Summary of the research paper about

A ROBUST LAYERED CONTROL SYSTEM FOR A MOBILE ROBOT

1. Introduction

Autonomous robot control system must perform when boundary conditions are changing rapidly. The usual approach to build control system is to decompose the problem into series of functional units. In this method functional units are decomposed and implemented individually and tied up to form robot control system.

Requirements

Every requirements put constraints in the control system.

Multiple Goals: Though robot has multiple goals, the relative importance of goals is context dependent. Control system must give importance to high priority goals and also serving necessary low level goals.

Multiple Sensors: Sensors do not directly indicate the physical quantity values. They often give inconsistent reading due to either sensor error or measurement conditions remain outside the domain of applicability. So, robot must take decisions considering this all.

Robustness: The robot should be able to manage when sensors fail or environment changes drastically.

Additivity: Robots need more power to implement all the sensors and capabilities.

Other approaches

Multiple Goals: Elfes and Talukdar designed control language letting the user code for parallelism and code an exception path for unexpected conditions.

Multiple Sensors: Flynn 85 investigated use of multiple sensors where if one fails other still gives value. The robot operates in a mode where particular sensor type is used at a time while others are completely ignored. Depending on the environmental conditions and operational level of sensor subsystems, the data from one sensor tends to dominate.

Additivity: This is achieved in three ways without completely building the physical control system.

1) Utilizing excess wasted processor power.
2) Upgrading processors to an architecturally compatible faster system.
3) Adding more processors to carry a new load.

As cost of memory to processor routing system dominates the cost, number of processors that can be added is limited.

Starting Assumptions

The design decision principles are based on nine dogmatic principles.

1. Complex behavior may be the reflection of complex environment not the control system.
2. Things should be simple.
   1) When building a system of parts attention must be given to interfaces.
   2) If a design involves the solution of an unstable problem, then it is not a good solution

.

3. Map making is important even when idealized blueprints of environment are available.
4. The robot must model the world as three dimensional if it is allowed to cohabitate with humans.
5. Relational maps are useful rather than absolute coordinate systems.
6. We will not build artificial environment for our robot.
7. Visual data is much preferred than sonar data.
8. Self-calibration should happen all the time.
9. Building robots that can survive for days, weeks and months without human assistance.

The Physical Robot

The robot is about 17 inches diameter and 30 inches from the ground to top platform. Three parallel wheels are steered together, and the two motors are served a single microprocessor. The robot body always points in the direction of wheels.

It has 12 symmetrically arranged Polaroid flight range sensors and two Sony CCD cameras. It has Intel 8031 processor and communicates with off board processors via 12kbps duplex radio link. Effective bit rate is less than half of normal due to error correction. It switches cameras through a single channel video transmitter. Multiple Lisp machines equipped with demodulator and frame grabber work together to control the robots.

2. Levels and Layers

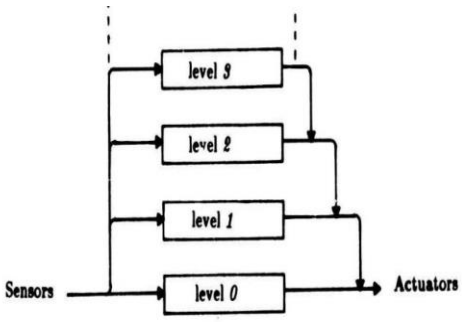
Decomposing the problem into pieces, solving subproblems for each piece , composing the solutions are the three steps to solve engineering problems. Mobile robot problem can be divided into subsets of sensing, mapping sensor data into a world representation, planning, task execution, motor control. They slice the problem based on the desired external manifestations of the robot control system. Level of competence is an informal specification of a desired class of behaviors for a robot on environments it will encounter.

They used following levels of competence.

0. Avoid contact with objects.
1. Wander aimlessly around without hitting things.
2. “Explore” the world by seeing places in the distance which are reachable.
3. Build a map of environment and plan routes from one place to another.
4. Notice changes in the static environment.
5. Reason about the world in terms of identifiable objects and perform tasks related to them.
6. Formulate and execute plans involving changing the world in desirable way.
7. Reason about the behavior of objects in the world and modify plans accordingly.

Each level of competence includes as a subset of each earlier level of competence. The main idea of levels of competence is that they can build layers of control system corresponding to each level of competence and simply add a new layer to an existing set to progress to the next higher degree of overall competence. They build a complete robot control system achieving 0 level competence and debugged thoroughly. Then build level1 control system which can interfere with level 0 data paths, but layer 0 continues to run unaware about the layer above it. This process is repeated to higher level of competence. This architecture solves the following problems,
Multiple goals-Each layer can work on individual goals concurrently.
Multiple Sensors- can ignore the sensor fusion problem using this architecture.
Robustness- Lower levels that are well debugged continue to run when higher levels are added. And intelligent use of multiple sensors results.

Additivity-Make each new layer run on its own processor.
They are free to use different decompositions for different sensor-set, task-set pairs. Chosen to build layers from a set of small processors that send messages to each other. Each processor is a finite state machine that can hold data structures. Processors send data through "wires" without handshakes or acknowledgement of messages, where data can be lost often. But there is no other form of communication or shared global memory. All processors are created equal that there is no central control on each layer. Each processor does its thing as best it can. The high-level layers subsume the role of lower layers by suppressing the inputs of processors and inhibiting the outputs by wires terminating from other processors.



3. A Robot Control System Specification Language

Robot layered control architecture has two aspects in components wise such that internal structure of the modules and the way in which they communicate.

3.1 Finite State Machines

Each module is a finite state machine which has a number of input lines and a number of output lines. Input lines have single element buffers. The most recently arrived message is always available for inspection. messages can be lost if a new one arrives on an input line before the last was inspected. There is a distinguished input to each module called reset. When the system first starts up all modules start in the distinguished state named "NIL". When a signal is received on the reset line the module switches to state "NIL".

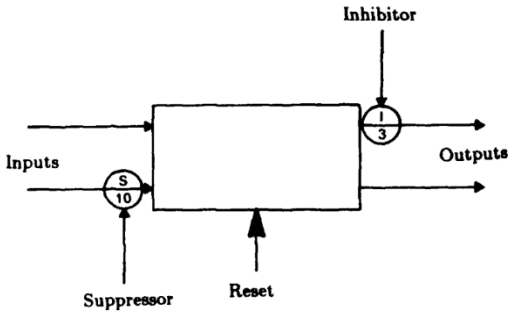
A state can be specified as one of four types.

1. Output.
2. Side effect.
3. Conditional dispatch.
4. Event dispatch.

```
(defmodule avoid
  :inputs (force heading)
  :outputs (command)
  :instance-vars (resultforce)
  :states
    ((nil (event-dispatch (and force heading) plan))
     (plan (setf resultforce (select-direction force heading))
           go)
     (go (conditional-dispatch (significant-force-p resultforce 1.0)
                               start
                               nil))
     (start (output command (follow-force resultforce))
            nil)))
```

In the above specification language, the **force input line** inputs a force with magnitude and direction found by treating each point found by the sonars as the site of a repulsive force decaying as the fifth power of distance. Function **select-direction** selects the instantaneous direction of travel by summing the forces acting on the robot. Function **significant-force-p** checks whether the resulting force is above some threshold-in this case it determines whether the resulting motion would take less than a second. And Function **follow-force** converts the desired direction and force magnitude into motor velocity commands.

3.2 Communication



The above figure shows that how the communication between input lines and output lines. Input signals can be suppressed and replaced with the suppressing signal. Output signals can be inhibited. A module can also be reset to state NIL.

For both suppression and inhibition, the time constants are written inside the circle. In the specification language, wires are written as a source followed by a number of destinations. For instance, the connection to the force input of the **avoid** module defined above might be the wire defined as:

(defwire (feelforce force) (runaway force) (avoid force))

Below, the suppression of the command input of the **motor** module is shown, a level 0 module by a signal from the level 1 module avoid.

(defwire (avoid command) ((suppress (motor command) 1.5)))

4. A Robot Control System Instance

After achieving level 0 and 1 by mobile robot control system now we have started implementation of level 2 bringing it to a stage which exercises the fundamental subsumption idea effectively. To that we need more work on an early vision algorithm.

Zeroth level

The lowest level layer of control makes sure that the robot does not come in to contact with other objects. To flee from moving obstacles and to stop collisions with stationary objects, robots require two tactics such as moving by its self and halting. A robot is not invincible and fast-moving object.

- Motor module allows it to send and receive commands to and from the actual physical robot directly. It accepts a command specifying angle and speed of turn, magnitude of forward travel and velocity. It sends a high, or busy, signal on a status line it maintains then waits in a delay-induced loop for it to be finished. If at any time a halt message is received, it commands the robot to halt and the module sets the status output line to low.
- The sonar module gets a vector of sonar readings, filter them and produces a robot centered map of obstacles in polar coordinates.

- The collide module monitors the sonar map and if it detects objects dead ahead it sends a signal on the halt line to the motor module. The collide module does not care whether the robot is moving. Halt messages sent while the robot is stationary are essentially lost.
- The feel force module sums the results of considering each detected object as a repulsive force, generating a single resultant force.
- The runaway module observes the force produced by the sonar detected obstacles and sends command to the motor module if it ever becomes significant.

First level

It means level 1 competence which imbues the robot to wander around without hitting obstacles when combine with zeroth. This depends in a large degree on zeroth level's aversion and it uses a simple heuristic to plan ahead a little in order to avoid potential collisions which would need to be handled by the zeroth level.

- The wander module forms a new heading for the robot every 10 seconds.
- The avoid module gets the result of the force computation from the zeroth level, and combines it with the desired heading to produce a modified heading. Output of the avoid module suppresses the output from the runaway module as it enters the motor module.

Second level

It is meant to add an exploratory mode of behavior to the robot, using visual observations to select interesting and desired places to visit even local obstacles present on its path.

- The grabber module confirms that level 2 has control of motors by sending a halt signal to motor module, then temporarily inhibiting number of communications in the lower levels so that no new actions can be started and waiting for the motor module to indicate that it is no longer controlling a robot motion. At this point the sensors able to give stable readings enough to plan a detailed motion, so a goal can be sent to the path plan module.
- The monitor module continuously observes the status of motor module, when it becomes inactive the monitor module queries the robot through a direct connection to get a reading from its shaft encoders on how far it has travelled and able to track each motion.
- The Integrate module gathers reports of motions from the monitor module and always sends its most recent result out on its integral line. It gets restarted by application of a signal to its reset input.
- The path plan module takes a goal specification and attempts to reach goal. To this it sends headings to the avoid module. The messages to avoid module checks random wandering of robot, so long as the higher-level planner remains active. When the position of the robot is close to the desired position it outputs the goal to the straighten module.
- The straighten module is responsible for modifying the final orientation of the robot. It thus sends its commands directly to the motor module and observes the integral line to make sure it is successful. For good measure it also prohibits the collision reflex, since there is no chance of a collision from forward motion. Straighten module would reinitiate another turn if the first were unsuccessful, but it does make for a smoother final reorientation.

Four of the inhibition and suppression nodes are used only to gain control of the lower levels of the system. The zeroth and first layers still play an active role during normal operation of the second layer. Extra monitoring of the integral output of the integrate module will confirm that if the robot wanders outside of the perceived limits of the corridor of free space, then it will stop and take more visual observations. This last capability will make additional use of the reset capability to halt the path planning behavior.

5. Performance

Simulation has to include all the real-world errors as much as possible. So that if they command the robot to turn by α it will turn by $\alpha+\Delta\alpha$. Here, $\Delta\alpha$ is the error due to the real-world environmental condition. For example, sonars may reflect many times and they may interfere each other or they could create a noise component in the reading. Sometimes the environmental humidity can affect the readings. It is important to have such a realistic simulation since it leads to the less incorrect algorithm while implementing in the real world.

Since we are using subsumption layered architecture here, for completing certain task at layer 2, it gets the help from the layer 1 and layer 0. For example, to move between from one goal to another goal with an aim, it uses the tactics **avoiding hitting obstacles** that is already programmed detailed in layer 1.

6. Implementation Issues

The layered control system is decomposed into asynchronous processors with low bandwidth communication and no shared memory should certainly assist in achieving that goal. New processors can be added to the network-there are no bandwidth or synchronization considerations in such connections. The bug in this idea is that finite state machines, have access to Lisp programs which require additional computational power.

1. A Spatial Processor
2. Sizing The Processors
3. Shorter Term Approaches

***