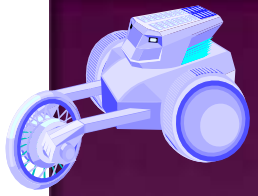
Lecture 6-2

Think Hard, Act Later:
Deliberative Control
Don't think, React!
Reactive Control

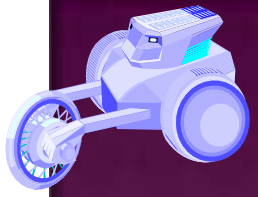
The Robotics Primer (Ch. 13, 14)

C.A. Berry



Course Announcements

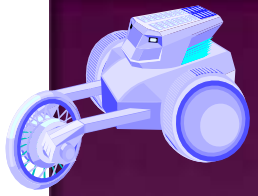
- Quiz on Monday, 4/27/09 on **Deliberative and Reactive Control**
- Lab 4 Demo due **Thursday, 4/23/09**
- Lab 4 Memo and code due by midnight on **Friday, 4/24/09**
- Upload memo and code to Angel
- Bring your laptop and robot everyday
- **DO NOT** unplug the network cables from the desktop computers or the walls



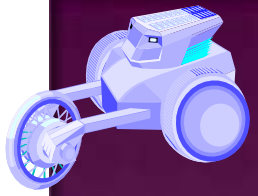
Quote of the Week

"The danger of the past was that men became slaves. The danger of the future is that men may become robots."

Erich Fromm

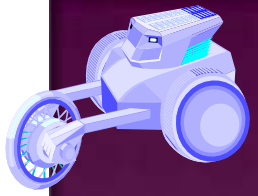


Deliberative Control (Ch. 12)



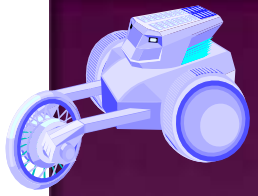
Deliberative Control

- *Deliberative systems* were similar to how humans solved problems such as playing chess
- *Deliberative Control* used on Shakey in the 1960s because the state of the art machine vision was the input to the planner that used it to decide what to do and how to move next



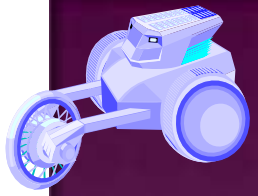
Planning

- *Planning* is the process of looking at the outcome of possible actions and searching for the sequence that will reach the desired goal
- Sometimes searching can be very slow if it is necessary to search the complete representation which may be large
- A partial search may be quicker if the first found solution is used
- With internal models, the robot can search several directions in parallel which cannot be done in the real world



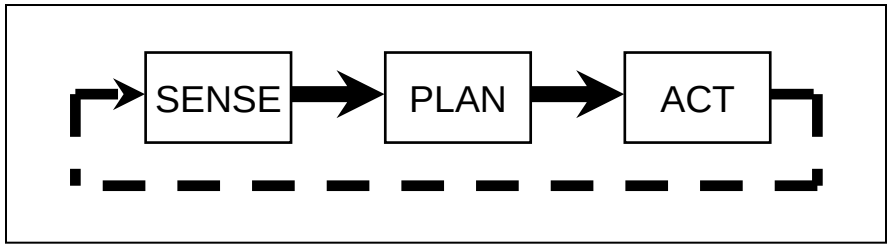
Optimization

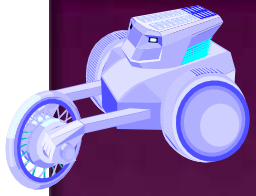
- The process of improving a solution to a problem such as shortest path or safest route for path planning is called *optimization*
- Optimization criteria is used to find the best path and sometimes there may be trade offs when the criteria conflict
- In order to search and plan a solution the world must be represented in states where the robot must go from the start state to the goal state



Costs of Planning

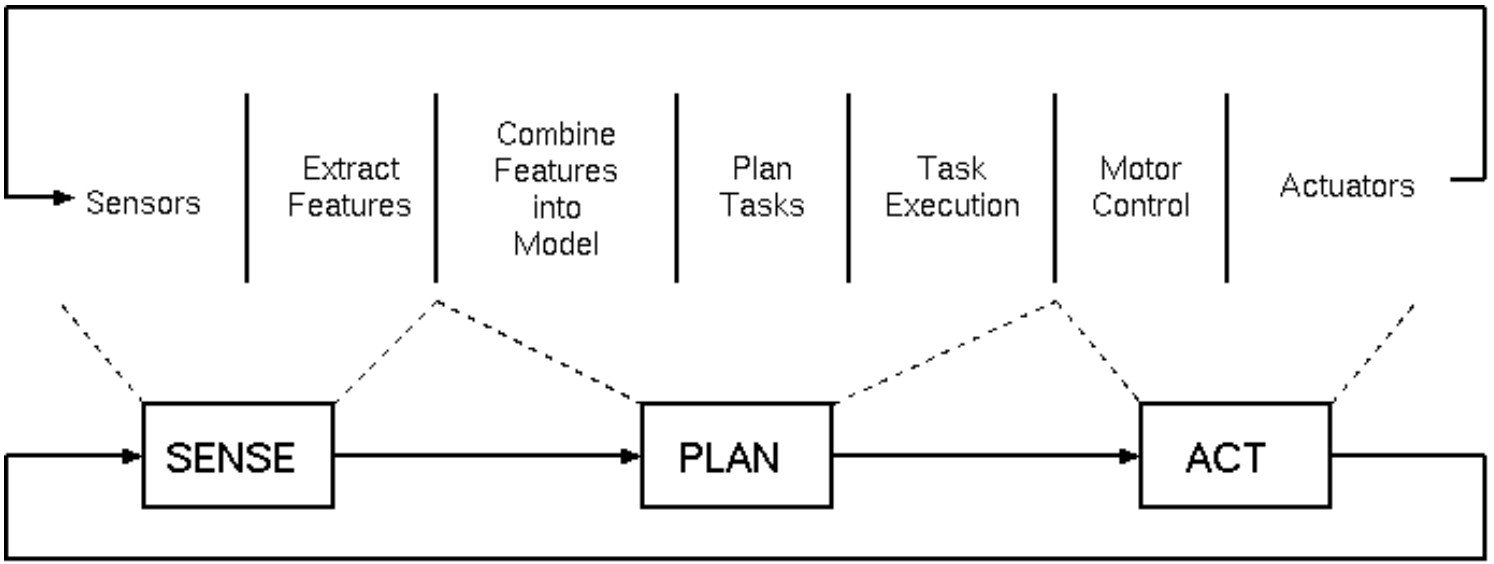
- A robot cannot afford to sit and deliberate in a dynamic environment and must sense and react in real time
- The deliberative planner has the following three steps performed in sequence:

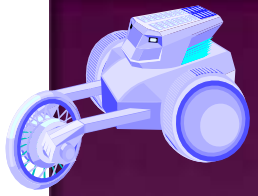




Hierarchical Architecture

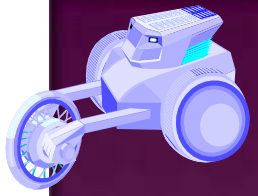
Deliberative Control is hierarchical and horizontal





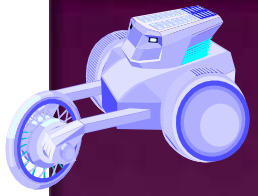
Time Scale Drawback

- It takes a long time to search a large state space
- Combinations of analog and digital sensory data creates a large sensor state space
- When the sensor state space combines with internal models or representations the state space is very large and hard to search
- If the planning is slow compared to the robot's motion it has to stop and wait for the plan to finish
- To make progress, the robot must plan rarely and move as much as possible which is open loop control and bad for dynamic environments



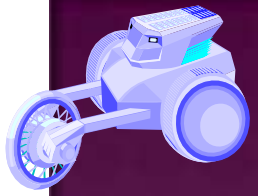
Space Drawback

- It takes a great deal of space or memory storage to represent the robot's state space representation
- Computer memory is cheap but all memory is finite and some algorithms may run out of it



Information Drawback

- The planner assumes that the state space representation is accurate and up to date
- The plan will not be useful is this is not a valid assumption
- The representation used must be updated and check as often as necessary to keep it accurate for the task
- Updating the plan for a real environment requires taking time to update the world model

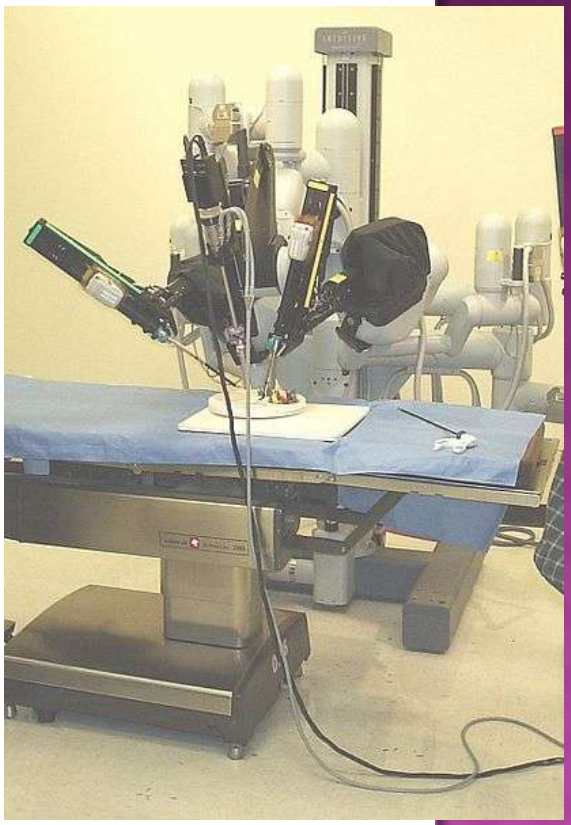


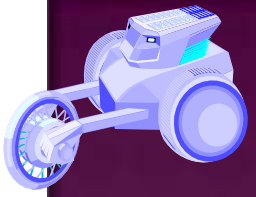
Use of Plans

- The plan is only useful if
  - the environment does not change during execution in a way that affects the plan
  - the robot knows the state of the world and of the plan it is in at all times
  - the robot's effectors are accurate enough to execute each step of the plan

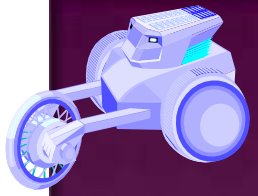
Fate of Deliberative Systems

- Due to these challenges, purely deliberative architectures are no longer used for the majority of physical robots
- Robot surgery is one system that can use deliberative systems because it requires a great deal of advance planning and no time pressure and the environment (patient's body) is kept perfectly static
- Deliberative control is also used for AI in chess systems





Reactive Control (Ch. 13)

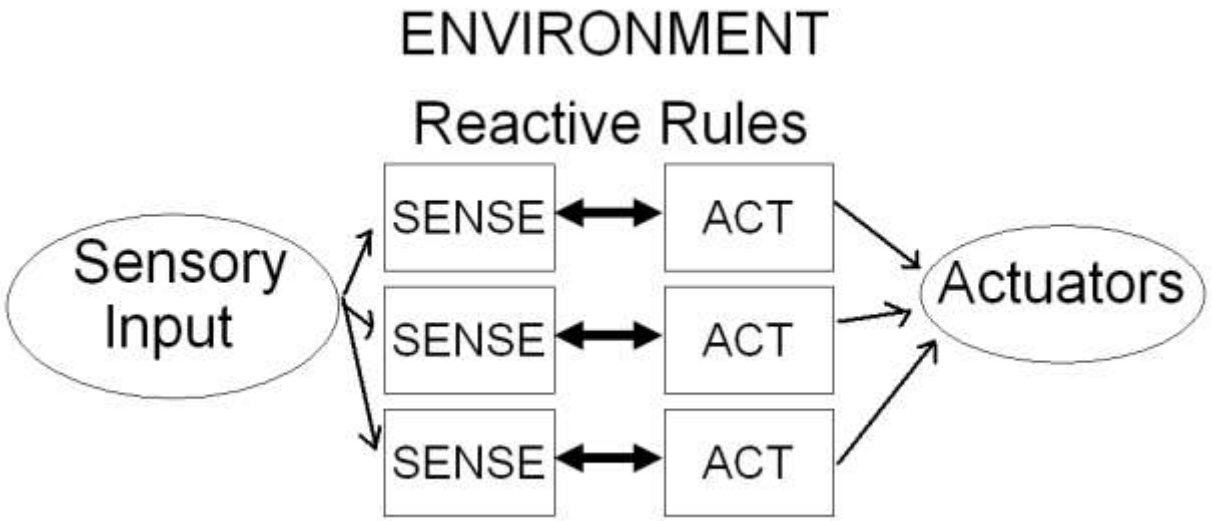


Reactive Control

- Reactive Control is one of the mostly commonly used methods for robot control and is based on a tight connection between the robot's sensors and effectors
- They do not use any internal representations and do not look ahead at the possible outcomes of their actions
- They operate on a short time scale and react to the current sensory information
- They have reactive rules (i.e. reflexes) to specific sensory input
- Complex computation is removed entirely in favor of fast, stored precomputed responses

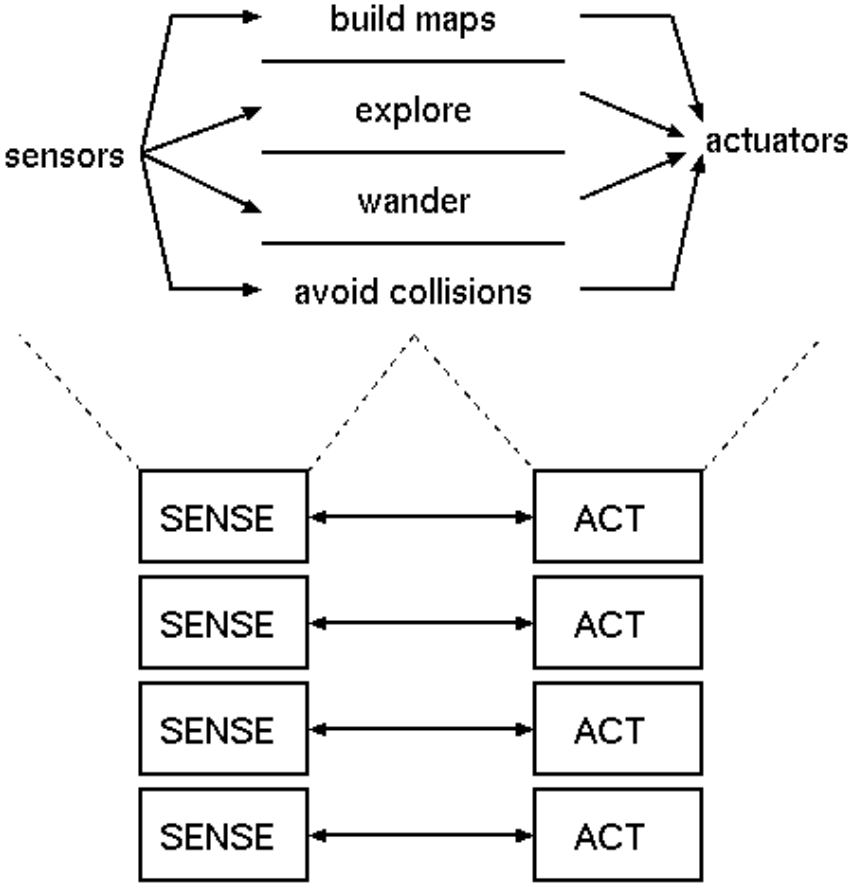
Stimuli and Behaviors

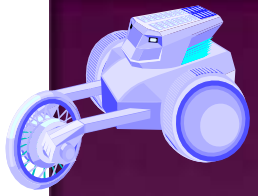
- The robot system has a set of situations (stimuli or coordinations) and a set of actions (responses, actions, behaviors)
- The situations may be based on sensory inputs or on internal state
- Examples are obstacle avoidance or random wander



Vertical Decomposition

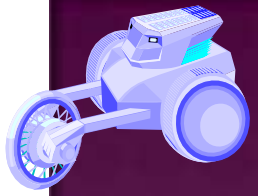
Biological systems are more vertical





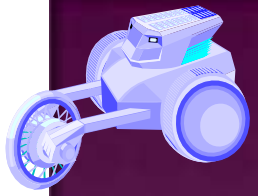
Mutually Exclusive Conditions

- To keep a reactive system simple have one unique behavior for each stimuli. The conditions are *mutually exclusive*.
- As the sensor state space grows the space may become unwieldy or intractable
- Coming up with the complete set of rules for the state space is typically done at design time not run time
- Typically there are only rules for important events and a default response for all others



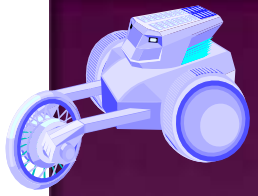
Stuck Situations

- A reactive controller may be get stuck if there is a default rule that covers some of the states. In wall following this may be resolved by the following:
  - Introduce randomness to get the robot unstuck from a corner by having ti turn by a random angle instead of a fixed one
  - Keep a history and remember the direction the robot turned last and use that information to make the decision about the turn direction



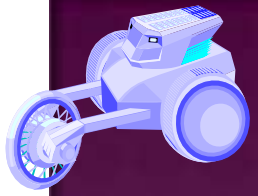
Action Selection

- *Action Selection* is the process of deciding among multiple possible behaviors when they are not mutually exclusive
- *Command arbitration* is the process of selecting one behavior from multiple candidates
- *Command fusion* is the process of combining multiple candidate behaviors into a single output behavior for the robot
- Reactive systems must support parallelism and the program must be able to multitask



Two Main Reactive Systems

- Subsumption Architectures
  - Layers of behaviors
  - Control relationships
- Potential fields
  - Concurrent behaviors
  - How to navigate
- They are equivalent in power
- Sometimes there is a mixture of the layers and concurrency

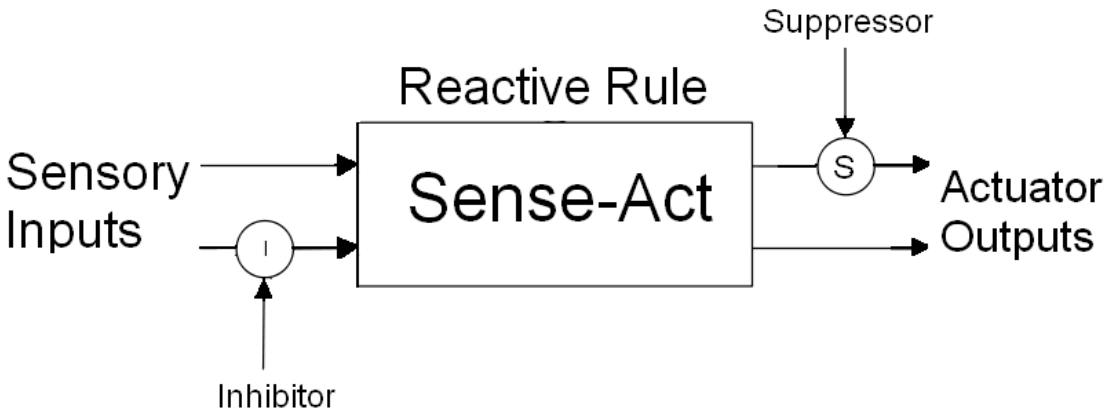


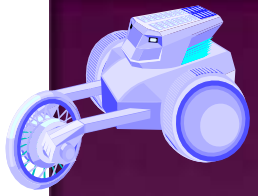
Subsumption Architecture

- *Subsumption Architecture (SA)* is a form of reactive control developed by Rodney Brooks in 1985
- SA builds systems incrementally from simple parts to more complex parts
- The complex parts use the simple existing components as much as possible
- The layers are added from the bottom up with the bottom layer being 0 and the most simple

Inhibition and Suppression

- The inputs of a layer/modules may be *inhibited* so that it receives no sensory inputs. It will compute no reaction and send no output to effectors or other modules
- Substitutes input going to a module
- The output of a layer/modules may be *suppressed* so that it receives sensory inputs but performs no computation and cannot control any effectors or other modules.
- Turns off output from a module



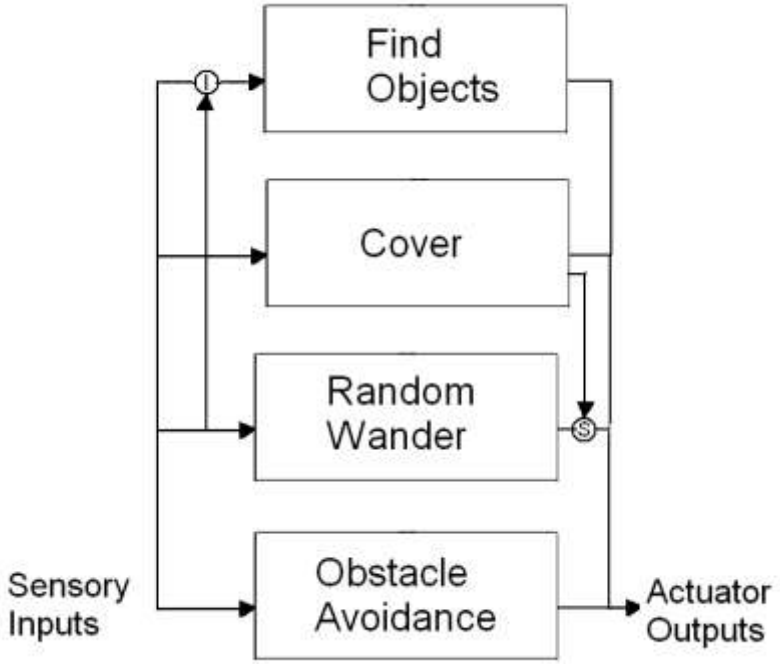


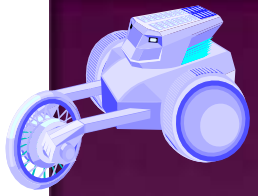
Subsumption Architecture

- Higher layers can assume the existence of lower ones and the goal they are achieving or use the lower ones to achieve their own goals
- Higher layers can subsume lower ones by using them while they are running or by suppressing them selectively
- The controller is bottom up because it progresses from simpler to more complex as layers are added incrementally
- The architectures should be taskable accomplished by a higher level turning lower levels on and off

Subsumption-based robot control system

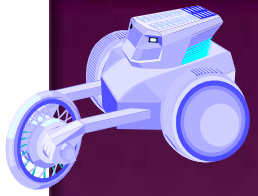
- The goal is to have very few connections between different layers
- The only intended connections are inhibition and suppression
- There are strongly coupled connections within layers
- Loosely coupled connections between layers





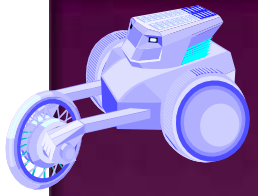
Representation and Summary

- In *Subsumption Architecture* the world is its own best model
- If the world can provide the information directly through sensing is best rather than to store it internally
- Components are task-achieving actions/behaviors
- Lowest layers handle the most basic tasks
- Higher layers exploit the existing ones



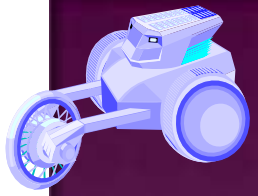
Limitations

- Minimal state if any
- No memory
- No learning
- No internal models/representations of the world



Schema Theory

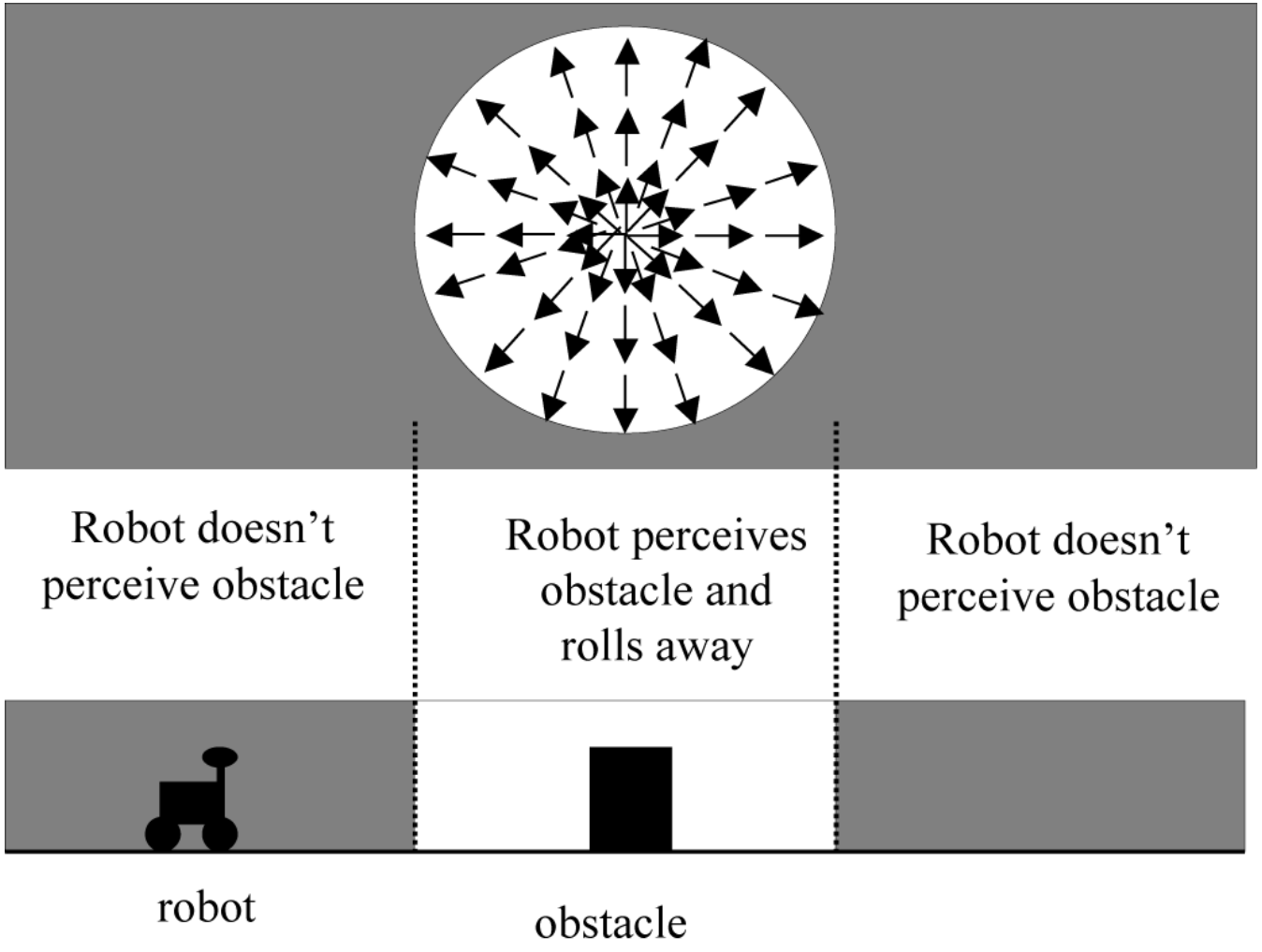
- Schema is used to express the basic unit of acitivity
- A schema consists both of the knowledge of how to act and /or perceive as well as the computational process by which it uses to accomplish the activity
- A schema class in C would contain both data and methods
- The *motor schema* represents the template for the physical activity
- The *perceptual schema* embodies the sensing



Potential Fields

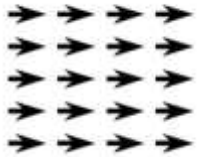
- The motor schema can be expressed with potential fields methodology
  - A potential field can be constructed from primitives summed together
  - The behavior output are combined using vector summation
- The robot feels a vector or force from each behavior
  - Magnitude
  - Direction
- Every point in space represents a force as a field that it would feel at that point

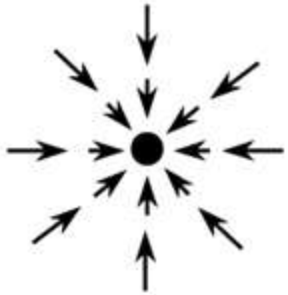
Robot Run Away Via Potential Field

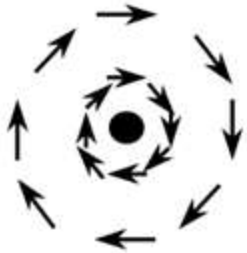


Primitive Potential Fields

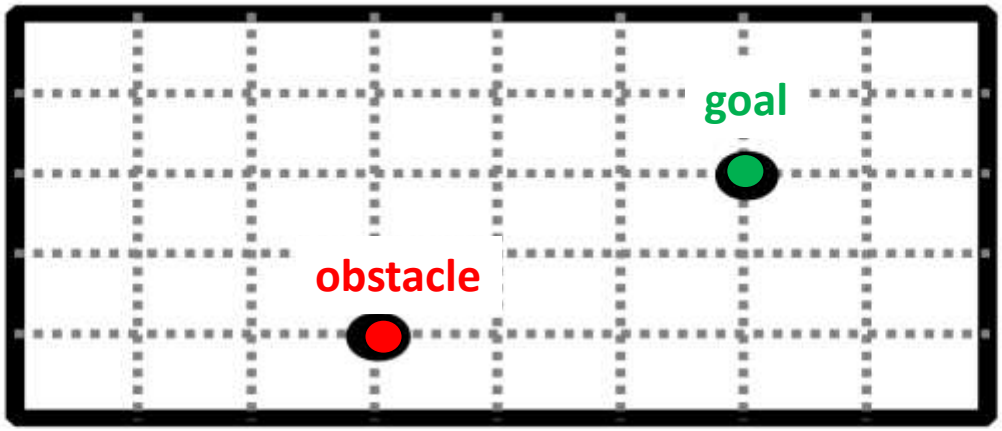
- Uniform
  - Move in a particular direction, wall following
- Repulsion
  - Run away, obstacle avoidance
- Attraction
  - Move to goal
- Tangential
  - Move through the door, docking
- Random
  - Potential field?

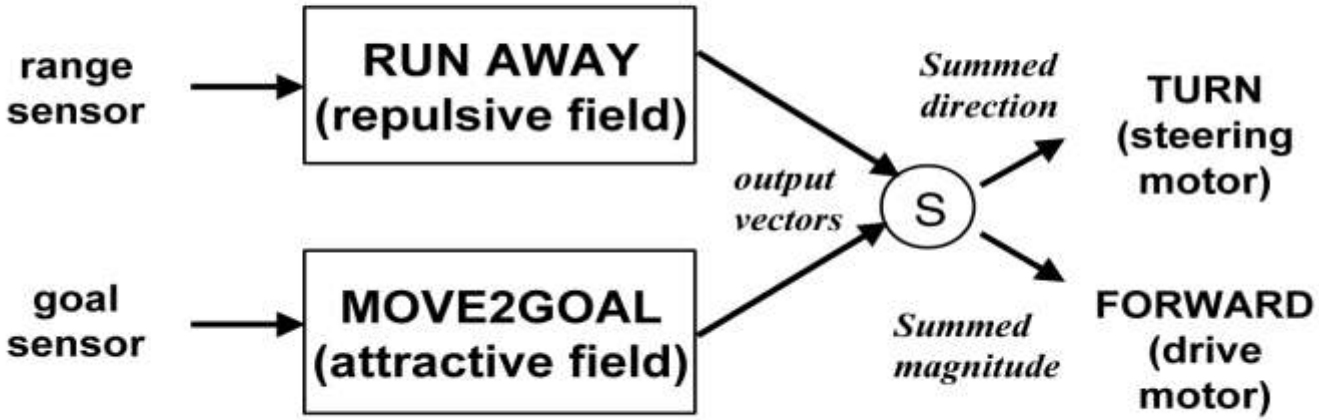




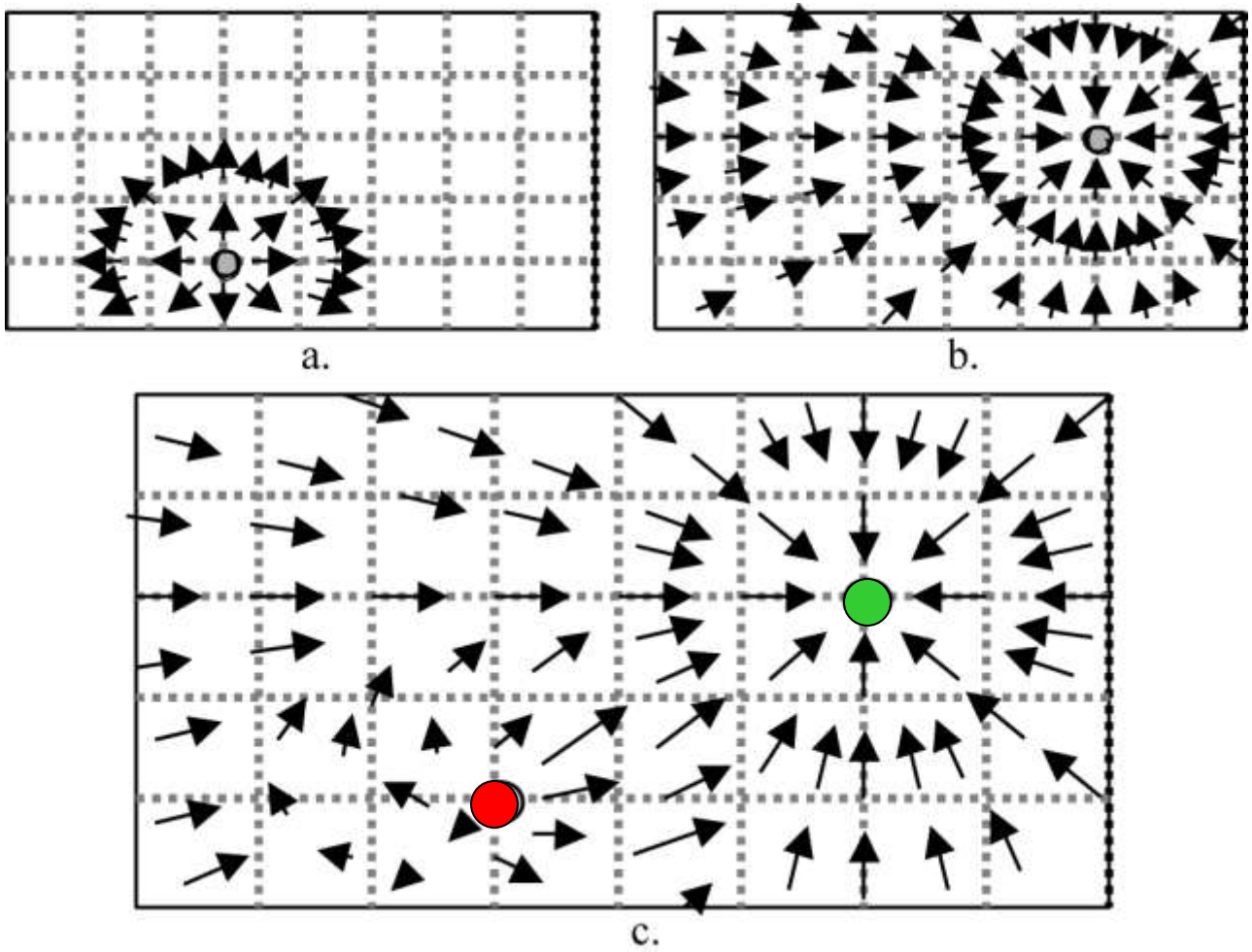


Combining Fields for Emergent Behavior



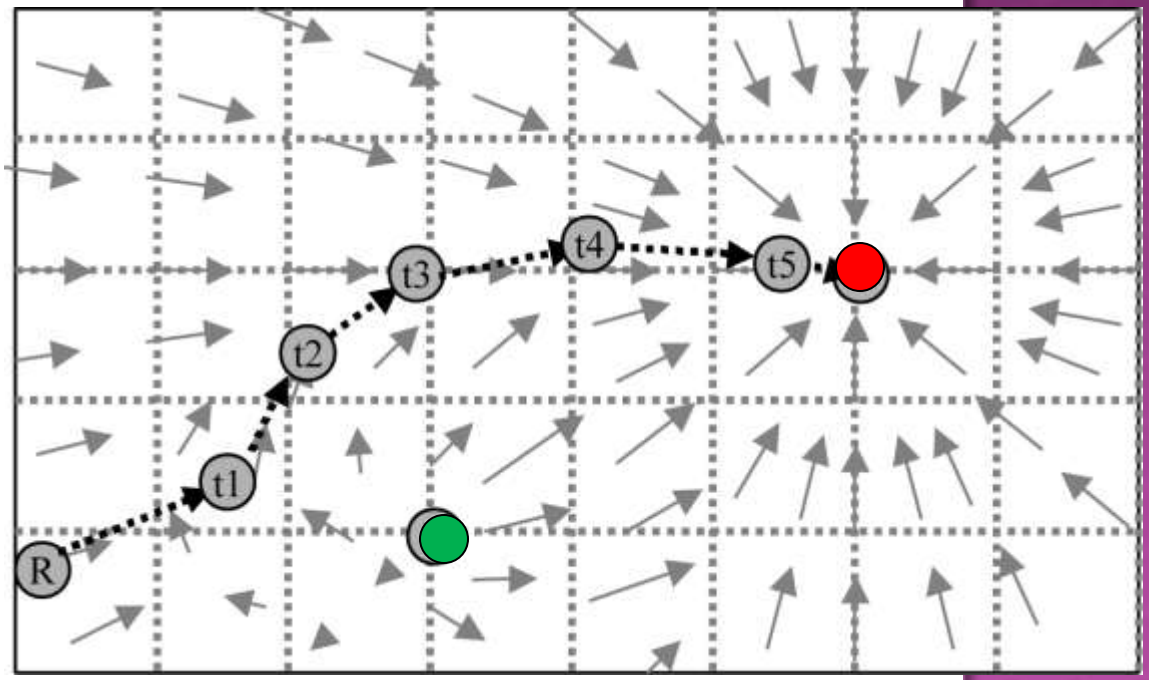


Attract and Replusive Fields

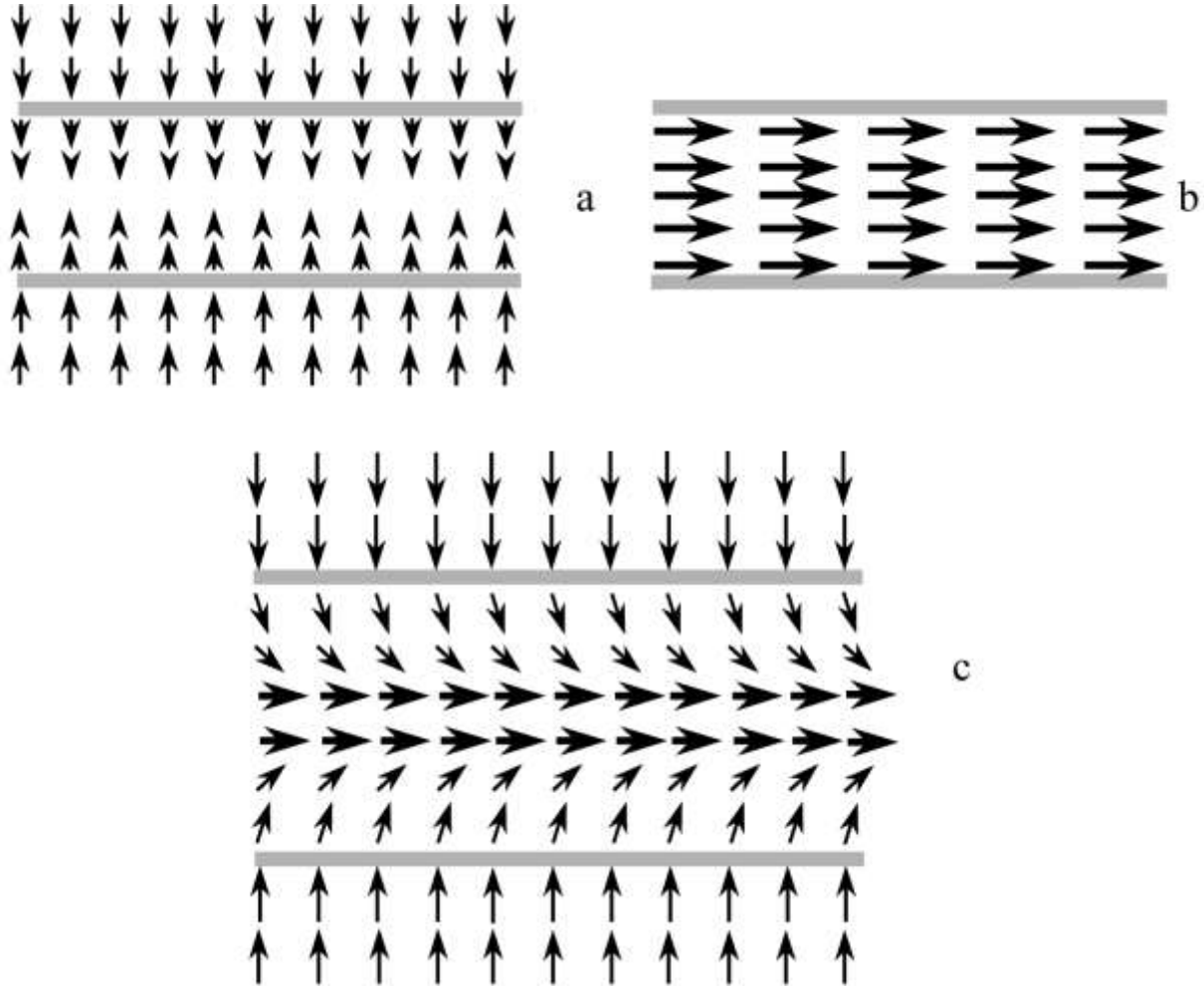


Robot's Path

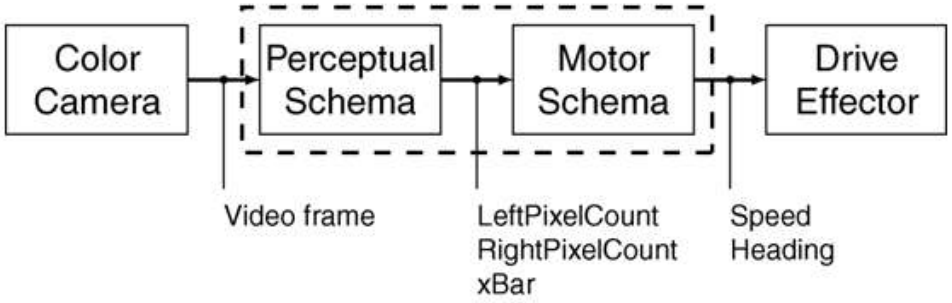
- If the robot starts at the location this is the path it takes
- The robot only feels the vector at the current location and then moves
- The robot never computes the field of vectors only the local vector

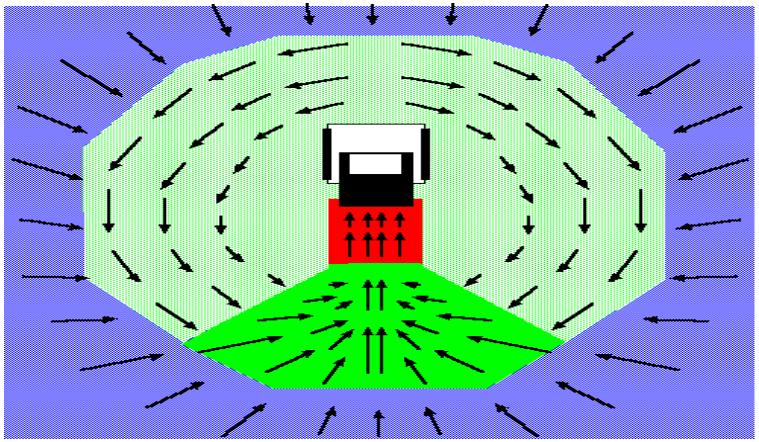


Follow Sidewalk Potential Fields

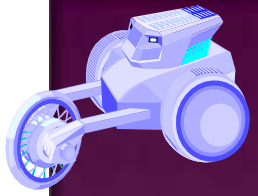


Docking using Potential Fields





•*Arkin and Murphy, 1990, Questa, Grossmann, Sandini, 1995, Tse and Luo, 1998, Vandorpe, Xu, Van Brussel, 1995. Roth, Schilling, 1998, Santos-Victor, Sandini, 1997*



Advantages and Disadvantages

- Advantages
  - Potential fields are easy to visualize
  - Easy to build up software libraries
  - Fields can be parameterized
  - Combination mechanisms are fixed and tweaked with gains
- Disadvantages
  - Local minima problem if it vectors sum to 0
  - Jerky motion