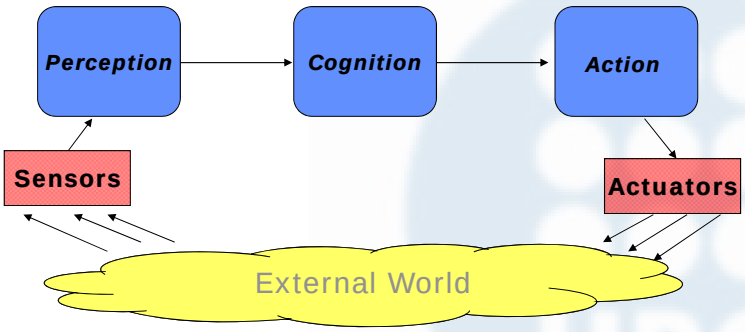
Multiagent Systems (SMA-UPC)

5. Situated Agents (Robots)
Part 2:
Planning and Motion.
Multi-Robot Systems.

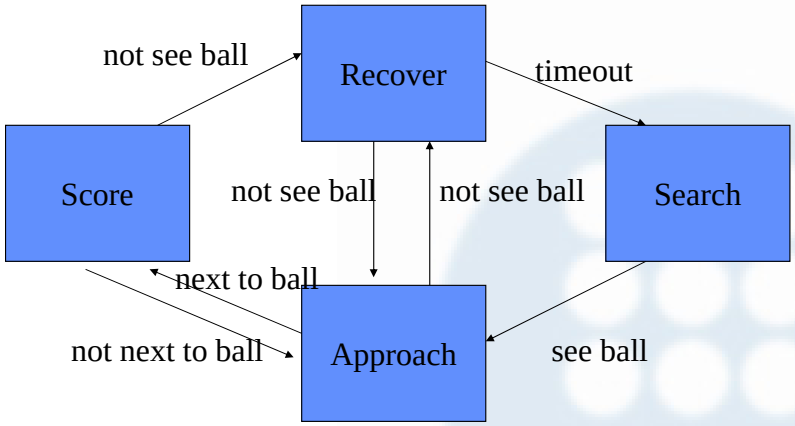
Javier Vázquez-Salceda
SMA-UPC

Task Planning

- Usually most of the tasks are organized in behaviors
  - Kicking, tracking, pushing, grabbing…
- Navigation through the environment is an special behavior to be managed
- Task Planning as behavior selection AND Navigation



Task Planning: Behavior selection



Motion

- **Task Planning**
- **Motion Kinematics**
- **Walking Engine**
- **Frame-based motion**

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
https://kemlg.upc.edu

Behavior control: Motion

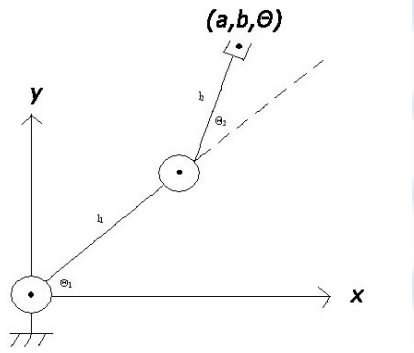
- We will use as example SONY Aibo's motion engine.
  - Four-legged walking (several joints with degrees of liberty)
  - Head motion (2 joints, 3 degrees of liberty)
- How to generate complex behaviors (turning, kicking?)
- Kinematics: relation between the control inputs and the robot motion
  - Forward kinematics problem
    - Given the control inputs, how does the robot move
  - Inverse kinematics problem
    - Given a desired motion, which control inputs to choose

Forward Kinematics

- Determines position in space based on joint configuration

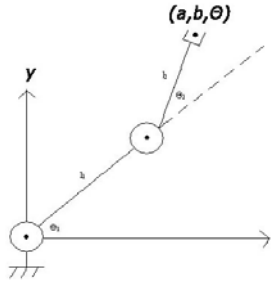
e.g., What is the position & orientation of the tool (end effector) relative to the origin?

Solve for a, b, q in terms of l_1, l_2, q_1, and q_2.



(Figures by Nick Aiwazian)

Forward Kinematics
Solution



Can be solved trigonometrically!

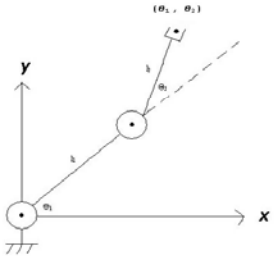
$$a = l_1 \cos q_1 + l_2 \cos q_1 + q_2$$

$$b = l_1 \sin q_1 + l_2 \sin q_1 + q_2$$

$$q = q_1 + q_2$$

Inverse Kinematics

- Going backwards
- Find joint configuration given position & orientation of tool (end effector)
- More complex (path planning & dynamics)
- Usually solved either algebraically or geometrically
- Possibility of no solution, one solution, or multiple solutions

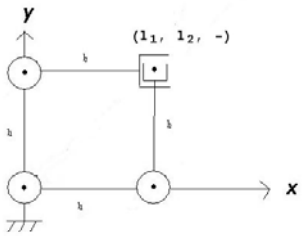


Let's assume $l_1 = l_2$

What is the configuration of each joint if the end effector is located at (l_1, l_2, -)?

(Solve for (θ_1, θ_2) when the tool is at {l1, l2, -})

Inverse Kinematics
Solution



$q_1 = 0,\ q_2 = 90$

Or

$q_1 = 90,\ q_2 = -90$

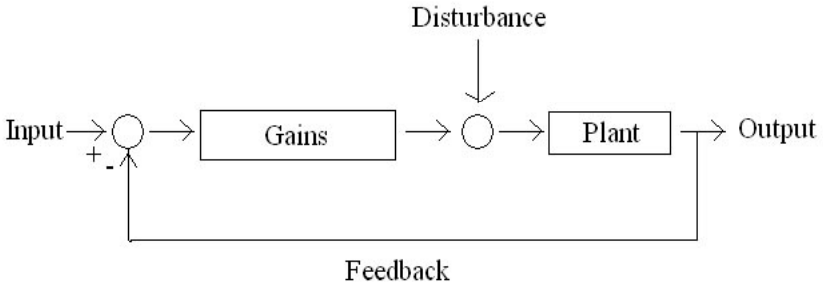
(Two Solutions)

What is PID Control?

- Proportional, Integral, & Derivative Control
  - ***Proportional***: Multiply current error by constant to try to resolve error
  - ***Integral***: Multiply sum of errors by constant to resolve steady state error (error after system has come to rest)
  - ***Derivative***: Multiply time derivative of error change by constant to resolve error as quickly as possible

PID Control

- The Basic Problem:
  - We have n joints, each with a desired position which we have specified
  - Each joint has an actuator which is given a command in units of torque
  - Most common method for determining required torques is by feedback from joint sensors
- The PID Control Loop:



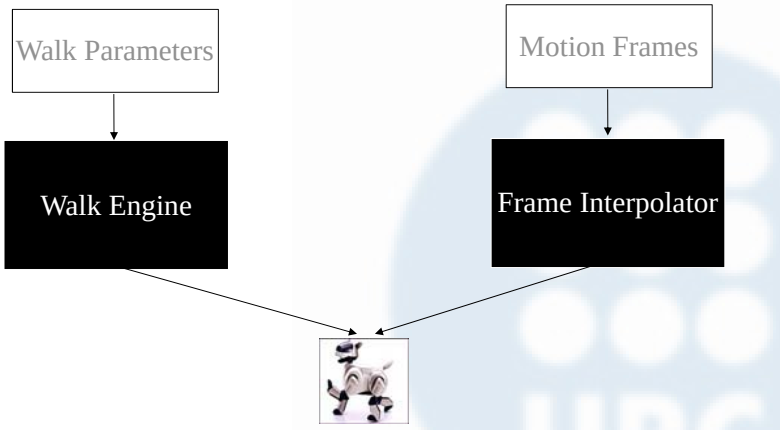
5. Situated Agents (Robots)

Defining movements

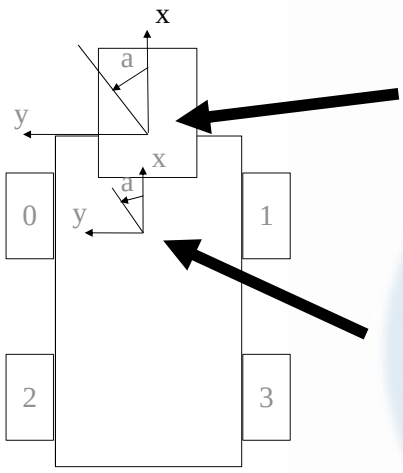
The Motion Interface in AIBO's

Dynamic Walking Motion

Static Frame-Based Motion



Defining movements
Coordinate Frames



Vision Coordinate Frame

Motion Coordinate Frame

Defining movements
Motor Control

- In AIBO's, each message to the motion library contains a set of target angles for the joints
  - Each target is used for a PID controller (part of the AIBO robot) that controls each motor
  - Each target angle is used for one 8ms motor frame
- Each message contains at least 4 motor frames (32ms)

Defining movements
The AIBO Walk Engine

- All of the inverse kinematics have been done for you!
- All you have to deal with are the “motion parameters”
- Your Goal: Create fluid, stable motion

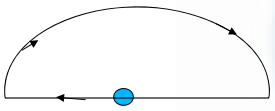
Defining movements
Dynamic Walking Motion

- In the AIBO, a 51-parameter structure is used to specify the gait of the robot.

Leg Parameters:	***Global Parameters:***
Neutral Kinematic Position (3x4)	Height of Body (1)
Lifting Velocity (3x4)	Angle of Body (1)
Lift Time (1x4)	Hop Amplitude (1)
Set Down Velocity (3x4)	Sway Amplitude (1)
Set Down Time (1x4)	Walk Period (1)
	Height of Legs (2)

Defining movements
Motion Parameters

- **Neutral Kinematic Position** (3D vector relative to the motion coordinate frame) - Position of the leg on the ground at *some point* during the walk cycle
- Think of it as the position the legs would be in if the dog was pacing in place using your walk parameters



Path of the leg during 1 cycle

Defining movements
Motion Parameters

- **Lift Velocity** (3D vector) – Velocity (mm/sec) with which the leg is lifted off the ground
- **Down Velocity** (3D vector) – Velocity (mm/sec) with which the leg is placed on the ground
- **Lift Time and Down Time** – This controls the order of the legs by specifying a percentage of the time through the time cycle that each leg is moved

Defining movements
Approaches for Parameter Setting

- Trial and error
  - Tedious, but controlled, and provides knowledge of parameters
- Search
  - Large parameter space, local vs. global optima
- Adaptation
  - Controlled change by feedback

Defining movements
Frame-Based Motion

- Each motion is described by a series of “frames” which specify the position of the robot, and a time to interpolate between frames
- Movement between frames is calculated through linear interpolation of each joint
- E.g.: Kicking
  - A series of set positions for the robot
  - Linear interpolation between the frames
    - Kinematics and interpolation provided by CMWalkEngine
  - Set robot in desired positions and query the values of the joints

Defining movements
Frame-Based Motion

Defining movements
Example: Kicks Behavior

- Modeling *effects* of kicking motions
  - Ball vision analysis
  - Ball trajectory angle analysis
  - Kick strength analysis
- Kick selection for behaviors
  - Selection algorithm
  - Performance comparison

Modeling *effects* of kicking motions
Ball Trajectory Angle

- Estimate the angle of the ball's trajectory relative to the robot
  - Track ball's trajectory after the kick
  - Retain information about ball position in each vision frame
  - Calculate angle of trajectory using linear regression

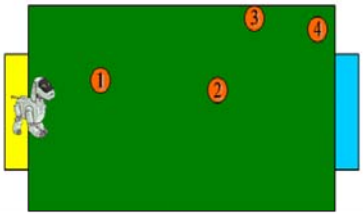
Modeling *effects* of kicking motions
Kick Strength

- Estimate the distance the ball will travel after a kick.
  - Impossible to track entire path of the ball
  - Calculate only the final location of the ball relative to the kick position
  - Estimate failure rate of the kick using distance threshold

Kick selection for behaviors
Selection algorithm

- Incorporate the kick models into the selection algorithm
  - The robot knows its position on the field relative to the goal and the desired ball trajectory
  - The robot selects appropriate kick by referencing the kick model
  - If no kick fits desired criteria, robot selects closest matching kick and turns/dribbles ball to appropriate position

Kick selection for behaviors
Performance analysis



Experiment Results

Point	CMPack'02 (sec)	Modeling & Prediction (sec)
P1	**56.7**	**39.8**
P2	**42.5**	**27.2**
P3	**76.5**	**60.0**
P4	**55.0**	**52.0**
Total	**57.8**	**44.8**

Summary

- Effectively moving a four-legged robot is challenging
- Effectiveness of motion is highly sensitive to motion parameters
- CMWalk provides the kinematics computations, so parameter setting can be at a high level of abstraction.
- Ideally, we would like to set parameters automatically.

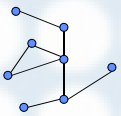
Planning and Motion

- **Motion Planning and Navigation**
- **Mapping**
- **Motion Planning with Uncertainty (Probabilistic Robotics)**

Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
https://kemlg.upc.edu

World Models (I)

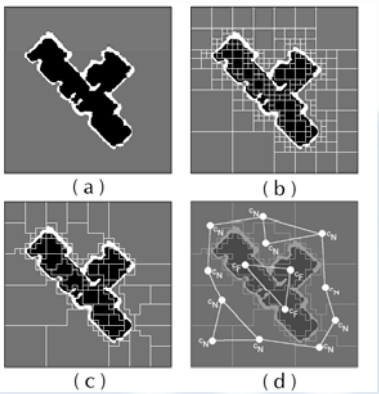
- Representations of the environment are usually built by means of:
  - Metric maps: explicitly reproduce the metrical structure of the domain
    - good for location, hard for planning
    - e.g., *Evidence grids*
  - Topological maps: represent the environment as a set of meaningful regions.
    - good for planning, hard for location



- Best solution: use *both* representations

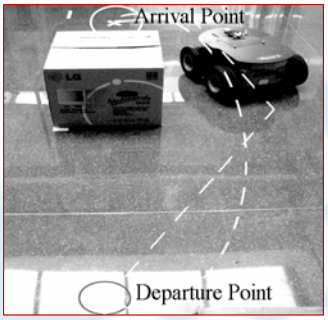
World Models (II)
Topological Map Extraction

- (a) Metric map thresholding
  - cell occupancy values
- (b) Hierarchical split
  - piramidal cell structure
- (c) Interlevel merging
  - homogeneous cells fusion
- (d) Intralevel merging
  - homogeneous cell classification



Navigation (I)

- Navigation consists of finding and tracking a safe path from a departure point to a goal.
- Navigation architectures belong to three broad categories: deliberative, reactive and hybrid.



Navigation (II)

- Deliberative schemes require extensive world knowledge to build high-level plans
  - Usually they use the *sense-model-plan-act* cycle
  - problem 1: inability to react rapidly
  - problem 2: not suitable for (partially) unknown environments.
- Reactive schemes try to couple *sensors* and *actuators* to achieve a *fast* response.
  - Easily combine several sensors and goals,
  - problem 1: the emergent behaviour may be *unpredictable*
  - problem 2: the emergent behaviour may be *inefficient* (prone to fall in local traps).
- Hybrid schemas get the best of both approaches.

Path Planning

Deliberative Architectures

- Global sensor info
  - Builds a global world model based on sensing the environment.
  - Pros
    - Guaranteed to find an existing solution
  - Cons
    - Computationally heavy
    - Requires frequent localization

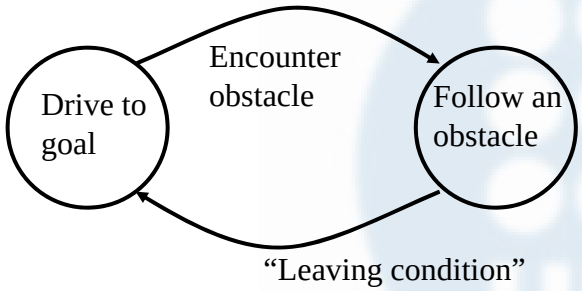
Reactive Architectures

- Local sensor info
  - Navigate using sensors around local objects
  - Pros
    - Much simpler to implement
  - Cons
    - Not guaranteed to converge – will get stuck in a local minima with no hope of escape

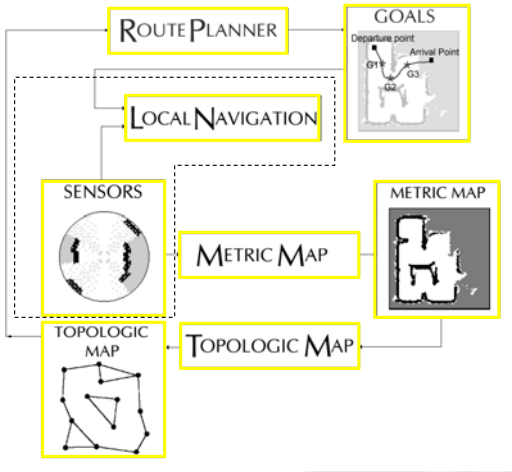
- We want something on the middle: **Hybrid Architectures**
  - get the best of both approaches.

Hybrid Architectures (I)

- Combine local with global information
- Guaranteed to converge if a solution exists

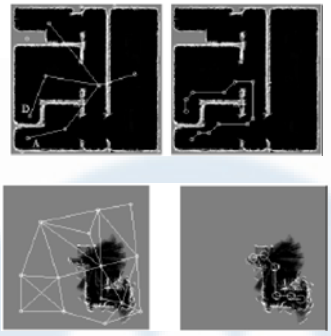


The hybrid architecture



Deliberative Layer

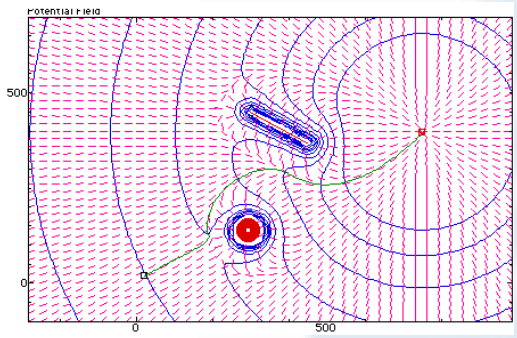
- Path planning algorithm (A*) works at topological level
- Resulting path of nodes linked to the metric map
- Extraction of points of maximum curvature
  - Partial goals



- The reactive layer *flexibly* moves the robot from one partial goal to the next
- Works also with partially explored environments.

Reactive Layer (I)
Classic approach

- Potential fields: Artificial repulsion field around obstacles plus attraction field around the goal.



Reactive Layer (II)
Classic approach

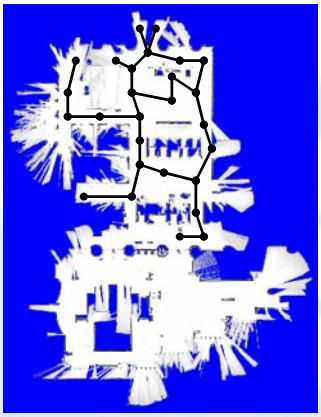
- Advantages:
  - Simple and efficient method
  - No model of the environment is required.
- Drawbacks:
  - Oscillations, local traps
  - The robot always tries to keep as far from obstacles as possible

Mobile Robot Mapping

- *What does the world look like?*
- Robot is unaware of its environment
- The robot must explore the world and determine its structure
  - Most often, this is combined with localization
  - Robot must update its location wrt the landmarks
  - Known in the literature as Simultaneous Localization and Mapping, or Concurrent Localization and Mapping : SLAM (CLM)
  - Example : AIBOs are placed in an unknown environment and must learn the locations of the landmarks

2D Mapping for Mobile Robots

- Extract meaningful spatial data from sensors
- Metric
  - Accurate sensing/odometry
  - Relative positions of landmarks
  - Sensors identify distinguishable features
- Topological
  - Odometry less important
  - Qualitative relationships between landmarks
  - Sensors identify locations



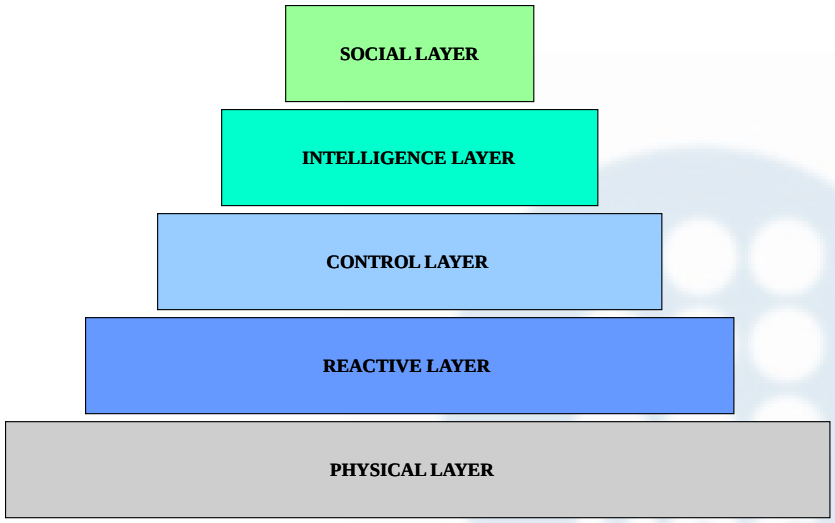
Knowledge Engineering and Machine Learning Group
UNIVERSITAT POLITÈCNICA DE CATALUNYA
https://kemlg.upc.edu

Multi-Robot Systems

- **Coordination, Competition**
- **Strategy**
- **Social Models, Roles, Task Allocation, Teamwork**
- **Mutual Perception**

Intelligent Robot (III)

Layers



General Coordination of Multiple Robots

- Cooperative Sensing
- Cooperative Self-Localization with landmarks
- Stigmergy
- Distributed Problem Solving

Cooperative Sensing

- Communicate sensor data to increase quality of the worldmodel
- Use Kalman filters to fuse measurements
  - A Kalman filter is an optimal estimator - it infers parameters of interest from indirect, inaccurate and uncertain observations. It is recursive so that new measurements can be processed as they arrive.

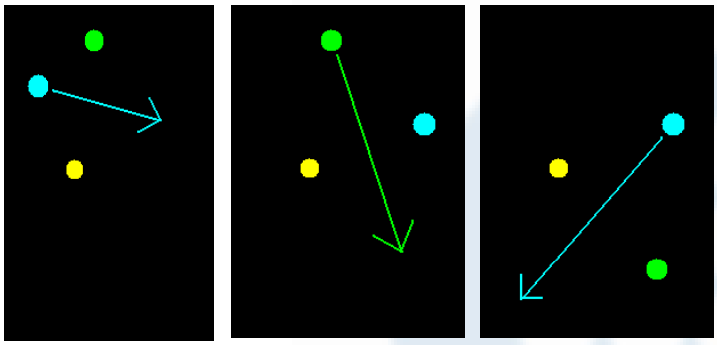
Kalman Filters

- Why is Kalman Filtering so popular?
  - Good results in practice due to optimality and structure.
  - Convenient form for online real time processing.
  - Easy to formulate and implement given a basic understanding.

Cooperative Self-Localization

- Robots often use landmarks to know where they are.
- When you have multiple robots you can use other robots as temporary landmarks.
- Useful in situations where the starting positions of a group of robots are known and the goal is to explore an unknown territory.

Using other robots as landmarks



Communication Problems

- Agents might not always be able to communicate
- Bandwith restraints
- Physically impossible because of objects blocking communication (in Robocup Rescue)
- Possible solution: Stigmergy

Stigmergy

- Stigmergy means that agents put signs, called stigma in Greek, in their environment to mutually influence each other's behavior.
- Useful for indirect communication since no explicit rendezvous amongst the agents is needed.
- Humans use it all the time.

Ants Example

- Multiple ants walk around randomly till they find food.
- They go back with the food, leaving a pheromone trail.
- Other ants will pick up the trail and go back for the rest of the food, strengthening the pheromone trail.
- When the food is gone, the pheromone trail will vanish since it won't be strengthened anymore and the ants will walk around randomly again.
- Here the pheromone trail is the stigma.

Ants Example

- The individual ants are not exposed to the complexity and dynamics of the situation.
- They don't need to keep a worldmodel.
- They don't have to communicate amongst each other about the world.
- They use the world itself to solve the problem.

Stigmergy for Robots

- You could use the same kind of system for robots instead of ants.
- Exploring an unknown terrain and finding objects works pretty good using this technique.
- It's also possible to use it for other problems than exploring.
- Used in a production line, where every tool, robot and object is considered an agent.

Distributed problem solving

- Demands group coherence (agents need to have the incentive to work together faithfully)
- Demands group competence (agents need to know how to work together well)
- Coherence is hard when agents are really self-interested. Agents have to be designed to work together to really make it work.

Advantages of Distributed problem solving

- Speedup in problem solving because of parallelism.
- Possible to use expertise of different agents.
- Certain agents are better suited for certain jobs.
- Beliefs and other data can be distributed.
- The agents can hold their own beliefs and only communicate what they think is necessary. (as opposed to a central based system)

Task Sharing or Task Passing

- When an agent has many tasks to do, it should enlist the help of agents with few or no tasks.

1. Task decomposition
2. Task allocation
3. Task accomplishment
4. Result synthesis

Open Questions

- How to divide tasks among team members?
- How to position robots to fulfill their roles without interferring with their teammates?
- What if a different robot becomes more suitable for the task?

- Solution 1: **Software Agent algorithms** for coordination
  - Good if there is enough CPU resources and time
- Solution 2: Adapt **Artificial Potential Fields** for coordination

Artificial Potential Fields for coordination (I)

- Low computational overhead compared to higher level approaches like path planning
- Require simple, local knowledge about the environment
- Robust in dynamic situations
  - No expensive replanning when environment changes
- Likely to guide robots to local minima
  - But, no major problem in highly dynamic environments

Artificial Potential Fields for coordination (II)

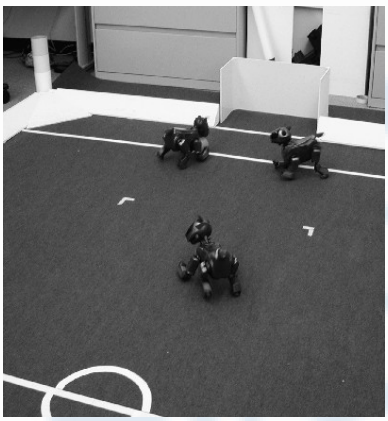
- Potentials encode heuristic information about the environment
- Used to position robots for particular roles
  - Roles must be assigned first!
- Continuous auction with bidding with suitability
- Robot with highest bid wins the task
- If robot becomes unavailable, the robot with next highest bid addresses the task

Artificial Potential Fields
Shared Information

- Small number of robots are collaborating, so just broadcast messages to share information
  - Does not scale to large numbers of robots
- *x* times per second, each robot broadcasts a message to its teammates, containing:
  - Position of the robot according to its localization system
  - Estimate of the uncertainty in that position
  - Robot's estimation of the ball's position
  - The uncertainty associated with that measurement
  - Robot is the goalie?
  - Robot sees the ball?

Artificial Potential Fields
Role Assignment

- Possible role assignment:
  - Primary attacker
  - Offensive supporter
  - Defensive supporter
  - Goalie (fixed)



Artificial Potential Fields
Role Assignment

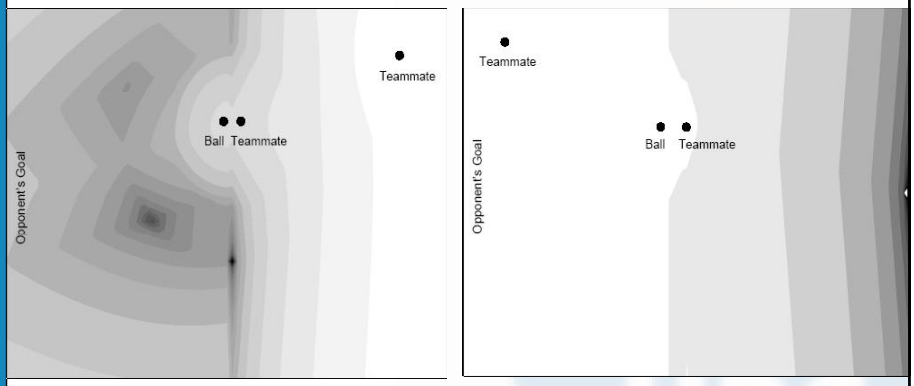
- Robots first calculate their own suitability using local information from their world models
- Use same function to calculate bids of teammates using only shared information
- Compare bids of each teammate; assume best role
- No synchronization needed
  - All robots perform same calculation on same shared data
  - Bid functions are self-reinforcing

Artificial Potential Fields
Coordination

- Robots use same mechanism for both coordination and obstacle avoidance
- Robots sample local points and follow the gradient of the potential field until they reach a local minimum
- The components of the field should create local minima at positions from which the robots can support primary attacker
  - The offensive supporter is guided to a good position to receive passes or recover the ball if the shot on goal goes wide
  - The defensive supporter is guided to a position where it blocks its own goal and can recover the ball if it is intercepted by the opposing team
- Primary attacker does not use the potential field
  - Always seek out the ball
  - Count on teammates to move out of the way instead of avoiding them

Artificial Potential Fields
Illustration Example

- Offensive supporter
- Defensive supporter



Artificial Potential Fields
Coordination

- Potential field is sum of several linear components
- These components either represents heuristic information about the world or obstacle information
- Typically the components of the potential functions are bounded at zero
  - Makes the effect of the terms local
  - Helps prevent undesirable interactins between terms

Artificial Potential Fields
Coordination

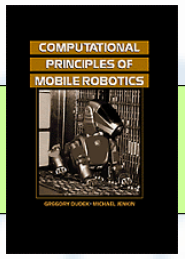
- Only teammates are included in list of robots to avoid
- High fidelity information about locations of opponents is not available
- This is a perceptual problem
  - Composite nature of the functions makes it trivial to add terms for opponents when the perceptual system is able to provide that information

References

[1] Russell, S. & Norvig, P. "Artificial Intelligence: A Modern Approach" Prentice-Hall Series in Artificial Intelligence. 1995 ISBN 0-13-103805-2

[2] Recommended book:

Computational Principles of Mobile Robotics
Gregory Dudek, Michael Jenkin
Cambridge University Press 2000



[3] More information on AIBO robots and OPEN-R
http://openr.aibo.com

[4] Robocup league
http://www.robocup.org

These slides are based mainly in [2], [1] and material from M. Veloso and N. Aiwazan. Special thanks to C. Hees, B. Steunebrink and T. Slijkerman.