

Lecture slides for
Automated Planning: Theory and Practice

Chapter 20

Planning in Robotics

Dana S. Nau
University of Maryland

5:34 PM January 24, 2012

What is a Robot?

- A machine to perform tasks
 - ◆ Some level of autonomy and flexibility, in some type of environment
 - ◆ Sensory-motor functions
 - » Locomotion on wheels, legs, or wings
 - » Manipulation with mechanical arms, grippers, and hands
 - ◆ Communication and information-processing capabilities
 - » Localization with odometers, sonars, lasers, inertial sensors, GPS, etc.
 - » Scene analysis and environment modeling with a stereovision system on a pan-and-tilt platform

Examples of Tasks and Environments

- Manufacturing tasks
 - ◆ painting, welding, loading/unloading a machine tool, assembling parts
- Servicing stores, warehouses, and factories
 - ◆ maintaining, surveying, cleaning, transporting objects.
- Exploring unknown natural areas, e.g., planetary exploration
 - ◆ building a map with characterized landmarks, extracting samples, setting various measurement devices.
- Assisting people in offices, public areas, and homes.
- Helping in tele-operated surgical operations

Status

- Reasonably mature technology when robots restricted to either
 - ◆ well-known, well-engineered environments
 - » e.g., manufacturing robotics
 - ◆ performing single simple tasks
 - » e.g., vacuum cleaning or lawn mowing
- For more diverse tasks and open-ended environments, robotics remains a very active research field

Robots without Planning Capabilities

- Requires hand-coding the environment model and the robot's skills and strategies into a reactive controller
- The hand-coding needs to be inexpensive and reliable enough for the application at hand
 - ◆ well-structured, stable environment
 - ◆ robot's tasks are restricted in scope and diversity
 - ◆ only a limited human-robot interaction
- Developing the reactive controller
 - ◆ Devices to memorize motion of a pantomime
 - ◆ Graphical programming interfaces

Requirements for Planning in Robotics

- online input from sensors and communication channels
- heterogeneous partial models of the environment and of the robot
- noisy and partial knowledge of the state from information acquired through sensors and communication channels
- direct integration of planning with acting, sensing, and learning

Types of Planning

- Domain-independent planning is not widely used in robotics
 - ◆ Classical planning framework too restrictive
- Instead, several specialized types of planning
 - ◆ Path and motion planning
 - » Computational geometry and probabilistic algorithms
 - » Mature; deployed in areas such as CAD and computer animation
 - ◆ Perception planning
 - » Younger, much more open area
 - ◆ Navigation planning
 - ◆ Manipulation planning

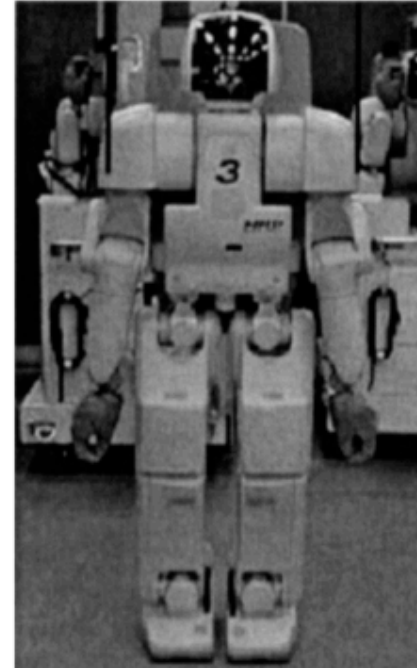
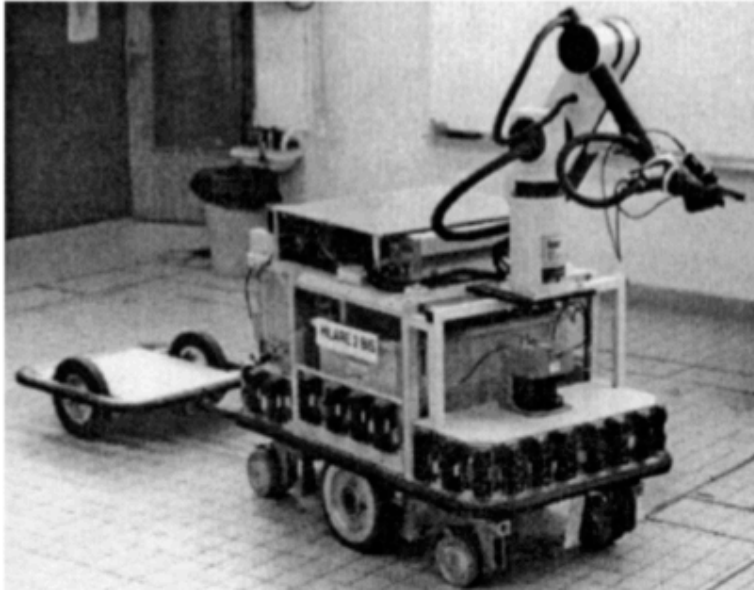
Path and Motion Planning

- Path planning:
 - ◆ Find a feasible geometric path for moving a mobile system from a starting position to a goal position
 - ◆ Given a geometric CAD model of the environment with the obstacles and the free space
 - ◆ A path is feasible if it meets the kinematic constraints of the mobile system and avoids collision with obstacles
- Motion planning:
 - ◆ Find a feasible trajectory in space and time
 - » feasible path and a control law along that path that meets the mobile system's dynamic constraints (speed and acceleration)
 - » Relies on path planning

Configuration Parameters

- Car-like robot
 - ◆ Three configuration parameters are needed to characterize its position: x, y, θ
 - » Path planning defines a path in this space
 - ◆ The parameters are not independent
 - » E.g., unless the robot can turn in one place, changing θ requires changing x and y
- Mechanical arm with n rotational joints
 - ◆ n configuration parameters
 - » Each gives the amount of rotation for one of the joints
 - ◆ Hence, n -dimensional space
 - ◆ Also, min/max rotational constraints for each joint

Examples

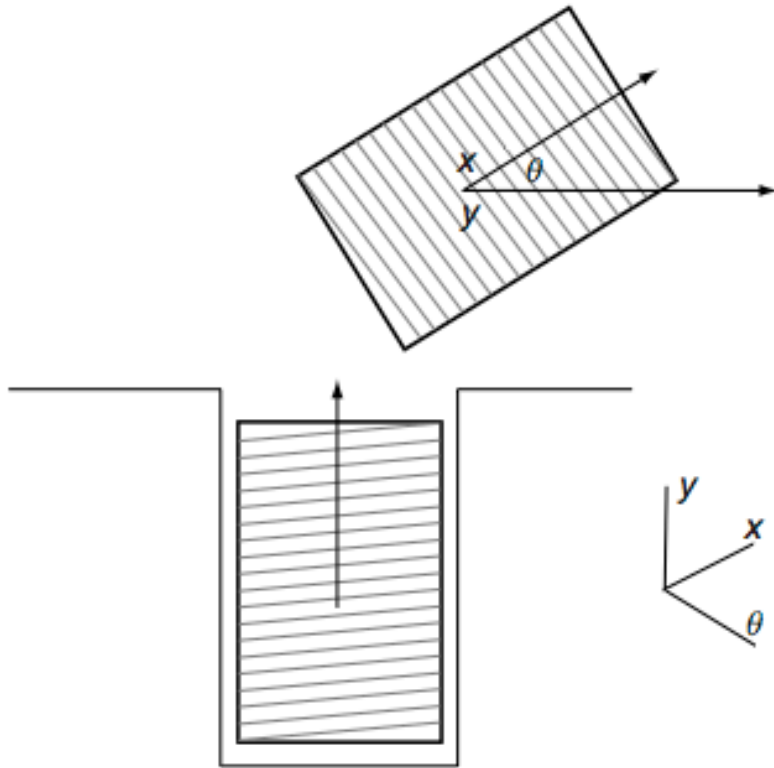


- The robot *Hilare*
- 10 configuration parameters:
 - ◆ 6 for arm
 - ◆ 4 for platform & trailer
- 52 configuration parameters
 - ◆ 2 for the head,
 - ◆ 7 for each arm
 - ◆ 6 for each leg
 - ◆ 12 for each hand

Path Planning

- Definitions
 - ◆ q = the configuration of the robot = an n -tuple of reals
 - ◆ CS = the configuration space of the robot
= {all possible values for q }
 - ◆ CS_{free} = the free configuration space
 - » {configurations in CS that don't collide with the obstacles}
- Path planning is the problem of finding a path in CS_{free} between an initial configuration q_i and a final configuration q_g
- Very efficient probabilistic techniques to solve path planning problems
 - ◆ Kinematic steering finds a path between two configurations q and q' that meets the kinematic constraints, ignoring the obstacles
 - ◆ Collision checking checks whether a configuration or path between two configurations is collision-free (i.e., entirely in CS_{free})

- Explicit definition of CS_{free} is computationally difficult
 - ◆ Exponential in the dimension of CS



Car-like robot and environment



Configuration space

Roadmaps

- Let $L(q, q')$ be the path in CS computed by the kinematic steering algorithm
- A *roadmap* for CS_{free} is any finite graph R whose vertices are configurations in CS_{free}
 - ◆ two vertices q and q' in R are adjacent in only if $L(q, q')$ is in CS_{free}
- Note:
 - ◆ Every pair of adjacent vertices in R is connected by a path in CS_{free}
 - ◆ The converse is not necessarily true

Planning with Roadmaps

- Given an adequate roadmap for CS_{free} and two configurations q_i and q_g in CS_{free} , a feasible path from q_i to q_g can be found as follows:
 - ◆ Find configuration q'_i in R such that $L(q_i, q'_i)$ is in CS_{free}
 - ◆ Find configuration q'_g in R such that $L(q_g, q'_g)$ is in CS_{free}
 - ◆ In R , find a sequence of adjacent configurations from q'_i to q'_g
- The planned path is the finite sequence of subpaths
$$L(q_i, q'_i), \dots, L(q_g, q'_g)$$
 - ◆ Postprocessing to optimize and smooth the path
- This reduces path planning to a simple graph-search problem, plus collision checking and kinematic steering
 - ◆ How to find an adequate roadmap?

Coverage

- Need to find a roadmap that *covers* CS_{free}
 - ◆ Whenever there is a path in CS_{free} between two configurations, there is also a path in the roadmap
 - ◆ Easier to use probabilistic techniques than to compute CS_{free} explicitly
- The *coverage domain* of a configuration q is
 - ◆ $D(q) = \{q' \in CS_{free} \mid L(q, q') \subseteq CS_{free}\}$
- A set of configurations $Q = \{q_1, q_2, \dots, q_n\}$ *covers* CS_{free} if
 - ◆ $D(q_1) \cup D(q_2) \cup \dots \cup D(q_n) = CS_{free}$

Probabilistic Roadmap Algorithm

- Probabilistic-Roadmap

- ◆ Start with an empty roadmap R
- ◆ Until (termination condition), do
 - » Randomly generate a configuration q in CS_{free}
 - » Add q to R iff either
 - q extends the coverage of R
 - e.g., there's no configuration q' in R such that $D(q')$ includes q
 - q extends the connectivity of R
 - i.e., q connects two configurations in R that aren't already connected in R

Termination Condition

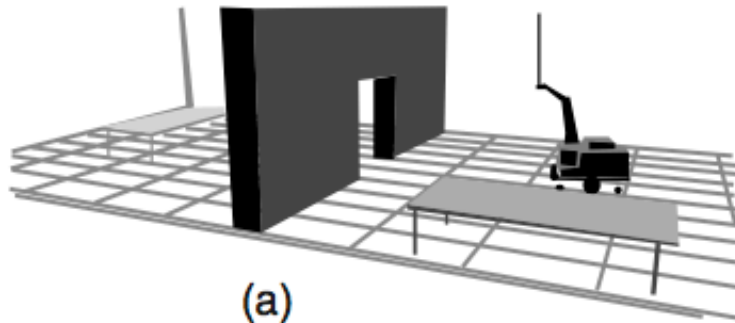
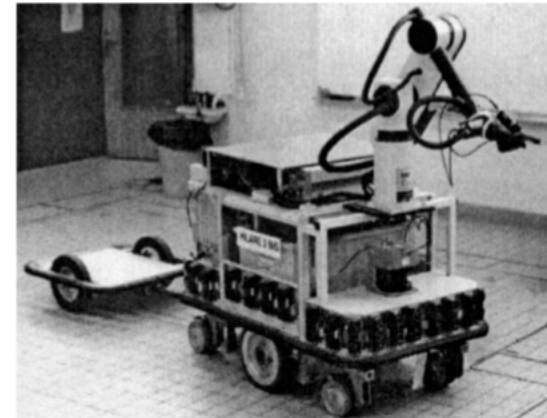
- Termination condition:
 - ◆ Let k = number of random draws since the last time a configuration was added to the roadmap
 - ◆ Stop when k reaches some value k_{\max}
- $1/k_{\max}$ is a probabilistic estimate of the ratio between the part of CS_{free} not covered by R and the total CS_{free}
 - ◆ For $k_{\max} = 1000$, the algorithm generates a roadmap that covers CS_{free} with probability 0.999

Implementation

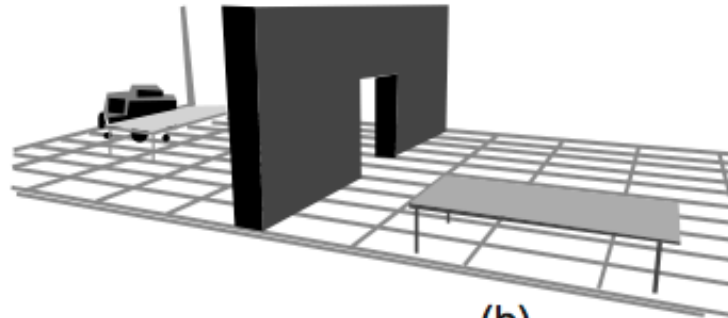
- Very efficient implementations
- Marketed products used in
 - ◆ Robotics
 - ◆ Computer animation
 - ◆ CAD
 - ◆ Manufacturing

Example

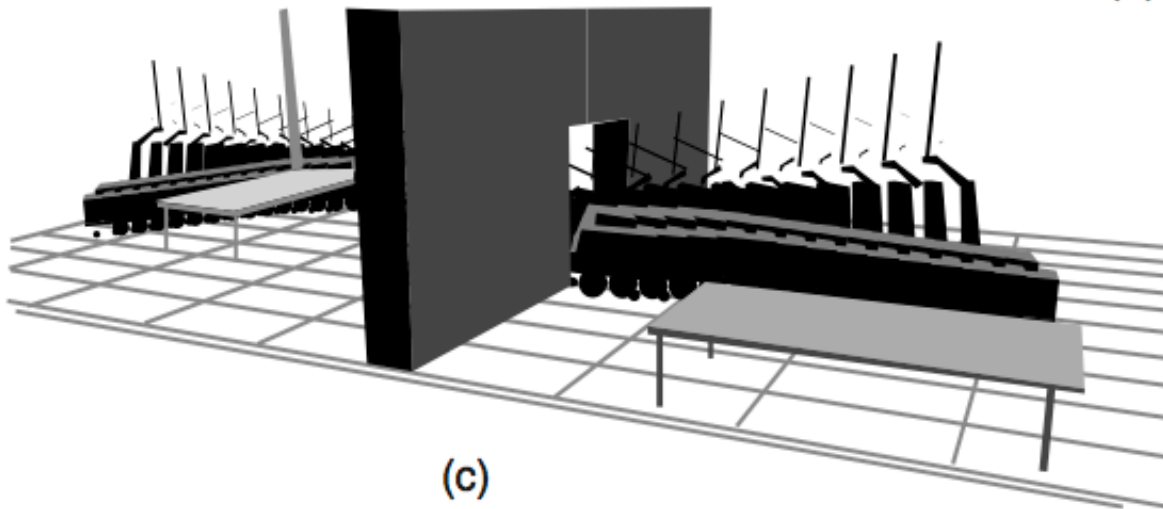
- Task: carry a long rod through the door
 - ◆ Roadmap: about 100 vertices in 9-dimensional space
 - ◆ Generated in less than 1 minute on a normal desktop machine



(a)



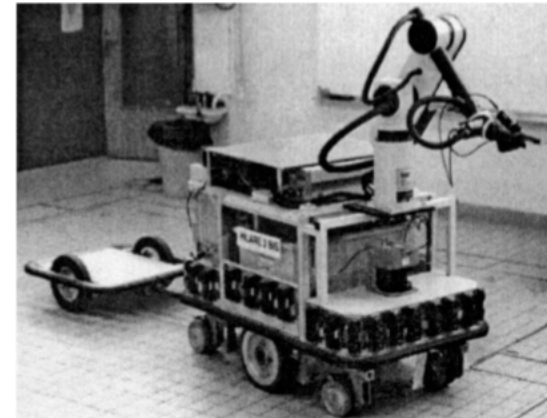
(b)



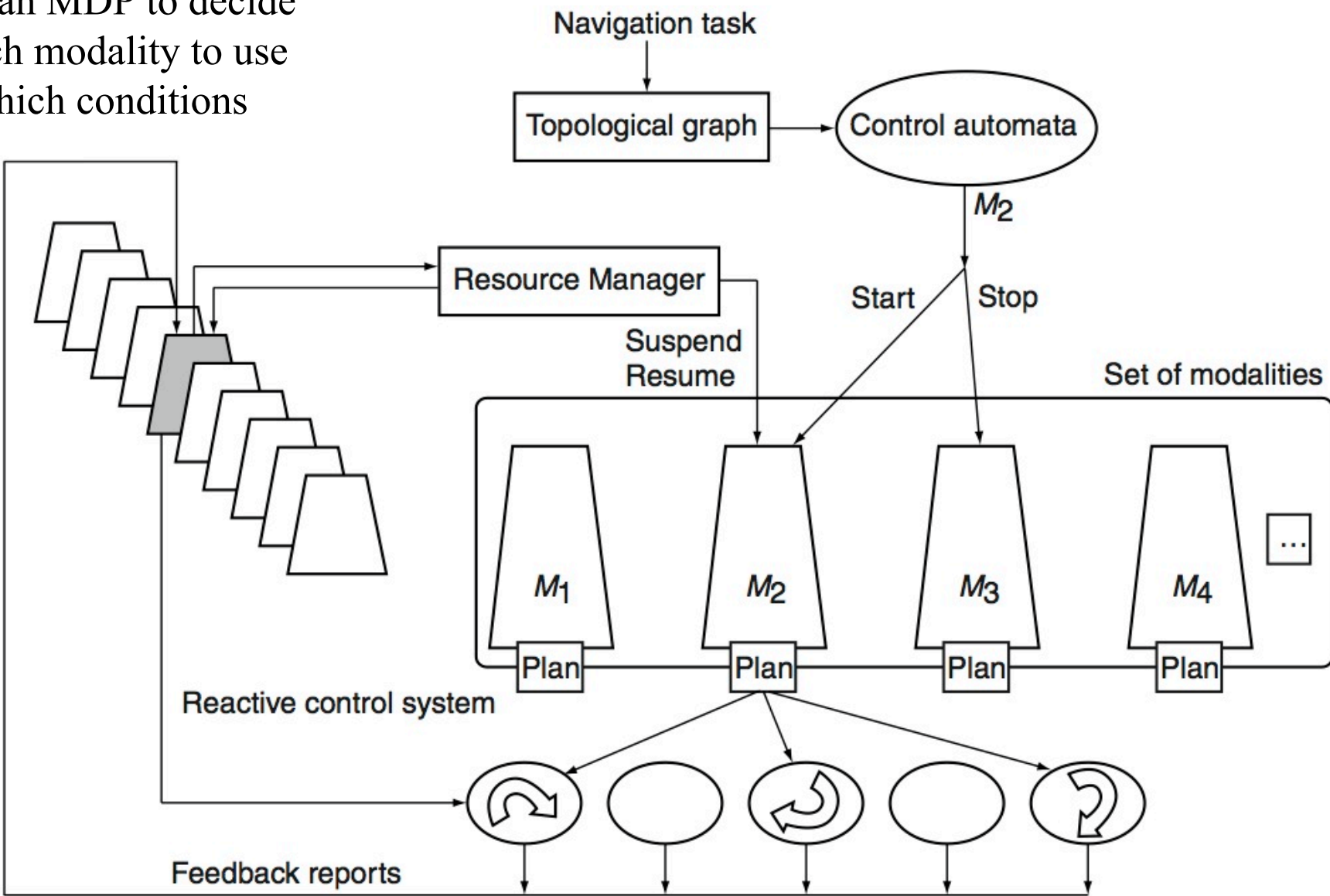
(c)

Planning for the Design of a Robust Controller

- Several sensors (sonar, laser, vision), actuators, arm
- Several redundant software modules for each sensory-motor (sm) function
 - ◆ Localizations
 - ◆ map building and updating
 - ◆ Motion planning and control
- Redundancy needed for robustness
 - ◆ No single method or sensor has universal coverage
 - ◆ Each has weak points and drawbacks
- Example: planning techniques

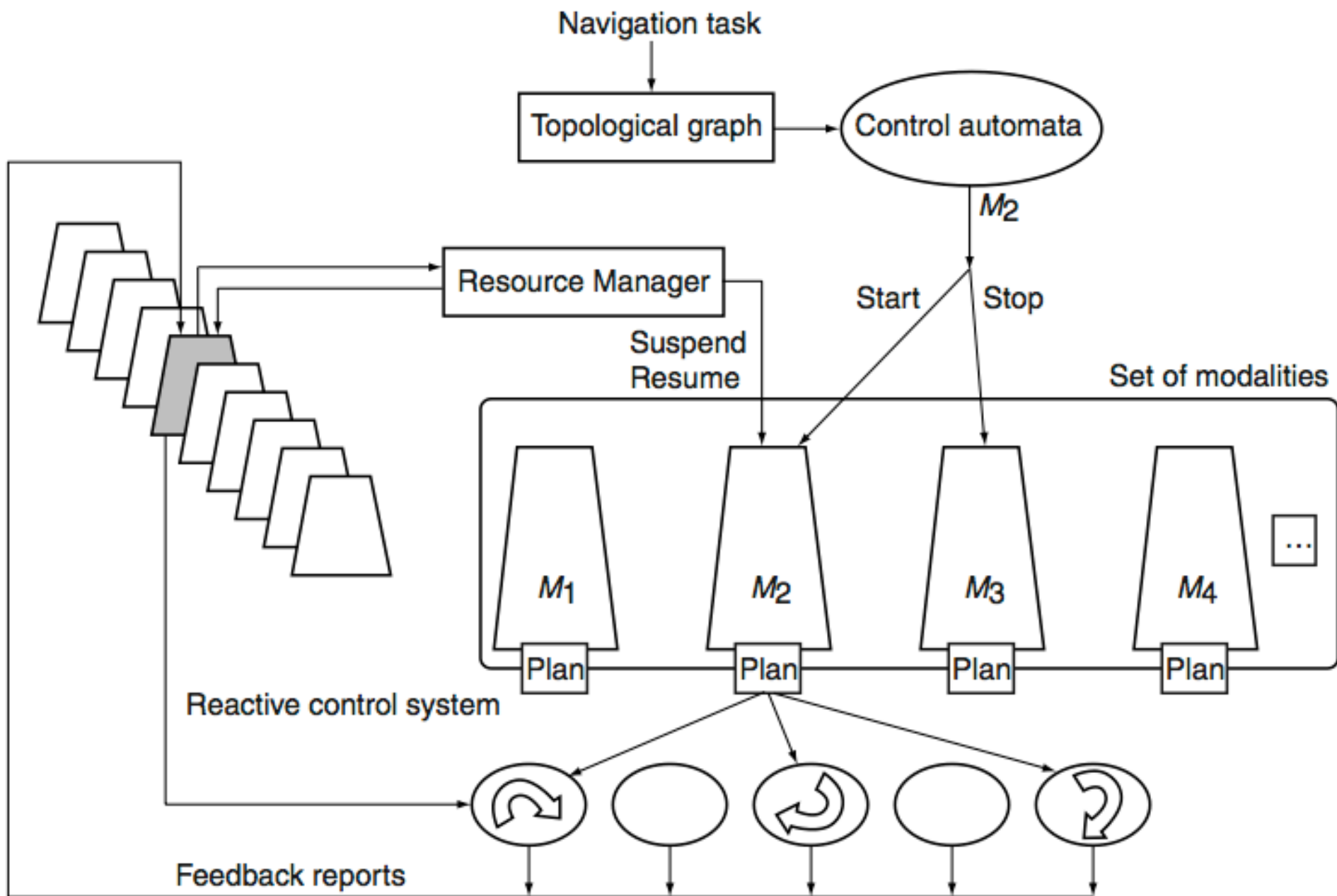


- Hilare has several Modes of Behavior (or *Modalities*)
- Each modality is an HTN whose primitives are *sm* functions
 - ◆ i.e., a way to combine some of the *sm* functions to achieve the desired task
- Use an MDP to decide which modality to use in which conditions



Sensory-Motor Functions

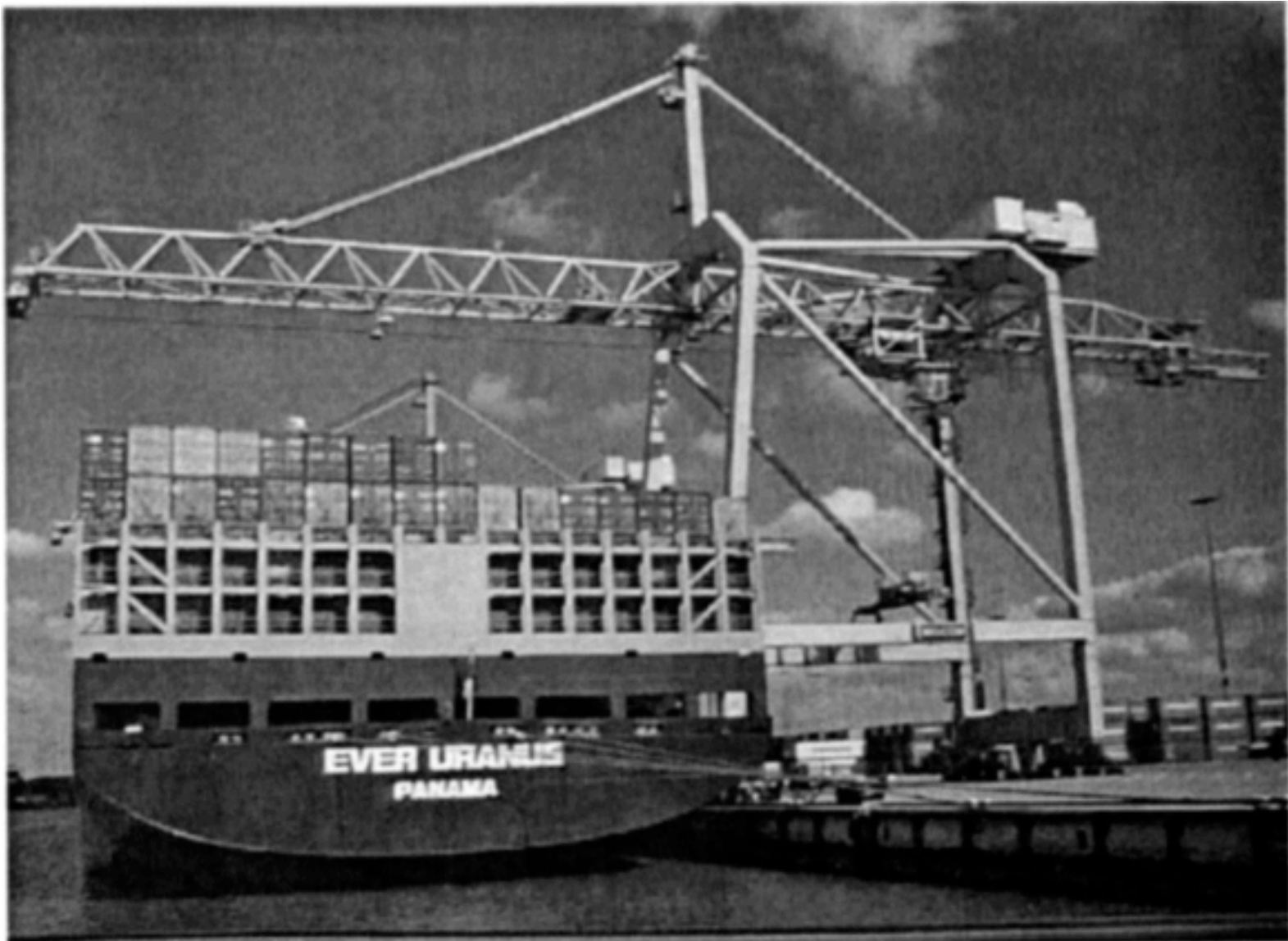
- Segment-based localization
 - ◆ Laser range data, extended Kalman filtering
 - ◆ Has problems when there are obstacles and/or long corridors
- Absolute localization
 - ◆ Infrared reflectors, cameras, GPS
 - ◆ Only works when in an area covered by the devices
- Elastic Band for Plan Execution
 - ◆ Dynamically update and maintain a flexible trajectory



Some Pictures You Might Like

- Here are some pictures of real dock environments

Loading the *Ever Uranus* in Rotterdam Harbor



A Dock Work Environment

