmann
Feature Maps

- Discrete feature map
- Easy for multi feature intg
- Easy to handle
- $O(\text{features})$
Topological Graphs

- Graph representation
- Place recognition
- $O(\text{places})$
- Coarse localisation
- A good planning rep.
Feature Detection

- Point Detection / Estimation
- Line Estimation
- Curvature points
- Plane Estimation
Point Estimation

- Many sensor generate range estimates (Sonar, GPS, Laser)
- Triangulation is a well known technique for estimation of points
- Fusion of multiple range readings into an estimate
- Theory is well known from phased array radar
Line estimation

- Lines are a predominant feature in engineered environments
- There is an abundance of methods for line estimation
- LSQ, Split-Merge, Hough, EM-estimation,
Voting Estimators

- Voting provides a simple estimator for detection
- Voting requires:
  1. A Voting Space
  2. A voting function (structure function)
  3. A decision function (often local extrema)
- Hough is one of the most widely used. Can also be used for lines and other shapes
The Hough transform

- Line model:
  \[ \rho = x \cos(\theta) + y \sin(\theta) \]
- Voting space: \([\theta, \rho]\)
- Voter: traverse \(\theta\) space
- Local maximum w. NMS for all points in \((x, y)\)
  - for each \(\theta : 0 \rightarrow \pi\)
    - calc \(\rho\) and increment \((\theta, \rho)\)
- Generates infinite lines.
Hough example
Scanning is in polar coordinates.
The density of points is varying.
Close structure will accumulate more points.
Range weighting can compensate. Weight by $\frac{1}{\cos(\psi_i - \theta)}$. 
Least square minimization:

- Line equation: \( y = ax + b \)
- Error in fit: \( \sum_i (y_i - ax_i - b)^2 \)
- Solution:
  \[
  \begin{pmatrix}
  \bar{y}^2 \\
  \bar{y}
  \end{pmatrix} = \begin{pmatrix}
  \bar{x}^2 & \bar{x} \\
  \bar{x} & 1
  \end{pmatrix} \begin{pmatrix}
  a \\
  b
  \end{pmatrix}
  \]

- Minimizes vertical errors. Non-robust!
LSQ on lasers

- Line model: \( r_i \cos(\phi_i - \theta) = \rho \)
- Error model: \( d_i = r_i \cos(\phi_i - \theta) - \rho \)
- Optimize: \( \text{argmin}_{(\rho, \theta)} \sum_i (r_i \cos(\phi_i - \theta) - \rho)^2 \)
- Error model derived in (Deriche92a)
- Well suited for “clean-up” of Hough lines
Line Estimation

- Total least squares (TLS):
  - Line equation: $ax + by + c = 0$
  - Error in fit: $\sum_i (ax_i + by_i + c)^2$ where $a^2 + b^2 = 1$.
  - Solution:
    $$\begin{pmatrix} \bar{x}^2 - \bar{x}\bar{x} & \bar{x}\bar{y} - \bar{x}\bar{y} \\ \bar{x}\bar{y} - \bar{x}\bar{y} & \bar{y}^2 - \bar{y}\bar{y} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \mu \begin{pmatrix} a \\ b \end{pmatrix}$$

  where $\mu$ is a scale factor.

- $c = -a\bar{x} - b\bar{y}$
The line representation is crucial

Often a redundant model is adopted

Line parameters vs end-points

Important for fusion of segments.

End-points are less stable
1. Introduction
2. History
3. Design of Systems
4. Kinematics
5. Sensing
6. Features
7. Mapping and Localisation
8. Planning and Navigation
9. Lab System
10. Summary
Mapping/Localisation

- A fundamental part of mobile robotics
- Two problems:
  1. Pose initialisation / Kidnapped Robot
     - Initialising the system / recovery
  2. Pose Maintenance
     - Updating pose as the robot moves about
- SLAM – Simultaneous Localisation and Mapping is the ultimate solution.
Pieces for the puzzle

- A model for the robot
- Model of the environment
  - Representation of environment
- A method for feature extraction
- A strategy to match features to the model
- A method to update the pose estimate
The basic pieces in localisation

- Prediction of Location
- Update of location estimate
- World Model
- Corrected position estimate
- Prediction of Location
- Position Estimate

Robot Tutorial
Henrik I Christensen
Introduction
History
Design of Systems
Kinematics
Sensing
Features
Mapping and Localisation
Planning and Navigation
Lab System
Summary
The issues involves

- Managing the uncertainty for the pose estimate
  - 1 vs Many hypotheses for pose
- Selecting a model for the environment
- Modelling of the system
- Efficient Implementation(s)
Approach

- Representations
- Uncertainty in vehicle model
- Updating the pose estimate
  - Selection of different models for uncertainty
Prediction of vehicle motion

- Consider the process as sense-move-sense-....
- As part of the movement step there is a need to estimate the new pose of the robot and the associated uncertainty in the position.
- Prediction is based entirely on odometric information and a model of the robot as discussed in kinematic modelling.
- Uncertainty is modelled by covariance propagation as discussed in under sensing.
Assume a pose estimate of

\[
p = \begin{bmatrix}
x \\
y \\
\theta
\end{bmatrix}
\]

A motion of left and right wheel by \( \Delta s_l \) and \( \Delta s_r \) respectively. The distance between the wheel is assumed to be \( 2l \).
Consequently:

\[
\begin{align*}
\Delta s &= \frac{\Delta s_l + \Delta s_r}{2} \\
\Delta \theta &= \frac{\Delta s_r - \Delta s_l}{2l} \\
\Delta x &= \Delta s \cos \left( \theta + \frac{\Delta \theta}{2} \right) \\
\Delta y &= \Delta s \sin \left( \theta + \frac{\Delta \theta}{2} \right)
\end{align*}
\]
Pose prediction – Differential drive robot

Or in condensed form:

\[
p' = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta s \cos \left( \theta + \frac{\Delta \theta}{2} \right) \\ \Delta s \sin \left( \theta + \frac{\Delta \theta}{2} \right) \\ \frac{\Delta s_r - \Delta s_l}{2l} \end{bmatrix}
\]
Pose Prediction – Uncertainty estimate

- Need to provide an estimate of uncertainty in position $\Sigma_{p'}$
- Assume the initial uncertainty is $\Sigma_p$
- Assume motion uncertainty is
  \[
  \Sigma_\Delta = \begin{bmatrix}
  k_r|\Delta s_r| & 0 \\
  0 & k_l|\Delta s_l| 
  \end{bmatrix}
  \]
- Assumption of uncertainty, and proportional to distance travelled.
- $k_i$ is determined by calibration
Pose prediction – Uncertainty estimate

- Update can be generated by covariance propagation, i.e.:
  \[ \Sigma_{p'} = \nabla_p f \Sigma_p \nabla_p f^T + \nabla_{\Delta_{rl}} f \Sigma_{\Delta} \nabla_{\Delta_{rl}} f^T \]

- Where \( \nabla_p f \) and \( \nabla_{\Delta_{rl}} f \) is the Jacobians for the pose and the motion, respectively.
Pose Updating – Pose/Motion Jacobian

\[ \nabla_p f = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta s \sin \left( \theta + \frac{\Delta \theta}{2} \right) \\ 0 & 1 & \Delta s \cos \left( \theta + \frac{\Delta \theta}{2} \right) \\ 0 & 0 & 1 \end{bmatrix} \]

\[ \nabla_{\Delta r l} = \begin{bmatrix} \frac{1}{2} \cos \left( \theta + \frac{\Delta \theta}{2} \right) - \frac{\Delta s}{4l} \sin \left( \theta + \frac{\Delta \theta}{2} \right) & \frac{1}{2} \cos \left( \theta + \frac{\Delta \theta}{2} \right) + \frac{\Delta s}{4l} \sin \left( \theta + \frac{\Delta \theta}{2} \right) \\ \frac{1}{2} \sin \left( \theta + \frac{\Delta \theta}{2} \right) - \frac{\Delta s}{4l} \cos \left( \theta + \frac{\Delta \theta}{2} \right) & \frac{1}{2} \sin \left( \theta + \frac{\Delta \theta}{2} \right) + \frac{\Delta s}{4l} \cos \left( \theta + \frac{\Delta \theta}{2} \right) \\ \frac{1}{4l} & \frac{1}{4l} \end{bmatrix} \]
Pose prediction – Linear Motion Example

Error Propagation in Odometry
Pose prediction – Circular Motion Example
Pose Updating

- Given an estimate of where the robot might be $p'$
- Acquire sensor data
- Extract Features
- Match features to model
- Update the pose estimate based on feature information
Pose updating – Uncertainty Model

- The selection of an uncertainty model
  - Single hypothesis
  - Sum of Gaussians
  - Probability grid
  - Topological Graph
  - Particle Based
The selection of an uncertainty model influences the updating methodology.

The uncertainty model is coupled to the environmental representation.

The model influences strongly the computational requirements.
Uncertainty Modelling – Markov Approach

- Assume the world is divided into places/states $s \in P$
- Estimation of $p(s_t)$ given $s_{t-1}$ and sensory data $z_t$
- Formally

$$p(s_t|z_t) = \int p(s_t|s'_{t-1}, z_t)p(s_{t-1})ds'_{t-1}$$

Integration needed as $s_t$ could be reached from multiple locations
Uncertainty modelling – Markov Approach

- Markov assumption: all knowledge encoded in the pose/state estimate
- There is a probability model for motion updating
- There is a model for \( p(z|s) \) i.e. a sensor model, as

\[
p(s|z) = \frac{p(z|s)p(s)}{p(z)}
\]

where \( p(s) \) is location model and \( p(z) \) is the sensor noise model
- These assumptions are relative weak
Topological modelling – dervish example

<table>
<thead>
<tr>
<th></th>
<th>Wall</th>
<th>Closed door</th>
<th>Open door</th>
<th>Open hallway</th>
<th>Foyer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nothing detected</td>
<td>0.70</td>
<td>0.40</td>
<td>0.05</td>
<td>0.001</td>
<td>0.30</td>
</tr>
<tr>
<td>Closed door detected</td>
<td>0.30</td>
<td>0.60</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>Open door detected</td>
<td>0</td>
<td>0</td>
<td>0.90</td>
<td>0.10</td>
<td>0.15</td>
</tr>
<tr>
<td>Open hallway detected</td>
<td>0</td>
<td>0</td>
<td>0.001</td>
<td>0.90</td>
<td>0.50</td>
</tr>
</tbody>
</table>
Topological modelling – dervish example

- Here the probability updating is used for direct lookup of $p(s|z)$, where $s$ is any of the nodes in the topological map.
- As robot moved through environment the graph is updated with new information.
- The probability table is small and efficient to handle.
- The localisation is coarse (location oriented).
Pose estimation with Gaussian Model

- The pose is approximated by a single Gaussian function

\[ p(s) = \frac{1}{\sqrt{2\pi \Sigma_s}} \exp \left( \frac{1}{2} (s - \bar{s})\Sigma_s^{-1}(s - \bar{s})^T \right)^2 \]

- \( s \) is here a continuous function and \( \Sigma_s \) is the associated uncertainty estimate
- Updating is normally performed using a Kalman filter model
Kalman filter – State space model

\[ s_t = F s_{t-1} + G u_t + w_t \]
\[ z_t = H s_t + v_t \]

where \( F \) is the system model, \( G \) is the deterministic input, \( H \) is a prediction of where features are in the world, \( w \) is the system noise, and \( v \) is the measurement noise.
Detour – Probability Updating

- Assume two measurement $x_1$ and $x_2$ with associated uncertainties $\sigma_1$ and $\sigma_2$. How does one generate an optimal estimate $\hat{x}$?
- Doing a weighted least square

$$S = \sum_{i=1,2} w_i (\hat{x} - x_i)^2$$

what are the optimal weights $w_i$?
- From $\frac{\partial S}{\partial \hat{x}} = 0$ we get . . .
Detour – Probability Updating

\[ \hat{x} = \frac{\sum w_i q_i}{\sum w_i} \]

with \( w_i = \frac{1}{\sigma_i^2} \) we get

\[ \hat{x} = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} x_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} x_2 \]

and

\[ \sigma_{\hat{x}} = \frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2} \]
The update can be rewritten to

\[ \hat{x} = x_1 + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} (x_2 - x_1) \]

I.e. the updating = the value + a correction term
The Kalman model

- Composed of a prediction and an update of the estimate
- Prediction is the same as kinematic prediction of pose
- Update is based on the sensory feedback
Kalman Prediction

\[ s_{t|t-1} = F s_{t-1|t-1} + G u_t \]
\[ \Sigma_{t|t-1} = F \Sigma_{t-1|t-1} F^T + Q_t \]

where \( Q \) is the uncertainty of the kinematic motion (odometric uncertainty)
Kalman Updating

\[
\begin{align*}
    s_{t|t} &= s_{t|t-1} + K_t (z_t - Hs_{t|t-1}) \\
    K_t &= \Sigma_{t|t-1} H^T S_t^{-1} \\
    S_t &= H\Sigma_{t|t-1} H^T + R_t \\
    \Sigma_{t|t} &= (I - K_t H) \Sigma_{t|t-1}
\end{align*}
\]
Kalman interpretation

- $F$ is the kinematic update
- $H$ is the measurement prediction $p(z|s)$ ie a prediction of where features in the world are in the sensory frame
- $\Sigma$ is the uncertainty in the pose/state estimate $s_{t|t}$.
- $S_t$ is the uncertainty in the sensory measurements
- $R_t$ is the sensor noise
Use of the Kalman filter

- Prediction / Update as described above
- Matching typically based on the Mahalanobis distance

\[ M = z_t S_t^{-1} z_t^T \]

- A validation gate may be used: \( M < \rho \)
Kalman Example – Prediction

- Kinematic prediction of position and uncertainty
- Standard model
Kalman Example – Features
Kalman Example – Matching

using nearest neighbour and predicted features matching is easy
Matched features generates an error in estimates (purple - model), (green - measured), and (red - update)

Updating is now trivial
Kalman – Discussion

- Makes an assumption of a single pose estimate
- By far the most frequently used model
- Easy to compute.
- Estimation of $F$ and $H$ can be difficult. For non-linear systems that are Jacobians and must be computed for each step.
Sum of Gaussians

- Sometimes there might be multiple interpretations of the feature matches
- Each interpretation generates a hypothesis for the robot pose
- Multiple Distributions are generated and a number of them are used in “different” Kalman updates.
- When a model receives no matches or the uncertainty grows too large, i.e. $\text{trace}(\Sigma) > \delta$ then the model is terminated.
- Can be computationally challenging
Sum of Gauss/Hypothesis – Door Detection
Sum of Gauss/Hypothesis – More Evidence
Probability Grids

Space can be tessellated and the update can be performed directly in pose space

\[ p(s) = \alpha \sum_{s' \in P} p(s')p(s|s', o) \]

where \( \alpha \) is a normalising factor.
Grid updating during motion
Grid updating example

Path of the robot

Belief states at positions 2, 3 and 4
Probability grids

- For accurate localisation in pose space the process can become challenging.
- Say, a $100 \times 100$ m space with a resolution of 10 cm and an angular resolution of 0.1 deg the space is $1000 \times 1000 \times 3600$ or $\approx 3.6$ GB of data that must be updated in each time step.
- Various approximation (windowing) methods can be used.
Monte-Carlo Based Methods

- Monte-Carlo based methods is using a sample model for approximation of the pose estimate
- Using a grid model as presented earlier
- Assume with have a number of particles in a collection

\[ S_t = \left\{ (s_t^{(i)}, \pi_t^{(i)}) | i = 1..N \right\} \]

each particle is a hypothesis for the position of the robot, and \( \pi_t^{(i)} \) is an associated weight

- We can now approximate \( p(s_t | z_0, z_1, ... z_t) \) for any distribution of the pose hypotheses
Monte-Carlo Strategy

1. Draw \(N\) samples from an initial PDF. Typically a uniform distribution. Give each sample a weight of \(\frac{1}{N}\).

2. Propagate the motion information and draw a new sample from the distribution \(p(s_{t+1}^{(i)}|s_t^{(i)}, o_t)\).

3. Set the weight of the sample to \(\pi_{t+1}^{(i)} = p(z_{t+1}|s_{t+1}^{(i)}) \times \pi_t^{(i)}\) based on sensory input.

4. Generate a new sample set by drawing samples from the current set and a basis distribution (typically uniform). Normalize the weights.

5. Go back to step 2.
Monte-Carlo Example – Burgard, Fox & Thrun
Monte-Carlo Discussion

- Efficient to approximate any distribution of the pose
- The number of particles can be adopted to a particular platform
- Can be used both for simple and multi robot localisation
Concurrent map and position estimation in a single framework

Formulated as an extended estimation problem

1. The robot pose (p)
2. The collection of world features (lines, points, vision data)

Can be formulated as EKF, Bayes net or Particle Based

video
Navigation and Path Planning

- Where am I?
- Where do I need to go?
- How do I get there?
Navigation competencies

- Challenges
  - Knowledge is only partial
  - The execution is often associated with uncertainty
- Often control is decomposed into high level functions termed behaviour or functions
  - Wall following, localisation, exploration, door traversal
- Two planning issues
  - Global path planning and
  - Local obstacle handling
Global Path Planning

- Typical assumption: there is an adequate model/map of the environment for navigation
  - Given task: the model could be topological, metric or a mixture

- Steps:
  - Generate a representation of the map for planning
  - Populate the map with an “distance metric”
  - Perform a search from present configuration to a goal configuration
Typical path planing examples

J-P. Lamond et al, LAAS
Typical planning representations

- Configuration space
- Visibility Graph
- Voronoi Diagramme
- Cell Decomposition
- Potential Fields
Configuration space

Approach
- Transform robot into a point
- Morph map to handle the reverse transform
The dimensionality of the C space is #DOF
The space can be very sparsely populated.
Details at http://msl.cs.uiuc.edu/planning/
The space for mobile platforms?
Path planning overview

- Road-map/Graph Definition
- Selection of nodes?
- Space decomposition
  - Free Space vs Obstacles
  - Method of tessellation?
Visibility Graph

- Connect “visible” vertices of the space
- Perform graph search to generate an itinerary
Voronoi Graph

- Path of maximum distance from the obstacles
- Search for shortest path along edges
- Example programme
Cell Decomposition – Planning Strategy

- Divide free space ($F$) into simple connected regions termed “cells”
- Generate a connectivity graph with connected open regions
- Locate “goal” and “start” cells.
- Search for a path through the graph
- Generate a motion strategy through the cells
Exact Cell Decomposition

- Simple strategy
- Results could be complex
- Not easy to design a motion strategy
- Good for sparse environments
Approximate Cell Decomposition

- Easy to implement
- Use of standard methods for search such as wavefront / distance transform
- Widely used for simple environments
Adaptive Cell Decomposition

- Efficient representation of space
- Quad-trees is a well known problem in computational geometry
- Well suited for sparse work space
Potential Field Path Planning

- Consider the robot as a particle in a potential field $U(q)$
- The goal serves as an attractor
- Obstacles represent repulsive forces
- When the potential field is differentiable, the force

$$F(q) = -\nabla U(q)$$

specifies locally how to move through the field.

- Potential fields can be represented by harmonics which allow use of the superposition principle.
  - Generate potential field for the goal point
  - Generate potential field for each obstacle independently
  - The sum of the potential field is also a potential field
Potential field example
Potential field details

- The superposition:
  \[ U(q) = U_{goal}(q) + \sum U_{obst_i}(q) \]

- Or as forces
  \[ F(q) = F_{goal}(q) - \sum F_{obst_i}(q) \]

- The design determines the actual layout
Consider a standard parabolic potential field

\[ U_{goal}(q) = \frac{1}{2} k_{goal} \rho_{goal}^2(q) \]

where \( \rho \) is the distance to the goal

Now the attractive force is

\[ F_{goal}(q) = -\nabla U = -k_{goal}(q - q_{goal}) \]

I.e. a linear force dependence towards the goal
Obstacle Potentials

- Obstacles should have a space bounded influence
- Strong repulsive force close to obstacles
- Less influence at a distance

\[ U_{obst}(q) = \begin{cases} 
\frac{1}{2} k_{obst} \left( \frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 & \text{if } \rho(q) < \rho_0 \\
0 & \text{if } \rho(q) \geq \rho_0 
\end{cases} \]
For convex obstacles the function is differentiable
Consequently:

\[ F_{\text{obst}}(q) = \begin{cases} 
  k_{\text{obst}} \left( \frac{1}{\rho(q)} - \frac{1}{\rho_0} \right) \frac{1}{\rho^2(q)} & \text{if } \rho(q) < \rho_0 \\
  0 & \text{if } \rho(q) \geq \rho_0
\end{cases} \]

The superposition is now trivial
Potential fields can be precomputed.
Sometimes referred to as navigational templates.
Obstacle avoidance

- In general the environment is not fully modelled
- The environment might be dynamically changing

⇒

- A need to modify a planned strategy online to handle such “unexpected” obstacles
- There are a rich variety of algorithms from basic re-planning to reactive changes in the control strategy.
- All of them are termed obstacle avoidance
Obstacle avoidance relies on
- Local context information (where am I?)
- Recent sensory information (a local map)
- Information about the goal position

To generate a revision of the plan, so the revision is usually local by nature.
Bug Algorithm

- The simplest possible strategy
- Hit (H) and Leave (L)
- Do a full tour and then leave from point of minimum distance
- Inefficient but does the job
Bug-2 Algorithm – Obvious improvement

- Do a traversal but exit on line to target
- More efficient for open spaces
- Could be trapped in maze structures
Vector Field Histogram

- Generate histogram of obstacles
- Find passable regions
- Compute best strategy

\[ C = a \theta_{\text{goal}} + b(\theta_R - \theta_{\text{goal}}) + c(\theta_R(t) - \theta_R(t - 1)) \]

weights determine the behaviour
VFH update

- Need to take kinematics into account
- “Mini” config approach
- Only feasible trajectories considered
Curvature based methods

- Robot moves on arcs
- Inclusion of dynamic and kinematic constraints
- Curvature $c = \omega / v$
- Constraints
  - $-v_{max} < v < v_{max}$
  - $-\omega_{max} < \omega < \omega_{max}$
- Divide curvature space into passable regions
- Select best option for goal achievement
Dynamic Window Approach

- Select window of possible motions and map to a configuration space
- Optimize over heading, speed and distance to obstacles to ensure fast and safe motion
Obstacle Avoidance – Summary

- Selection of an efficient method for handling of obstacles in all setting is a challenge
- Handling of dynamic obstacle such as people is very difficult
- Often a local potential field approach might be adequate
Outline

1. Introduction
2. History
3. Design of Systems
4. Kinematics
5. Sensing
6. Features
7. Mapping and Localisation
8. Planning and Navigation
9. Lab System
10. Summary
Laboratory Systems

- Evolution Robotics - Scorpion System
- Differential Drive System
- Linux (RH 7.3) laptop for control
- Quadrature encoders on motors
- IR ranging for close range obstacle handling
- Logitech QR4000 wideangle USB camera system
- ERSP – development environment
Scorpion

- QC 4000 Camera (1024x768)
- Linux Laptop (RH 7.3)
- Bumper
- IR Range Sensor(s)
Scorpion Pieces
ESRP Introduction

- The Evolution Robotics Software Platform (3.0)
- Includes all layers from hardware abstraction to recognition and control
- Has C++ API
- Description of behaviours in XML
- Facilities for behaviour composition without programming!-)}
ERSP Layers

Python

TEL
- Tasks
- Primitive Tasks

BEL
- Behavior Networks
- Behaviors

HAL
- Resources
- Drivers

Behavior Networks
- XML files

Behavior Schema

Hardware Configuration
- XML files

Resource Schema
ERSP Behaviouir Composer - To get started
ERSP Coordinate system
ERSP Material

- 10 robots with software installed!
- Includes software for the following days
- ERSP Getting Started (for programming)
- ERSP Users Guide (the reference)
- ERSP Tutorial Guide – Get started here
Todays exercise

- Try out the behaviour composer (see tutorial guide - net 1)
- Write a small behaviour to go to a particular position (hint: check out the example code!) - (with a port for stream access?)
- If adventurous include handling of obstacles
- Get used to the system!
The week assignment

- **Monday** – basic robot control
- **Tuesday** – try out grammar system and speech
- **Wednesday** – recognize objects
- **Thursday** – build a small graph for navigation
- **Friday** – what can you integrate it to be?
Organisation

- Divide into teams of 3-4 people, preferably not people from the same lab!
- Robots are in the robot lab (ground floor) and in room 22:an which is on the 1st floor of Teknikringen 14
- The KTH people will show the way
Outline

1. Introduction
2. History
3. Design of Systems
4. Kinematics
5. Sensing
6. Features
7. Mapping and Localisation
8. Planning and Navigation
9. Lab System
10. Summary
Summary

- A brief look into some of the key components of robotics
- Kinematics for modelling of the vehicle
- Sensors for description of the environment
- Mapping and Estimation for Localisation
- Simple path planning and planning.
- Try to get the robots moving around.
- Space will be crammed! Sorry
- All will get a copy of software on a DVD.
THE END!