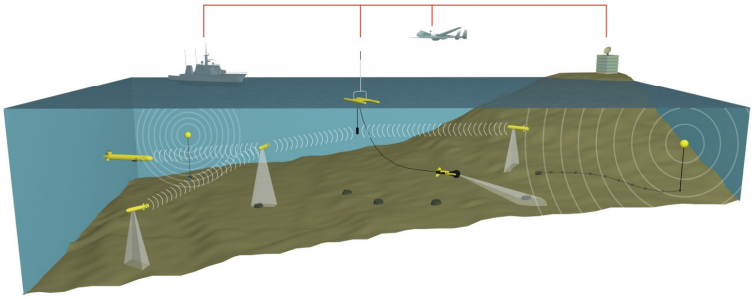
Autonomous Robotics

- Introduction
- Motion
- Perception
- Control
- Concluding Remarks
- LEGO Mindstorms

DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY



Lecture 3: Robot Perception, Motion & Control

Dalhousie University

September 23, 2011

Lecture Outline

- Introduction
- Motion
- Perception
- Control
- Concluding Remarks
- LEGO Mindstorms

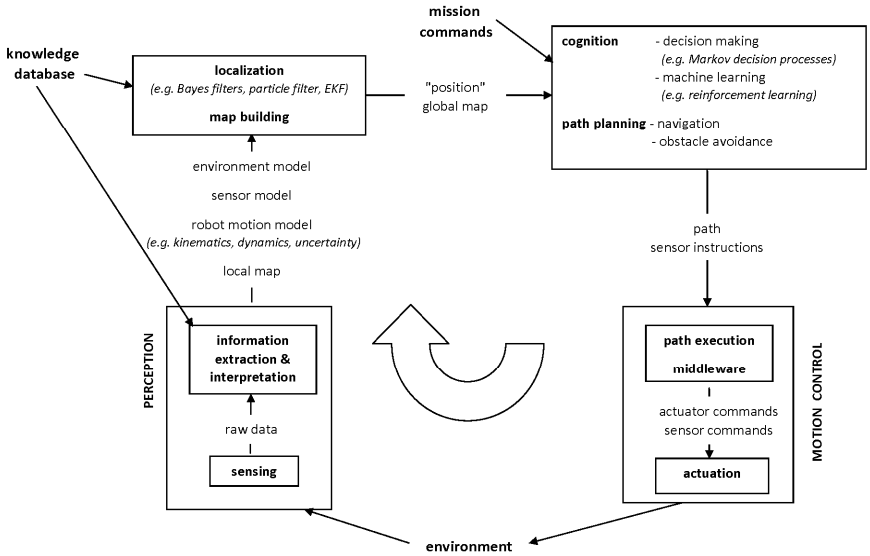
DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

- Introduction
- Motion
- Perception
- Control
- Concluding Remarks
- LEGO Mindstorms

Control Scheme for Autonomous Mobile Robot

- Introduction
- Motion
- Perception
- Control
- Concluding Remarks
- LEGO Mindstorms

DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY



Control Scheme for Autonomous Mobile Robot

- Introduction
- Motion
- Perception
- Control
- Concluding Remarks
- LEGO Mindstorms

DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

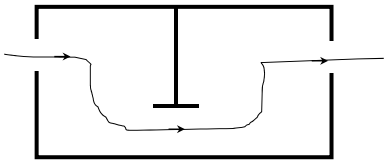
- have identified the components covered in the course outline in the larger context of the control scheme for autonomous mobile robots
  - provide perspective on where the components fit
- *motion control* addresses the robot's low level locomotive ability
  - robot is implementing decisions on the path to navigate and the sensor settings and modes to implement
  - motion capabilities of the robot are quantitatively evaluated through the mobile robot kinematics
  - the robot then goes and interacts with the *environment*

Control Scheme for Autonomous Mobile Robot

- the robot *perception* of its environment is through its sensors and its interpretation of the sensed measurements
  - includes strategies for extracting high level information like features from range-based sensing data
  - armed with locomotion mechanisms and hardware and software for perception the mobile robot moves and perceives the world
- for the sensor information to be relevant the robot takes on the more complex task of *localization*
- how a robot navigates, interprets data, localizes and controls its motion is addressed through the even more complex tasks of deliberation under *cognition* and *path-planning*

Control Scheme for Autonomous Mobile Robot – the plan

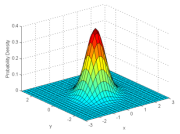
- Thomas will cover generalized Bayesian filters for localization next week
- Mae sets up the background for him, today, by discussing motion and sensor models as well as robot control
- Mae then follows on Bayesian filters to do a specific example, underwater SLAM



Probabilistic Robotics

- with the advent of probabilistic robotics the robotics world is divided into classical robotics and probabilistic robotics
- the strength of probabilistic robotics is its ability to *account for uncertainty* in sensing, modelling, and control
- the trade-off is increase computational complexity and the requirement to model by approximation
- end result: more robust to real-world considerations and uncertainty
- uncertainty captured through probability distribution functions

$$p(x) = \frac{1}{\sqrt{\det(2\pi\Sigma)}} e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)} \qquad p(x) \sim N(\mu, \Sigma)$$



Probabilistic Estimation Methods

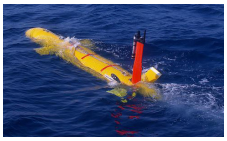
- have two steps: *prediction* and *update*
  - estimate the system state in the form of a probability distribution function, PDF (which Thomas covered)
  - *prediction* propagates the PDF according to the *robot commands* together with a *motion model* for the robot
  - *update* step corrects the prediction by merging the predicted PDF with *sensor information*
  - new estimate given by updated PDF & process iterated
  - In each iteration the prediction step accounts for info lost due to error in the motion model while the update step incorporates information gained from sensors

Definitions

- *state* – all aspects of the robot or environment that can impact the future
- *pose* – location and orientation or attitude for a robot relative to a global coordinate system; *pose without attitude is referred to as location*
- *dead-reckoning* – estimating the current position of a body in 3 space by applying knowledge of a previous position on the basis of assumed distance and direction travelled
  - the intended direction of travel may differ from actual direction due to winds or currents
  - prior to GPS that is what aircraft and ship used
  - problem is position error grows unbounded with time
  - in absence of absolute references this is the fall-back

Motion or Motion-Planning

- objective: enable the robot to automatically compute its motions from high-level descriptions of tasks and models acquired through sensing
- more relevant now than ever
- robots were initially developed for industrial manufacturing and were stationary
  - now, they are engaged tasks that are less repetitive and environments that are less structured for e.g. medical surgery, ocean and space applications, search and rescue, etc.



Prediction Stage Needs Robot Motion Model

- motion of autonomous robots is inherently uncertain, non-deterministic, or stochastic due to the robot, environment, and the interaction between the two
- uncertainty stems from incomplete knowledge of all three
- the Kalman filter (example of more general class of probabilistic estimation techniques) provides a recursive method of estimating the *state* of a dynamic system in the presence of noise
- in the context of localization, the Kalman filter output is a distribution of likely robot positions instead of a single deterministic position estimate
- how to capture or model the motion uncertainty so it can be used as an input to the Kalman Filter?

Kalman Filter requirements

- motion model
- sensor model

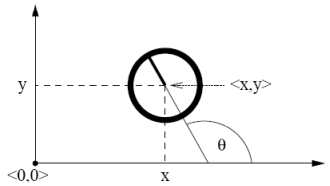
- environment model
- map

Prediction Stage
Robot Motion Model

- the probabilistic motion model is the *state* transition probability expressed as a conditional probability density function: $p(x_t \mid x_{t-1}, u)$
  - this is posterior probability that a command, input, action, or control, u, transitions robot from pose or *state* x_{t-1} to x_t
  - the task is to model $p(x_t \mid x_{t-1}, u)$ based on the equations of motion (the physics or kinematics) of the robot
  - calculate the robot's new pose or state based on velocity and time elapsed (dead-reckoning)

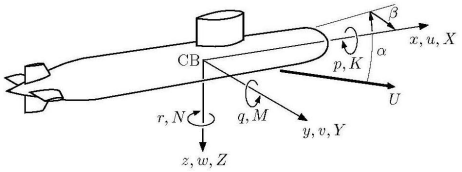
Robot Motion Model

- a rigid body's configuration can be described by six variables
  - 3 translational and 3 rotational (yaw, pitch, and roll) degrees-of-freedom
  - the example of a wheeled robot (LEGO Mindstorm) on a flat surface would be described by a two-dimensional *state space* spanned by (x, y, θ)



Robot Motion Model

- the more general example of a 6 degree-of-freedom autonomous underwater vehicle would have a six-dimensional state space spanned by: *(x,y,z,θ,ϕ,ψ)*



$$\alpha = \tan^{-1}\frac{w}{u}, \quad \beta = \tan^{-1}\frac{-v}{u}$$

Robot Motion Models

- two classes of motion models
  1. odometry – when wheel encoders are used (e.g. LEGO Mindstorm and Roomba robots)
     - if used for positioning, assume no wheel slippage
  2. velocity or dead-reckoning as in AUVs
- odometers are more accurate than velocity measurements since they are a displacement and a velocity is an integration of a displacement, however:
  - odometry is not available until after executing a motion command – not usable for motioning planning
  - practice is to *use odometry models for estimation and velocity models for motion planning*

Algorithms for PDF to Model Uncertainty

DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

- for a normal distribution

1. Algorithm **prob_normal_distribution** (a,b):
2. return $\frac{1}{\sqrt{2\pi\, b^2}} \exp\left\{-\frac{1}{2}\frac{a^2}{b^2}\right\}$

b^2 = variance centered at 0

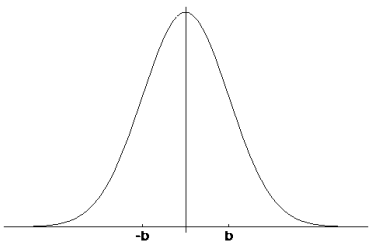
- for a triangular distribution

1. Algorithm **prob_triangular_distribution** (a,b):
2. return $\max\left\{0, \frac{1}{\sqrt{6}\, b} - \frac{|a|}{6\, b^2}\right\}$

Error Distributions for PDF to Model Uncertainty

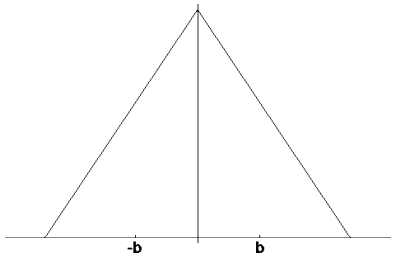
DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

Normal distribution



$$\varepsilon_{\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\frac{x^2}{\sigma^2}}$$

Triangular distribution



$$\varepsilon_{\sigma^2}(x) = \begin{cases} 0 \text{ if } |x| > \sqrt{6\sigma^2} \\ \frac{\sqrt{6\sigma^2} - |x|}{6\sigma^2} \end{cases}$$

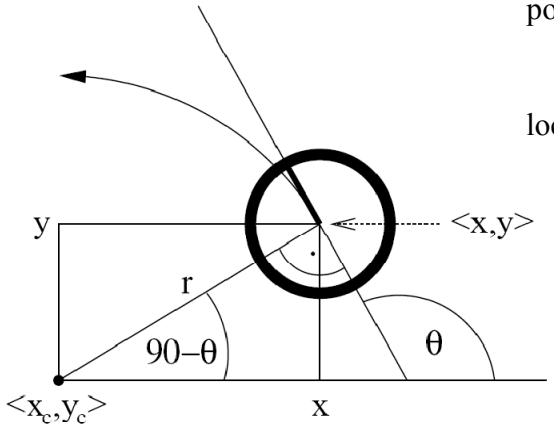
1. Velocity Motion Model Nomenclature

DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

- robot pose in global coordinate system

$$\text{pose} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}$$

$$\text{location} = \begin{pmatrix} x \\ y \end{pmatrix}$$



1. Velocity Motion Model Equations for 2D Model

DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

$x_t = (x', y', \theta')^T$ is noise - free robot state after Δt

Center of circle:

$$\begin{pmatrix} x^* \\ y^* \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -\lambda \sin\theta \\ \lambda \cos\theta \end{pmatrix} = \begin{pmatrix} \frac{x+x'}{2} + \mu(y - y') \\ \frac{y+y'}{2} + \mu(x' - x) \end{pmatrix}$$

with

$$\mu = \frac{1}{2} \frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}$$

1. Velocity Motion Model Algorithm

1: **Algorithm motion_model_velocity**(x_t, u_t, x_{t-1})**:**

2: $\mu = \frac{1}{2} \frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}$

3: $x^* = \frac{x + x'}{2} + \mu(y - y')$

4: $y^* = \frac{y + y'}{2} + \mu(x' - x)$

5: $r^* = \sqrt{(x - x^*)^2 + (y - y^*)^2}$

6: $\Delta\theta = \text{atan2}(y' - y^*, x' - x^*) - \text{atan2}(y - y^*, x - x^*)$

7: $\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$

8: $\hat{\omega} = \frac{\Delta\theta}{\Delta t}$

9: $\hat{\gamma} = \frac{\theta' - \theta}{\Delta t} - \hat{\omega}$

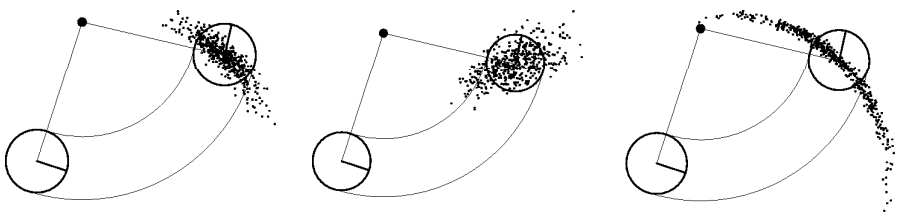
10: *return* $\textbf{prob}(v - \hat{v}, \alpha_1|v| + \alpha_2|\omega|) \cdot \textbf{prob}(\omega - \hat{\omega}, \alpha_3|v| + \alpha_4|\omega|) \cdot \textbf{prob}(\hat{\gamma}, \alpha_5|v| + \alpha_6|\omega|)$

$x_{t-1} = (x\ y\ \theta)$
$x_t = (x\ y\ \theta)^T$
$u_t = (v\ \omega)^T$
α_i = robot specific motion error parameters
$x_t = (x', y', \theta')^T$ is noise - free robot state after Δt

controls for error-free robot

1. Velocity Motion Model Sampling Algorithm

- more general Bayesian filters (which Thomas will address next class) sample from the motion model at discrete time steps as opposed to calculating the posterior probability
- density: given x_t compute $p(x_t \mid x_{t-1}, u)$
- sampling: given u_t and x_{t-1} generate a random x_t as described in motion model $p(x_t \mid x_{t-1}, u)$



velocity motion model (sampled) for different noise parameter settings

1. Velocity Motion Model Sampling Algorithm for Filters

- generate random samples from $p(x_t \mid x_{t-1}, u)$ for given u_t and x_{t-1} to obtain random pose x_t

1: **Algorithm sample_motion_model_velocity**(u_t, x_{t-1})**:**

2: $\hat{v} = v + \textbf{sample}(\alpha_1|v| + \alpha_2|\omega|)$

3: $\hat{\omega} = \omega + \textbf{sample}(\alpha_3|v| + \alpha_4|\omega|)$

4: $\hat{\gamma} = \textbf{sample}(\alpha_5|v| + \alpha_6|\omega|)$

5: $x' = x - \frac{\hat{v}}{\hat{\omega}}\sin\theta + \frac{\hat{v}}{\hat{\omega}}\, sin(\theta + \hat{\omega}\Delta t)$

6: $y' = y + \frac{\hat{v}}{\hat{\omega}}\cos\theta - \frac{\hat{v}}{\hat{\omega}}\, cos(\theta + \hat{\omega}\Delta t)$

7: $\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$

8: *return* $x_t = (x', y', \theta')^T$

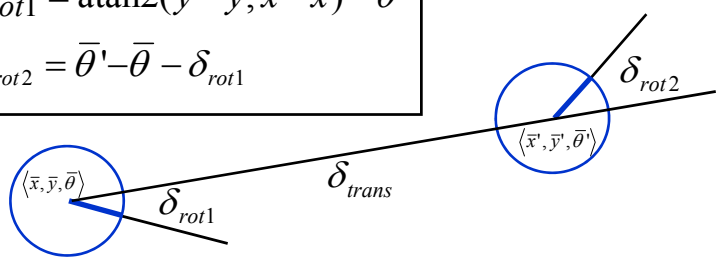
2. Odometry Motion Model Model Equations

- robot transitions from pose $(\bar{x}, \bar{y}, \bar{\theta})$ to $(\bar{x}', \bar{y}', \bar{\theta}')$ after Δt
- thus u_t is modelled by an initial rotation (δ_{rot1}), translation (δ_{trans}), and another rotation (δ_{rot2})

$$\delta_{trans} = \sqrt{(\bar{x}'-\bar{x})^2 + (\bar{y}'-\bar{y})^2}$$

$$\delta_{rot1} = \text{atan2}(\bar{y}'-\bar{y}, \bar{x}'-\bar{x}) - \bar{\theta}$$

$$\delta_{rot2} = \bar{\theta}'-\bar{\theta} - \delta_{rot1}$$



2. Odometry Motion Model Noise Model

- measured motion= true motion + noise

$$\hat{\delta}_{rot1} = \delta_{rot1} + \varepsilon_{\alpha_1 |\delta_{rot1}| + \alpha_2 |\delta_{trans}|}$$

$$\hat{\delta}_{trans} = \delta_{trans} + \varepsilon_{\alpha_3 |\delta_{trans}| + \alpha_4 |\delta_{rot1} + \delta_{rot2}|}$$

$$\hat{\delta}_{rot2} = \delta_{rot2} + \varepsilon_{\alpha_1 |\delta_{rot2}| + \alpha_2 |\delta_{trans}|}$$

2. Odometry Motion Model Algorithm (given *x, x', u*)

1. Algorithm **motion_model_odometry** *(x,x',u)*
2. $\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$
3. $\delta_{rot1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$
4. $\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$

   odometry values (u)
5. $\hat{\delta}_{trans} = \sqrt{(x' - x)^2 + (y' - y)^2}$
6. $\hat{\delta}_{rot1} = \text{atan2}(y' - y, x' - x) - \bar{\theta}$
7. $\hat{\delta}_{rot2} = \theta' - \theta - \hat{\delta}_{rot1}$

   values of interest (x,x')
8. $p_1 = \text{prob}(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1 |\hat{\delta}_{rot1}| + \alpha_2 \hat{\delta}_{trans})$
9. $p_2 = \text{prob}(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3 \hat{\delta}_{trans} + \alpha_4 (|\hat{\delta}_{rot1}| + |\hat{\delta}_{rot2}|))$
10. $p_3 = \text{prob}(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1 |\hat{\delta}_{rot2}| + \alpha_2 \hat{\delta}_{trans})$
11. return $p_1 \cdot p_2 \cdot p_3$

2. Odometry Motion Model Sampling Algorithm for Filters

1. Algorithm **sample_motion_model** (u, x):

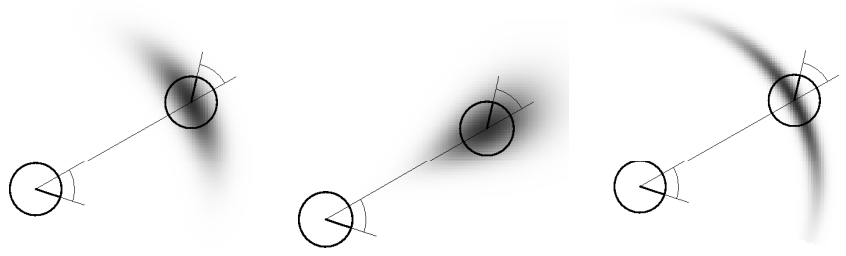
$u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle, x = \langle x, y, \theta \rangle$

1. $\hat{\delta}_{rot1} = \delta_{rot1} + \text{sample}(\alpha_1 \mid \delta_{rot1} \mid + \alpha_2 \mid \delta_{trans} \mid)$
2. $\hat{\delta}_{trans} = \delta_{trans} + \text{sample}(\alpha_3 \mid \delta_{trans} \mid + \alpha_4 (\mid \delta_{rot1} \mid + \mid \delta_{rot2} \mid))$
3. $\hat{\delta}_{rot2} = \delta_{rot2} + \text{sample}(\alpha_1 \mid \delta_{rot2} \mid + \alpha_2 \mid \delta_{trans} \mid)$
4. $x' = x + \hat{\delta}_{trans} \cos(\theta + \hat{\delta}_{rot1})$
5. $y' = y + \hat{\delta}_{trans} \sin(\theta + \hat{\delta}_{rot1})$
6. $\theta' = \theta + \hat{\delta}_{rot1} + \hat{\delta}_{rot2}$
7. Return $\langle x', y', \theta' \rangle$

sample_normal_distribution

2. Odometry Motion Model Effect of Noise

- motion model with varying noise parameters (unsampled)



Motion and Maps

- the environment, as represented through a map, has not figured in the picture so far
- for some problems a map, m, details where the robot may or may not navigate
- occupancy maps distinguish free (traversable) from occupied terrain
  - robot can only occupy free space
- map-based motion model: $p(x_t \mid u_t, x_{t-1}, m)$
  - calculates likelihood a robot placed in a world of map m arrives at pose x_t upon executing u_t at pose x_{t-1}
- yields better results than a map-free motion model
  - however, difficult to get a closed tractable form

Motion and Maps

- approximation for map-based motion model that is good when $x_{t-1} - x_t$ is small (i.e. < half robot dimension):

$$p(x_t \mid u_t, x_{t-1}, m) = \eta \frac{p(x_t \mid u_t . x_{t-1}) p(x_t \mid m)}{p(x_t)}$$

- η is for normalization

early Mars Rover (JPL)

Posterior Probability with Map

1: Algorithm **motion_model_with_map** (x_t, u_t, x_{t-1}, m)
2: Return $p(x_t|u_t,x_{t-1}) \cdot p(x_t|m)$

1: Algorithm **sample_motion_model_with_map** (u_t,x_{t-1},m):
2: do
3: x_t = **sample_motion_model**(u_t,x_{t-1})
3: $\pi = p(x_t|m)$
4: until $\pi > 0$
5: Return (x_t,π)

$p(x_t \mid m)$ is the 'consistency' of pose x_t with m

in an occupancy map $p(x_t|m) = 0$ iff robot would be in a cell already occupied

Motion Model Summary

- motion models: odometry-based and velocity-based systems
- ways to calculate the posterior probability $p(x| x', u)$ and how to sample from $p(x| x', u)$ to provide input to Bayes filters
- model calculations are done in fixed time intervals Δt
- robot-specific parameters of the models have to be learned
- extended motion model that takes the map into account.

Fire-X UAV

Perception

DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

- process by which the robot uses its on-board sensors to obtain information about the *state* of the environment, e.g. an AUV uses its side scan sonar to gather images regarding objects on the sea floor
- in the process it obtains a *measurement*, *observation* or *percept* which adds to its *belief* on the environment state
- these measurements have a latency associated with them and hence report on a past (albeit, recent) environment state

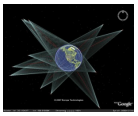
Single axis optical gyro

3-axis optical gyro

Robot Sensors Quantities Measured

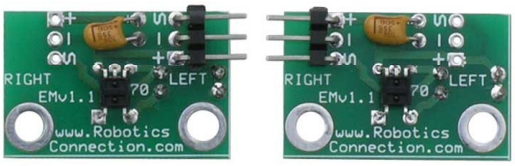
DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

- contact:
  - bumper
- internal:
  - accelerometers (spring-mounted masses)
  - gyroscopes (spinning mass, laser light)
  - compasses, inclinometers (earth magnetic field, gravity)
- proximity or range finding:
  - sonar (time of flight)
  - radar (phase and frequency)
  - laser range-finders (triangulation, time-of-flight, phase)
  - infrared (intensity)
- visual:
  - cameras
- satellite-based : GPS



Robot Sensors
E.g. Wheel Encoders

- require +5V and GND to power them, and provide a 0 to 5V output
- provide +5V output when they "see" white, and a 0 V output when they "see" black.



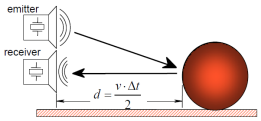
- disks are manufactured out of high quality laminated color plastic to offer a very crisp black to white transition
  - enables a wheel encoder sensor to easily see the transitions

Robot Sensors
Interaction with Environment

- wide variety of sensors used in mobile robots
  - proprioceptive: measure values internal to the robot, for e.g. speed through water, distance from the bottom, battery voltage, attitude
  - exteroceptive: measure information about the robot's environment, for e.g. distance measurements, sound amplitude, conductivity, temperature, and density
  - passive sensors measure ambient environmental energy entering the sensor
  - active sensors emit energy into the environment then measure the environment reaction
- example of proximity sensor will be developed for probabilistic robotics

Proximity Sensors

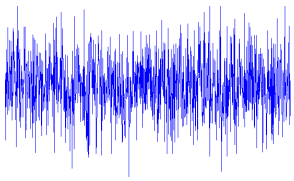
- *proximity* (or *range finding*) sensors are widely used on autonomous robots
  - measure the proximity (range, distance, displacement) from a robot to an object
  - can be achieved through emitting a beam from a laser or a cone from an ultrasonic sensor
- measurement can be caused by …
  - a known obstacle
  - cross-talk
  - an unexpected obstacle (people, furniture, …)
  - void of real obstacles (total reflection, glass, …)



sonar

Proximity Sensors Uncertainty

- noise is due to uncertainty …
  - in measuring distance to known obstacle
  - in position of known obstacles
  - in position of additional obstacles
  - whether obstacle is missed
- modeling sensors probabilistically captures reality



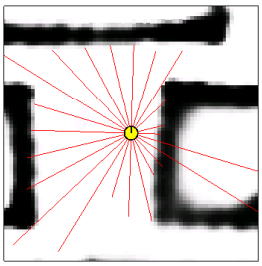
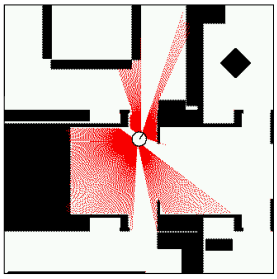
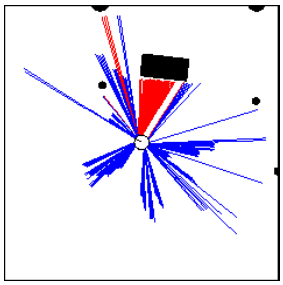
Probabilistic Sensor Models

DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

- *deterministic* sensor models do not capture the uncertainty inherent in sensor measurement, noise, or their interaction with the environment
- *probabilistic* model captures the uncertainties that affect a sensor measurement through modelling the noise
- probabilistically, the measurement process is modelled as a *conditional* probability, $p(z_t \mid x_t, m)$

Probabilistic Proximity Sensors

DALHOUSIE UNIVERSITY *Inspiring Minds*
ACADIA UNIVERSITY

- determine $p(z_t \mid x_t, m)$, i.e., the probability of a measurement z_t given that the robot is at position x_t within map, m
- a quick word about maps in the probabilistic sensor world

Probabilistic Sensors - Mapping

- environment or world in which a measurement is made is described through a map, m
  - details objects and their location within the environment

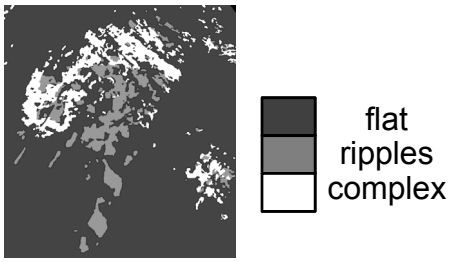
$$m = \{m_1, m_2, ..., m_N\}: \quad \begin{array}{l} m_n = \text{an object property,} \\ N = \text{total number of objects} \end{array}$$

- a robot may occupy different locations in this environment
- map indexing:
  - feature-based maps:
    - n is a feature index and m_n contains property feature as well as feature location
  - location based maps (e.g. occupancy grid map):
    - n corresponds to a specific location

Probabilistic Proximity Sensor Models

- where does the probability density functions like $p(z_t \mid x_t, m)$ come from?

- proximity probabilistic sensor models include:
  1. beam
  2. scan
  3. feature



underwater environment sensing

1. Beam Sensor Model

- usable for sonar or laser based sensors
- range may be measured along a beam

- scan z consists of k measurements

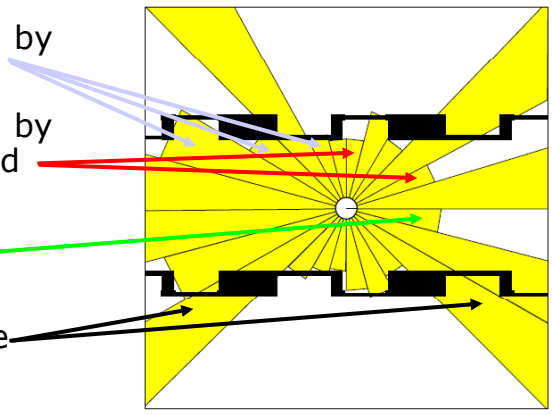
$$z = \{z_1, z_2, \ldots, z_K\}$$

- individual measurements are independent given the robot position

$$P(z_t \mid x_t, m) = \prod_{k=1}^{K} P(z_{t,k} \mid x_t, m)$$

1. Beam Model Sensor Errors Proximity Measurements

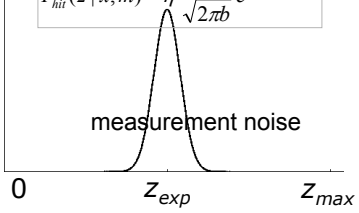
1. beams reflected by obstacles
2. beams reflected by persons / caused by crosstalk
3. random measurements
4. maximum range measurements



1. Beam Sensor Model PDFs to Model Uncertainty

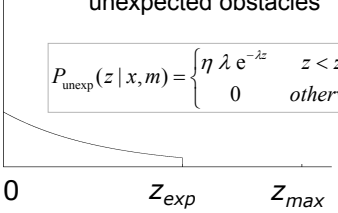
$$P_{hit}(z \mid x,m) = \eta \frac{1}{\sqrt{2\pi b}} e^{-\frac{1}{2}\frac{(z-z_{exp})^2}{b}}$$

measurement noise



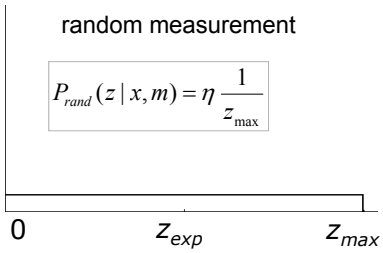
unexpected obstacles

$$P_{unexp}(z \mid x,m) = \begin{cases} \eta \, \lambda \, e^{-\lambda z} & z < z_{exp} \\ 0 & otherwise \end{cases}$$



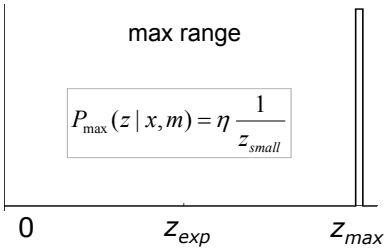
random measurement

$$P_{rand}(z \mid x,m) = \eta \frac{1}{z_{max}}$$



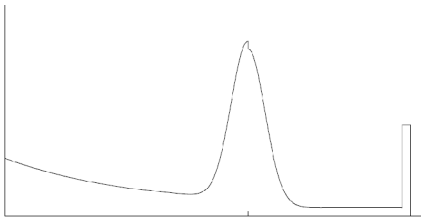
max range

$$P_{max}(z \mid x,m) = \eta \frac{1}{z_{small}}$$



1. Beam Sensor Model PDF Incorporating Uncertainty

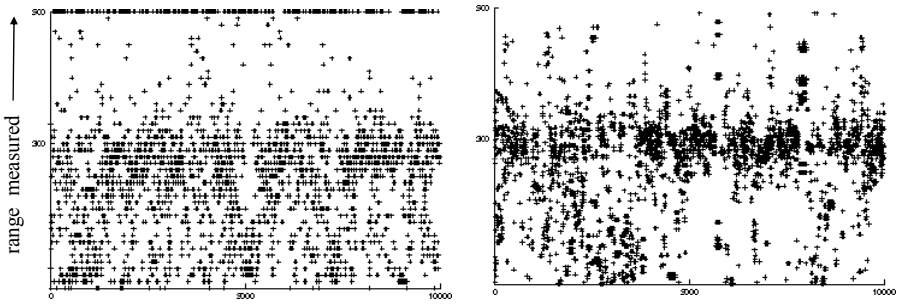
- probability density function that accounts for the four error sources discussed
- determine parameters based on analyzing or learning from data (offline or on-the-fly)
  - an area where machine learning has made some good contributions!



$$P(z \mid x,m) = \begin{pmatrix} \alpha_{hit} \\ \alpha_{unexp} \\ \alpha_{max} \\ \alpha_{rand} \end{pmatrix}^T \cdot \begin{pmatrix} P_{hit}(z \mid x,m) \\ P_{unexp}(z \mid x,m) \\ P_{max}(z \mid x,m) \\ P_{rand}(z \mid x,m) \end{pmatrix}$$

1. Beam Sensor Model PDF Adjusting Intrinsic Parameters

- measured distances for expected 300 cm and maximum range of 500 cm
- ultrasound has more measurement and detection noise
- laser more accurate but reports false ranges



sonar sensor laser range sensor

1. Beam Sensor Model Limitations

- discontinuous or suffers from lack of smoothness in a cluttered environment (like a rock strewn surface)
  - belief representations may not capture the correct states as adjacent states may be very different
  - optimizations will find many local extrema making it difficult to establish the global extremum
- computationally intense

1. Beam Sensor Model Summary

- assumes independence between beams.
  - models physical causes for measurements
  - mixture of densities for these causes
  - assumes independence between causes, problem?
- implementation
  - learn parameters based on real data
  - different models should be learned for different angles at which the sensor beam hits the obstacle
  - determine expected distances by ray-tracing
  - expected distances can be pre-processed

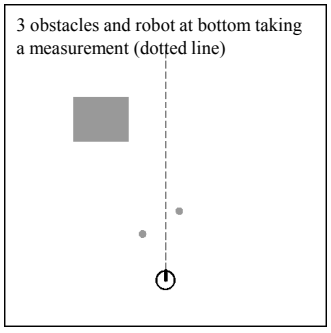
2. Scan Sensor Model Likelihood Field - Pros

- better sensor model
- ad hoc but probabilities are smoother in cluttered spaces and computations more efficient (pre-comps in 2D) – used in lasers
  - not as sensitive to robot pose
- project sensor scan end point into the global coordinate space of the map based on geometrical and trigonometric constraints
- captures following noise sources and uncertainty:
  - measurement noise
  - failures
  - unexplained random measurements

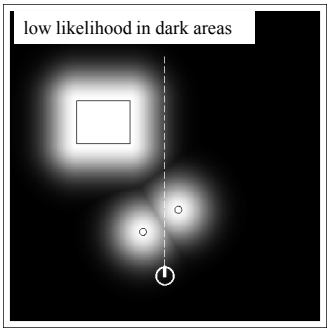
2. Scan Sensor Model Likelihood Field - Cons

- likelihood model disadvantages
  - does not model obstacles and dynamics that yield short readings
  - sensors can see through wall (no ray casting)
  - map uncertainties not accounted for
- can remedy by refining map occupancy stages (e.g. occupied, free, and unknown)

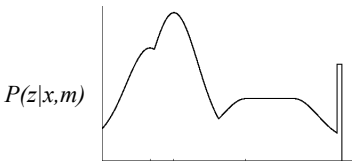
2. Scan Sensor Model Likelihood Field



map *m* of environment



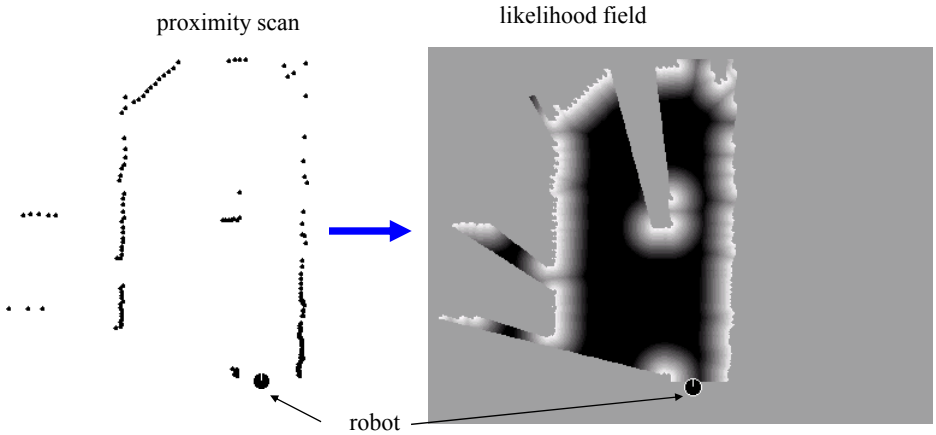
likelihood field



2. Scan Sensor Model Scan matching

DALHOUSIE UNIVERSITY Inspiring Minds
ACADIA UNIVERSITY

- extract likelihood field from scan and use to match different scan



2. Scan Sensor Model Summary

DALHOUSIE UNIVERSITY Inspiring Minds
ACADIA UNIVERSITY

- efficient, uses 2D tables only
- smooth with respect to small changes in robot pose
- allows gradient descent, scan matching
- completely ignores physics of the beams

3. Feature-Based Measurement Models

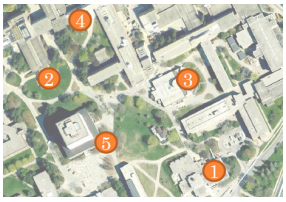
- beam and scan sensor models process raw sensor measurements (sonar, laser, vision)
- alternatively, one could extract features from the measurements
  - reduces computational complexity since small number of features are extracted from high-dimensional sensor measurements
  - e.g. pick out lines, corners, local minima which map to corners, continental shelf, ridges, etc.
  - landmarks are features selected to aid robot navigation (e.g. distinctive rocks, ridges, etc.)

3. Feature-Based Measurement Probabilistic Sensor Model

1. Algorithm **landmark_detection_model** *(z,x,m)*:
   $z = \langle i, d, \alpha \rangle, x = \langle x, y, \theta \rangle$
2. $\hat{d} = \sqrt{(m_x(i) - x)^2 + (m_y(i) - y)^2}$
3. $\hat{a} = \text{atan2}(m_y(i) - y, m_x(i) - x) - \theta$
4. $p_{\text{det}} = \text{prob}(\hat{d} - d, \varepsilon_d) \cdot \text{prob}(\hat{\alpha} - \alpha, \varepsilon_\alpha)$
5. Return $z_{\text{det}} p_{\text{det}} + z_{\text{fp}} P_{\text{uniform}}(z \mid x, m)$

3. Feature-Based Sensor Model

- sensor models that use landmarks measure the range and bearing from robot to landmark in robot local frame
- feature extractor generates a *signature* or value that characterizes the feature
- sensors have the ability to calculate landmark range and bearing relative to the robot's local coordinate frame



4. Correlation-Based Measurement Models

- correlation between a measurement and the map e.g. map matching
  - mosaic together consecutive scans into local maps
  - compare local map, m_{local}, to the global map, m, and calculates local $p(m_{local} \mid x_t, m)$
  - local map transformed into coordinate frame of the global map
  - the more similar m and m_{local}, the larger $p(m_{local} \mid x_t, m)$
  - once both maps are in the same reference frame, they are compared using the map correlation function (not shown)

Sensor Model Summary

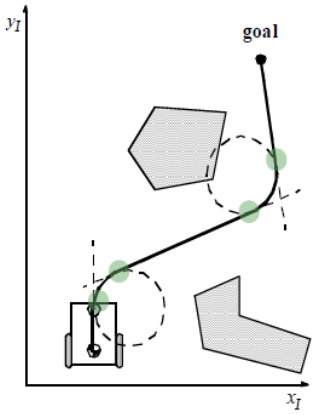
- explicitly modeling uncertainty in sensing key to robustness
- in many cases, good models can be found by the following approach:
  1. determine parametric model of noise free measurement.
  2. analyze sources of noise
  3. add adequate noise to parameters (eventually mix in densities for noise)
  4. learn (and verify) parameters by fitting model to data
  5. likelihood of measurement is given by "probabilistically comparing" the actual with the expected measurement
- this holds for motion models as well

Motion Control

- the robot acts on its environment to effect in itself or the environment some change
- control carries information about the change of state in the environment or robot
- objective of a kinematic controller is to follow a trajectory described by its position and/or velocity profiles as function of time
- motion control is not straight forward because mobile robots are typically non-holonomic and MIMO systems.
- most controllers (including the one presented here) are not considering the dynamics of the system

Open-Loop Control

- trajectory (path) divided in motion segments of clearly defined shape:
  - straight lines and segments of a circle
  - Dubins car, and Reeds-Shepp car
- control problem:
  - pre-compute a smooth trajectory based on line, circle (and clothoid) segments
- disadvantages:
  - It is not at all an easy task to pre-compute a feasible trajectory
  - limitations and constraints of the robots velocities and accelerations
  - does not adapt or correct the trajectory if dynamical changes of the environment occur.
  - resulting trajectories are usually not smooth (in acceleration, jerk, etc.)



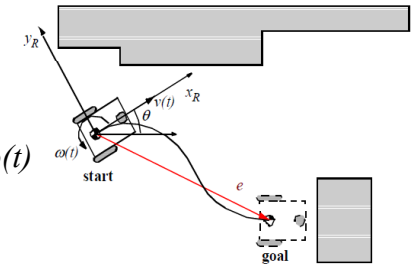
Feedback Control

- find control matrix, K, if it exists so:

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix}$$

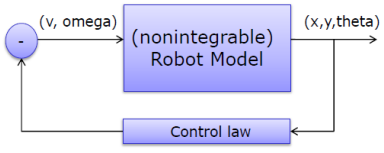
- *with* $k_{ij}=k(t,e)$
- such that the control of $v(t)$ *and* $\omega(t)$

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = K \cdot e = K \cdot \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$



- drives the error *e* to zero

$$\lim_{t \to \infty} e(t) = 0$$

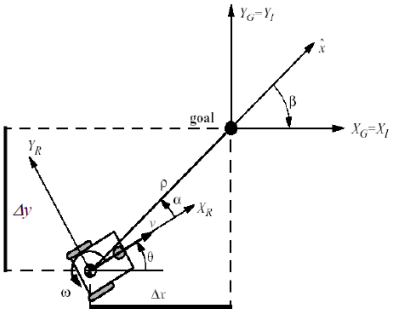


- MIMO state feedback control

Kinematic Position Control

- kinematics of differential drive mobile robot described in the inertial frame $\{x_I, y_I, \theta\}$ is given by,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = K \cdot e = K \cdot \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} v \\ \omega \end{bmatrix}$$



- where x' and y' and are the linear velocities in the direction of the x_I and y_I of the inertial frame.
- α denote the angle between the x_R axis of the robots reference frame and the vector connecting the center of the axle of the wheels with the final position

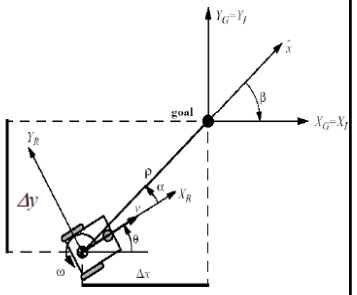
Kinematic Position Control Coordinate Transformation

- coordinates transformation into polar coordinates with origin at goal position:

$$\rho = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\alpha = -\theta + \text{atan2}\,(\Delta y, \Delta x)$$

$$\beta = -\theta - \alpha$$



- system description, in the new polar coordinates:

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -\cos\alpha & 0 \\ \frac{\sin\alpha}{\rho} & -1 \\ -\frac{\sin\alpha}{\rho} & 0 \end{bmatrix} \times \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Kinematic Position Control Control Law

It can be shown, that with

$$v = k_\rho \rho \qquad \omega = k_\alpha \alpha + k_\beta \beta$$

the feedback controlled system

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -k_\rho \rho \cos\alpha \\ k_\rho \sin\alpha - k_\alpha \alpha - k_\beta \beta \\ -k_\rho \sin\alpha \end{bmatrix}$$

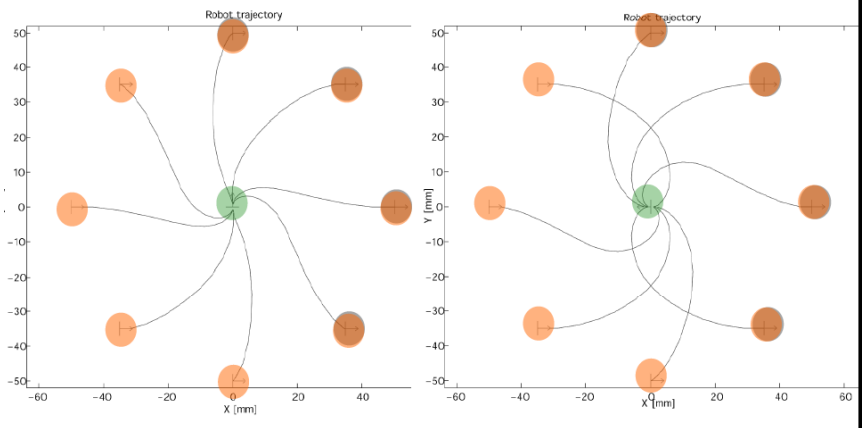
will drive the robot to $(\rho,\alpha,\beta)=(0,0,0)$

The control signal v has always constant sign,

- the direction of movement is kept positive or negative during movement
- parking maneuver is performed always in the most natural way and without ever inverting its motion.

Kinematic Position Control Resulting Path

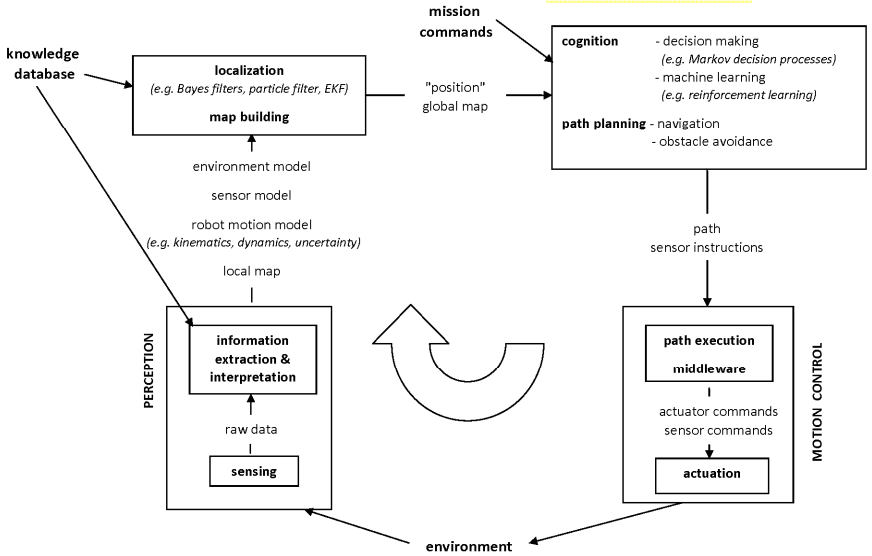
- The goal is in the center and the initial position on the circle.



$$k = (k_\rho, k_\alpha, k_\beta) = (3,8,-1.5)$$

Control Scheme for Autonomous Mobile Robot

- Introduction
- Motion
- Perception
- Control
- Concluding Remarks
- LEGO Mindstorms



LEGO Mindstorms

- Introduction
- Motion
- Perception
- Control
- Concluding Remarks
- LEGO Mindstorms

- sensors and actuators