mc375: intro to robotics

intro to robot control topics.

- autonomy
- problem solving
- modeling
  - knowledge
  - representation
- control architectures
- deliberative control
- reactive control
- hybrid control

autonomy.

- to be truly autonomous, it is not enough for a system simply to establish direct numerical relations between sensor inputs and effector outputs
- a system must be able to accomplish *goals*
- a system must be able to *solve problems*

problem solving.

- the ability to achieve goals
- need to represent problem space
  - which contains goals
  - and intermediate states
- there is always a trade-off between *generality* and *efficiency*
- more specialized more efficient
- more generalized less efficient

problem solving:

example.

- GPS = General Problem Solver [Newell and Simon 1963]
- Means-Ends analysis

operator	*preconditions*	*results*
PUSH(obj,loc)	at(robot,obj)∧large(obj)∧ clear(obj)∧armempty()	at(obj,loc)∧ at(robot,loc)
CARRY(obj,loc)	at(robot,obj)∧small(obj)	at(obj,loc)∧ at(robot,loc)
WALK(loc)	none	at(robot,loc)
PICKUP(obj)	at(robot,obj)	holding(obj)
PUTDOWN(obj)	holding(obj)	¬holding(obj)
PLACE(obj1,obj2)	at(robot,obj2)∧holding(obj1)	on(obj1,obj2)

modeling.

- world modeling
- the way in which *domain knowledge* is embedded into a control system
- information about the environment stored internally; *internal representation*
- e.g., maze -- robot stores a “map” of the maze “in its head”

modeling:

knowledge.

- information in a context
- organized so it can be readily applied
- understanding, awareness or familiarity acquired through education or experience
- physical structures which have correlations with aspects of the environment and thus have a predictive power for the system

knowledge: philosophy.

- two branches of philosophy deal directly with knowledge
- *epistemology*
  - the study or theory of the nature of knowledge, especially with respect to its limits and validity
- *ontology*
  - a particular theory about the nature of being or the kinds of existents

knowledge: memory.

- divided into 2 categories according to duration
- long term memory (LTM)
  - persistent
- short term memory (STM)
  - transitory

knowledge: short-term memory.

- used as a buffer to store only recent sensory data
- data used by only one behavior
- examples:
  - *grids* : vary in resolution/area, shape, uniformity
  - *avoid-past* : avoid recently visited places to encourage exploration of novel areas
  - *wall-memory* : store past sensory readings to increase correctness of wall detection
  - *instantaneous obstacle map* : store detected obstacles projected onto the ground plane
  - *vector field histogram* : stores probabilistic sensor model in a form that is fast to update and use

knowledge: long-term memory.

- *metric maps* use absolute measurements and coordinate systems
- *qualitative maps* use landmarks and their relationships
- examples:
  - *a priori map representations* : utilize domain knowledge, many sources exist and can be combined; may contain errors and/or become outdated; frame of reference may be incompatible
  - *internalized plans* : pre-compile a map into a gradient vector field for a specific goal
  - *Markov models* : graph representation which can be augmented with probabilities for each action associated with each sensed state

knowledge: representation.

- must have a relationship to the environment
  - temporal (duration)
  - spatial
- must enable predictive power
  - looking ahead
  - but if inaccurate, it can deceive the system

knowledge: representation, 2.

- explicit
  - symbolic
  - discrete
  - manipulable
  - typical of traditional AI
- implicit
  - non-explicit
  - reconstructable
- tacit
  - embedded within the system
  - non-reconstructable

knowledge:
symbolic representations.

- require *symbolic grounding*
  - connecting the meaning (semantics) of an arbitrary symbol to the real world
- difficult because:
  - sensors provide signals, not symbols
  - symbols are often defined with other symbols (circular, recursive)
  - requires interaction with the world

representation:
environment.

- the world is
  - noisy
- states are
  - totally vs partially vs un- observable
  - discrete vs continuous
  - static vs dynamic
- other factors
  - speed of sensors
  - response time of effectors

representation:
components.

- spatial
  - metric or topological maps
- objects
  - instances of detectable things in the world
- actions
  - outcomes of specific actions on the self and the environment
- self/ego
  - stored proprioception (sensing internal state), self-limitations
- intentional
  - goals, intended actions, plans
- symbolic
  - abstract encoding of state/information

representation:
maps.

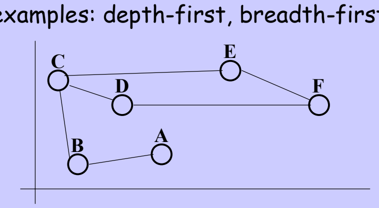
- euclidean map
  - represents each point in space according to its metric distance to all other points in the space
- topological map
  - represents locations and their connections, i.e., how/if they can be reached from one another; but does not contain exact metrics

representation:
state.

- there is a difference between *state* and *representation* !
- *state*
  - status of the system itself
- *representation*
  - arbitrary information that may be contained in the system

representation:
graphs.

- graph G = (V,E)
- many search models
  - examples: depth-first, breadth-first



representation: types of knowledge.

- spatial world knowledge
  - navigable surroundings and their structure
- object knowledge
  - categories or instances of things in the world
- perceptual knowledge
  - how to sense
- behavioral knowledge
  - how to (re)act
- ego knowledge
  - self-limits and capabilities
- intentional knowledge
  - goals

representation: Markov models.

- a Markov chain is a probability model for a multi-state system
- a probability is associated with each state transition in the system
- if you are currently in state **i**, then there is a probability p_{ij} that you will be in state **j** in the next time step

representation: Markov example.

current state	*next state* 1	2	3	4	5
1	0	.3	0	.5	.2
2	.5	0	0	.5	0
3	.4	0	0	.4	.2
4	1	0	0	0	0
5	0	0	.1	0	.9

representation: cognitive maps.

- term comes from animal navigation literature
- means of both previous experience storage and its use for action
- used by animals that forage and home
- may be simple collections of vectors
- support a wide range of behaviors
- based on strong biological evidence

representation: cognitive maps in rats.

- rats are extremely well adapted for navigation
- integrate various environmental cues (visual, auditory, scent, magnetic)
- populations of cells in the hippocampus encode specific places in the world
- cells are activated through movement

architecture.

- same usage as in *computer architecture*
- set of principles for designing computers out of well-understood building blocks

architecture: overview, 1.

- a *control architecture* provides a set of principles for organizing a control system
  - provides structure
  - provides constraints
- refers to *software control* level, not hardware!

architecture: overview, 2.

- implemented in a programming language
- a Turing-universal language
  - sequencing
  - conditional branching
  - iteration
- theoretically, any language could be implemented on any architecture

architecture: overview, 3.

- don't confuse "programming language" with "robot architecture"
- research has shown that even newly invented "architectures" continually fall into one of these four classes
- architectures guide how programs are structured

architecture: overview, 4.

- no architecture can compute more or less than any other
- since they are all implemented in Turing-universal programming languages
- and all Turing-universal languages are Turing equivalent

architecture: overview, 5.

- usually just by watching a robot in action, you cannot tell which control architecture is being used
- just as you cannot tell which language was used to write a working program
- but this is less true with very complex robot programs

architecture: classes of robot control architectures.

- deliberative
  - look-ahead; think, plan, then act
- reactive
  - no look-ahead; react!
- hybrid
  - think slowly, react quickly
- behavior-based
  - distribute thinking over acting

architectures: classes, 2.

- *time scale:* good way to distinguish control architectures
- reactive
  - respond to real-time requirements of environment
- deliberative
  - plan, so work on a longer time scale
- hybrid
  - combine the two time scales, generally through a middle layer, so also called three-layer architectures
- behavior-based
  - bring the time scales together by distributing computation over concurrent behavior models

architecture: criteria.

- support for parallelism
  - the ability of the architecture to execute parallel processes/behaviors at the same time
- hardware targetability
  - how well the architecture can be mapped onto real-robot sensors and effectors
  - how well the computation can be mapped onto real processing elements (i.e., microprocessors)

architecture: criteria, 2.

- run-time flexibility
  - does the architecture allow run-time adjustment and reconfiguration?
- modularity
  - how does the architecture address encapsulation of control?
  - how does it treat abstraction?
  - does it allow many levels?
    - e.g., feedback loops, primitives, agents
  - does it allow re-use of software?

architecture: criteria, 3.

- niche targetability
  - how well does the architecture allow the robot to deal with its environment?
- robustness
  - how well does the architecture perform if individual components fail?
  - how well does it enable and facilitate writing controllers capable of *fault tolerance*?

architecture: criteria, 4.

- ease of use
  - how easy to use and accessible is the architecture?
  - are there programming tools and expertise?
- performance
  - how well does the robot perform using the architecture?
  - does it act in real-time?
  - does it get the job done?
  - is it failure-prone?

architecture: representation.

- strong relationship between class of control architecture and representation methodology used
- time to build
- time to use

architecture:
representation, 2.

- role varies in different architectures:
  - deliberative: extensive
  - reactive: have none
  - hybrid: use it
  - behavior-based: avoid it or distribute it

architecture:
topics.

- today:
  - deliberative
  - reactive
  - hybrid
- after break:
  - behavior-based

deliberative control.

- classical control architecture (first to be tried)
- first used in AI to reason about actions in non-physical domains (like chess)
- natural to use this in robotics at first

deliberative control:
Shakey.

- 1960's at SRI (Stanford Research Institute)
- state-of-the-art machine vision used to process visual information
- used classical planner (STRIPS)

deliberative control:
planning.

- looking ahead, searching for what to do next
- the goal is a state
- entire state space is enumerated and searched, from current state to goal state
  - different paths are tried
  - optimal path is the one we want to use

deliberative control:
SPA.

- planner-based architecture
- involves 3 steps:
  (1) sensing (S)
  (2) planning (P)
  (3) acting (A)
- SPA has serious drawbacks...

deliberative control:

SPA problem 1: time scale.

- it may (probably) take a very long time to search robot's state space
- real robots may have input from multiple sensors
- hard to enumerate all possible states
- there's too much information!
- generating a plan is slow

deliberative control:

SPA problem 2: space.

- a lot of memory is needed to store/search robot's large state space
- representation must be robust to store all the information properly
- generated plan can be large
- however this is less of a problem than time

deliberative control:

SPA problem 3: information.

- planner assumes representation is accurate and up-to-date
- not necessarily true!
- representation must be constantly updated and checked for accuracy, consistencies
- too little information!
- and/or inaccurate information!

deliberative control:

SPA problem 4: use of plans.

- resulting plan is only useful if
  - environment does not change during planning
  - environment does not change during execution in such a way as to invalidate the plan
  - representation was accurate enough that plan will actually work
  - robot's effectors are accurate enough so that predicted and actual action results are the same

deliberative control:

summary of problems.

- require search and planning, which are slow
- encourage open-loop execution, which is limiting and dangerous
- NOTE: if planning were not slow, then execution could be closed-loop since re-planning could occur based on feedback

deliberative control:

after deliberation.

- real robot practitioners objected strongly to SPA
- in early/mid 1980's alternatives were proposed:
  - reactive systems
  - hybrid systems

deliberative control:

role of deliberation.

- deliberative architectures no longer used on real robots, after "revolution" in mid 1980's
- however, deliberation is still used in other areas of AI, such as chess and other static domains

deliberative control:

expansion.

- in robotics, SPA has been expanded to overcome previous issues:
  - since search/planning is slow, save/cache important and/or urgent decisions
  - since open-loop execution is bad, use closed-loop feedback and be ready to respond or re-plan when a plan fails

reactive control.

- operate on a short time scale
- does not look ahead

reactive control:

overview, 1.

- reactive control is based on a tight loop connecting the robot's sensors with its effectors
- purely reactive controllers do not use any internal representation; they merely react to the current sensory information
- use a direct mapping between sensor and effectors; minimal state information (if any)

reactive control:

overview, 2.

- collection of rules that map situations to actions
- simplest form:
  - divides perceptual world into a set of mutually exclusive situations
  - recognize which situation we are in
  - react to it

reactive control:

overview, 3.

- usually too hard to define mutually exclusive situations
  - what if multiple sensors are involved?
  - robot's entire sensory space could be very large!

reactive control:
overview, 4.

- mapping from sensory input to actions is done during system design time, not at run-time
- often humans can filter/shrink the entire sensory space

reactive control:
arbitration.

- deciding between two or more different possible actions or behaviors
- can be done based on:
  - fixed priority hierarchy
  - dynamic hierarchy
  - learning

reactive control:
universal plans.

- suppose all possible plans for all possible actions can be generated in advance
- and an optimal reaction for each situation can be identified
- this is a *universal plan*
- also called a *complete mapping*
- reactive. planning is done at compile-time, not run-time.
- but not viable, because:
  - world must be deterministic
  - world must not change
  - goals must not change
  - world is too complex (state space is too large)

reactive control:
situated automata.

- formal notion of finite state machines (FSM)
  - inputs connected to sensors
  - outputs connected to effectors
- *"situated"* = interacting with a complex world
- used to create reactive principled control systems

reactive control:
control with situated automata.

- two ways to construct
  - manually
    - e.g., subsumption architecture [Brooks 1986]
  - pre-compiling a complete plan
    - similar to universal plans, but in terms of FSM circuitry

reactive control:
subsumption architecture.

- best known reactive control architecture
- Rod Brooks, MIT, 1985
- principles:
  - systems are built from the bottom up
  - components are task achieving actions/behaviors (not functional modules)
  - components can be executed in parallel
  - components are organized in layers, from the bottom up
  - lowest layers handle most basic tasks
  - newly added components and layers exploit existing ones
  - each component provides and does not disrupt tight coupling between sensing and action
  - no internal models ("the world is its own best model")

hybrid control.

- basic idea:
  - use the best of both worlds (deliberative and reactive)
  - combine open-loop and closed-loop execution
  - combine different time scales and representations

hybrid control:
organization.

- typically consists of three components:
  (1) reactive layer
  (2) planner
  (3) integration layer to combine (1) and (2)
- often called three-layer architectures
- planner and reactive layers are standard

hybrid control:
the magic middle.

- middle / integration layer has to:
  - compensate for limitations of both planning and reactive layers
  - reconcile different time scales of the other two layers
  - reconcile different representations of the other two layers
  - reconcile any contradictory commands between the two

hybrid control:
re-using plans.

- some frequently useful planned decisions may need to be reused
- so to avoid planning, these can be stored (cached) and looked up in the middle layer
- examples:
  - intermediate-level actions (ILA's): stored in contingency tables
  - macro operators: (small) plans compiled into more general operators for future use

hybrid control:
dynamic re-planning.

- reaction can influence planning
- important changes discovered by low-level controller go back to planner; planner uses them to re-plan
- planner is interrupted when an answer is needed in real-time
- reactive controller stops, waits for new plan

hybrid control:
planner-driven reaction.

- planning can influence reaction
- important optimizations the planner discovers are passed down to the reactive controller
- planner's suggestions are used if safe and possible
- *who has priority: reactor or planner?*

hybrid control: strengths.

- deliberative planners
  - rely heavily on world models
  - can readily integrate world knowledge
  - have broader perspective and scope
- reactive and behavior-based systems
  - afford modular development
  - provide real-time robust performance in dynamic world
  - provide for incremental growth
  - tightly coupled to incoming sensory data

hybrid control: interaction of layers.

- interaction of layers
  - hierarchical integration
  - planning guides reaction
  - coupled planning and reacting
- types of interaction
  - selection: planning is viewed as configuration
  - advising: planning is viewed as advice giving
  - adaption: planning is viewed as adaption of controller
  - postponing: planning is viewed as least commitment process

hybrid control: examples.

- there are many, many examples
- review of these could be someone's term project :)

reading.

- Intro to Robotics (McKerrow), 2.9-2.11 (1st course pack)
- Robotic Explorations, ch 8 (web)

- 2nd course pack will be at bookstore this week (hopefully!)
- reading for spring break
- other readings on web (rest of semester)