COSC 4P78

Hybrid Control

Week 7

(Work based on Matarić)

Brock University

Last Time, on Robotics 4P78...

- Reactive control is fast, but inflexible
  - It's only good for the precise conditions the designer anticipated
- Deliberative control is powerful
  - But it doesn't like surprises, which is... kinda a problem

¿Por qué no los dos?

And now... the conclusion...

(dunn dunn!)

We create three layers:

1. Deliberative (Planner)
2. ???
3. Profit! (Sorry. Had to. It's actually Reactive)

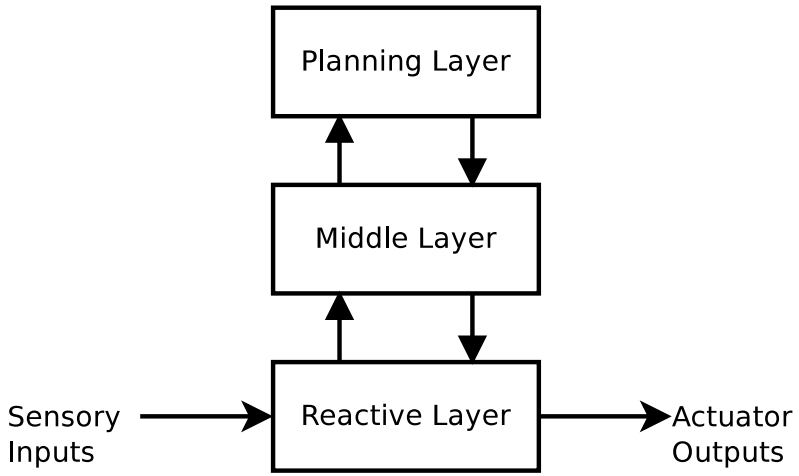
The question is, what do we put in that middle layer?

- Also, this should be obvious by this point in the course, but each of those layers will, themselves, also be composed of multiple layers or modules

The Missing Link

Our connecting layer has to:

- Reconcile differing time-scales
- Deal with different representations
- Reconcile conflicting commands to be sent to the robot
- Compensate for any other limitations of the other two layers



Case Study:

Delivering Medication in a Hospital

From a navigation standpoint, what does this entail?

- Avoiding fast-moving people and objects (gurneys, etc.)
- Efficiently finding paths to specific rooms (for both pickup and delivery)

So, elements of both, right?

How easy should it be to combine the two?

Case Study

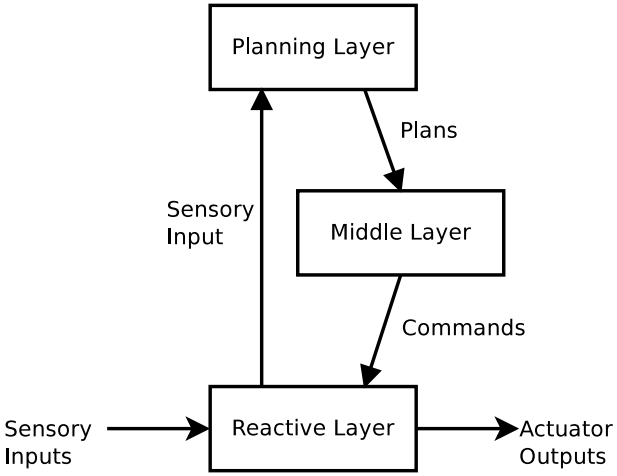
This is starting to sound hard...

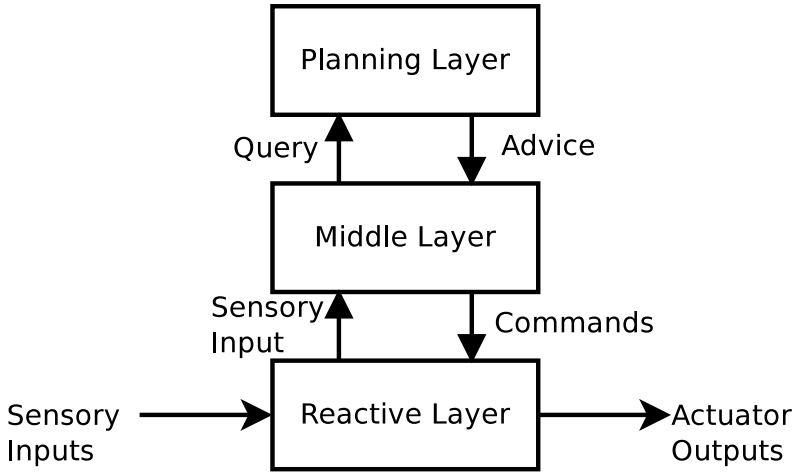
- The reactive system should, of course, always be ready to go, but how long should we wait for deliberative portion before we start moving? Should the robot just sit still while it's thinking (and someone might be bleeding out)? Should it start rolling, possibly in the wrong direction?
- What if it starts moving, but is temporarily (how could we even identify "temporarily"?) blocked? e.g. by doctors or a stretcher
  - Wait them out, or treat them like an impassable permanent wall?
- What if the map is out of date?
- What if certain destinations/routes are commonly repeated? Does it make sense to treat it like a fresh problem each time?

eh... this starting to sound tedious...

A "middle layer" may not necessarily *solve* all of these issues, but it'll likely have to at least *address* them!

Some Possible Configurations





Of course, there are many variations on these, as well as other options entirely

Coping With Change

If the reactive system can't do its job (obstacle, closed door, zombie clown invasion, etc.), it can notify the deliberative layer

- The planner can update its own world state (which is good)
- This eliminates the need to go through the same process if the task is repeated (or a similar task is required)
  - The deliberative layer can save computation cycles for where it's necessary, rather than wasting time on redoing the same work in the future
- This does present the problem of whether or not the robot will ever notice the obstruction being removed...

Dynamic Replanning

Since planning is expensive, we'd like to limit ourselves to doing it when actually necessary

- The reactive layer realizing a plan isn't navigable is a good time to notify the deliberative layer
- *Dynamic replanning* is when the deliberative layer replans as a response to discovering the current plan won't work
  - Otherwise, with a plan in place, it might not clock cycles without good reason

Action or Deliberation?

If a robot starts moving for the sake of not just sitting still, the deliberative layer might discover a new, better path

- If this happens, it might send a signal to the reactive layer to stop what it's doing and take the new path
- It might, for example, start sending it to the basic direction of where it needs to go — e.g. to J-Block at Brock — before working out a complete path — e.g. take the top floor because there are fewer obstructions
- Potentially, the differences in timing could make this a bit unwieldy — e.g. if you were starting in D-Block, you wind up already in J-Block before it had the bright idea of going upstairs
  - Better yet, imagine trying to get from an office in J-Block to H310!
- (In practice, it would be reasonable for the deliberative layer to actually do this *through* the middle, managing layer)

Reusing Your Work

Avoiding Replanning

Every plan the deliberative layer comes up with is unique to a *specific* task (i.e. set of initial state, goal state, and parameters)

- But what if the same problem (or *sub-problem*!) might come up more than once?
  - It might be useful to stash solutions for future use
  - Reusing old plans can potentially be faster
  - Specifically, reusing *portions* of plans can be very handy
    - ★ Consider the case where part of a path might be blocked by a temporary obstacle (e.g. students waiting to get into a seminar room in the MC hallways)
    - ★ If it isn't really a planning task, we might not want to use the deliberative layer for it
    - ★ It's still more involved (with more steps) than appropriate for reactive
    - ★ So, it's somewhat in the *middle*... (hint hint)

Situational Miniplans

(No, that isn't really a term)

So, we're saying the managing layer might retain resolutions to commonly-occuring snags?

- What we're talking about is a *contingency table*
- A direct lookup table
  - Inputs mapped to outputs — like a reactive layer
  - Produces a partial plan to be enacted by a navigator or reactive layer — like a deliberative layer
- Fifty-cent phrase: *macro operators*
  - (We're still talking about the same thing; some people just like sounding fancy)

Who's the Boss?

So, which layer's actually in control?

- It depends?
- Seriously, it depends on which layer makes most sense for the task and approach
  - It could be hierarchical
    - ★ BTW have you noticed by now that that term isn't *quite* the same thing as deliberative? They just tend to be used together
  - The deliberative layer could always be in charge
    - ★ With the reactive layer simply finding a way to enact the received instructions
- Theoretically, it would usually be most effective and efficient for the answer to be *both*
  - The deliberative layer could interrupt the reactive layer if it found a new, better path
  - The reactive layer could preempt the current plan if the route is blocked or new vital information is discovered (e.g. finding a survivor in a burning building sooner than anticipated)
  - This sort of coupling can be hard to actually effect

Planning

On-line? Off-line?

Remember we said that we can — potentially — store multiple plans for later use, whether complete or partial

- Suppose we were dealing with an environment like Brock
  - A finite space
  - Known floor plans
  - Well-defined pathways for traversal
  - Would it be *possible* to devise plans between all major destinations on-campus?
    - ★ If so, and if this is the expensive part, why would it have to be done by the robot while it was on the job?
    - ★ Why not just precompute them on a more powerful computer?
    - ★ *If* we could create a set of *all* possible plans, that would be a *Universal Plan*
    - ★ Note that we're talking about a *closed world* here!
  - If we were to take it to this extreme, we'd really be talking about a lookup table that mapped all possible inputs to a specific output
    - ★ Doesn't that sound familiar?

Situated Automata

(What we're really talking about here...)

- This approach was attempted by taking all of the available *domain knowledge* — distilled facts about the robot, task, and environment — integrating it into a special declarative language, and created "virtual circuits" to map that knowledge directly to navigation
  - These *situated automata* were mathematical models to *fully* describe a possible state space
  - Of course:
    - ★ Representing, effectively, a multiverse isn't *actually* feasible, memory-wise
    - ★ A single change to the environment would require recomputing said multiverse. From scratch.
    - ★ Hopefully nobody will ever want to define a new goal, because that's the same problem as trying to change the environment (i.e. saying, "I'd like to go get pizza instead" is tantamount to redefining the universe. of universes.)
  - (Personal opinion: They found a way to re-invent STRIPS, except even less practical)

(Mini-conclusion)

So, basically, we're stuck "having to do deliberation and reaction in real time". The *hybrid* model is a good way to do that.

- Ostensibly, this is the correct lesson to glean from the previous thought exercise, but that doesn't negate the potential value of pre-planning
  - Making use of, for example, common bottlenecks in pedestrian traffic — especially at specific times — would be entirely reasonable for navigating through Brock
  - Just don't think that preplanning will be a general *replacement* for the deliberative layer

Actual conclusion

So then, we're on slide 17, and we still haven't very clearly defined what **the** hybrid model really is, have we?

- Because, for all intents and purposes, there isn't one
  - Which isn't to say there *isn't* one, but that there isn't *one*
- Actual robots can have any number of possible controller architectures, with discrete modules operating on fundamentally different levels of abstraction and planning, and running on different time-scales
  - One could argue that any of them are 'hybrid'

My best advice is to focus on approach more than labels. Break down the actual problem into parts, and then look at what tools you need to address each of them. Worry about coordinating everything later.

One Final Note

Murphy's *Introduction to AI Robotics* takes a significantly different approach to this topic

- She focuses more on abstracting the principle, and on *specific* implementations of the paradigm
  - It's a really interesting read; I simply didn't want to focus quite so much on *specific* solutions
- Two of the more interesting points of note in the chapter were:
  - Still using three layers, but organizing them as a deliberative planner, a reactive skill manager, and a sequencer that qualified as a combination of both deliberative and reactive
    - ★ The planner included the cartographer, and a hierachical breakdown of goals into subgoals into tasks
    - ★ The skill manager consisted of specific skills and events, for immediate perceptions and actuation
    - ★ The sequencer prioritized and managed tasks, and translated tasks into commands for completion via skills
  - The possibility of managing tasks and resources in a similar fashion to how an Operating System's scheduler does

One Finaler Note

(I ran out of space)

We still haven't discussed much that we can directly *use*, applicable to real tasks, right?

First, this is a neat: `http://www.cs.cmu.edu/~TCA/tca.orig.html`
Related: `http://www.cs.cmu.edu/~TCA/`
(Seriously, the book's in the library; it's really interesting!)

But, besides that, consider how we'd likely combine all of this into a real robot... (next slide)

Practical Approach

- Rather than lookup tables, or inventing a new declarative language, we'd use an off-the-shelf one, like Prolog or SQL
  - Create a database of facts, and query for solutions
  - Failing that, whenever a new solution is found, store it in a queryable hash table
- Have a 'middle' layer that acts as a client to the deliberative layer (the knowledge base and planner), and coordinates the lower-level functions
- All three systems would be live simultaneously, and possibly even spread across more than one processor
- Use message-passing (either Inter-Process Communication, or over a communication bus like serial or I^2C to communicate

For a look at how you might make a *modern* robot, look at the Robot Operating System (ROS), which would facilitate much of this, and is readily extensible.

Thought Experiment Time!

This has nothing to do with the topic at hand, but before we either move on to the next topic or finish for the evening, let's try discussing a possible task for a robot.

You want to explore a body of water. Maybe to look for salvage, maybe to map out its geometry. We can decide on a specific task together.

What sorts of design considerations would you need to account for in terms of actuators/effectors? Sensors? Communication? Anything else?

What other sorts of information would we need before answering those questions? e.g.:

- Freshwater or salt?
  - Does it even matter?
- Enclosed like a small lake, or just a portion of a larger space?
- What sort of plant life is present? What about fauna?
- Is particulate obfuscation a concern?