

multi-layered control system employing a number of different programming technologies. The first level takes care of smoothing sensor data and motor trajectories, the next level contains a library of local reactive navigation routines, and the final level does symbolic mapping and planning. We have showed how this approach can be productively applied to two other domains. If mobile robot navigation is viewed as the archetypal embodiment of these ideas, then everything is navigation.

References

- [1] R. Arkin, "Motor schema-based mobile robot navigation", *International Journal of Robotics Research*, 8(4):92-112, 1989.
- [2] R. Bolles, and R. Cain, "Recognizing and locating partially visible objects, the local-feature-focus method", *International Journal of Robotics Research*, 1(3):57-82, 1982.
- [3] M. Brady and H. Asada, "Smoothed local symmetries and their implementation", *International Journal of Robotics Research*, 3(3), 1984.
- [4] P. Burt, "Algorithms and architectures for smart sensing", In *Proceedings of the 1988 DARPA Image Understanding Workshop*, pages 139-153, 1988.
- [5] J. Connell, "Learning shape descriptions: Generating and generalizing models of visual objects", Technical Report AI-TR-853, Massachusetts Institute of Technology, 1985.
- [6] J. Connell, "A behavior-based arm controller", *IEEE Transactions on Robotics and Automation*, 5(6):784-791, 1989.
- [7] J. Connell, *Minimalist Mobile Robotics: A Colony-style Architecture for an Artificial Creature*, Academic Press, 1990. Also available as MIT AI TR 1151.
- [8] J. Connell, "SSS: A hybrid architecture applied to robot navigation", In *Proceedings of the 1992 IEEE Conference on Robotics and Automation*, pages 2719-2724, 1992.
- [9] J. Connell and M. Brady, "Generating and generalizing models of visual objects", *Artificial Intelligence*, 31:159-183, 1987.
- [10] B. Kuipers and Y. Byun, "A robust, qualitative method for robot spatial learning", In *Proceedings of the Sixth AAAI*, pages 774-779, 1988.
- [11] M. Mataric, "Integration of representation into goal-driven behavior-based robots", *IEEE Transactions on Robotics and Automation*, 8(3):304-312, 1992.
- [12] P. Pook and D. Ballard, "Contextually dependent control strategies for manipulation", Technical Report CS-416, University of Rochester, 1992.
- [13] S. Stansfield, "A robotic perception system utilizing passive vision and active touch", *International Journal of Robotics Research*, 7(6):138-161, 1988.

Figure 6: A *smoothed local symmetry* is defined as an extended collection of points such as P. P is the midpoint of the chord joining the tangent points of a circle inscribed between the edges of an object.

current SLS could be traced in the opposite direction. Alternatively, the system could continue to travel ballistically forward in the hope of finding a compatible symmetry nearby. It is often the case that SLS's become fragmented due to small glitches on the contour of an object. While all this was happening we would be monitoring the position of the focus of attention as well as the width of the symmetry region in order to derive some basic shape parameters for the piece. After one part of the object was fully explored, we could shift our attention to the ends or bounding contour of the current region in order to find another part connected to this one. The direction of this move and the relative positions and orientations of the two "corridors" could be used to produce a description for the "intersection". The advantage of this approach over a completely bottom-up analysis is that the whole image does not necessarily have to be analyzed. In addition, the strategic component might decide to activate special exploration behaviors for specific classes of objects.

The "route" problem can also be mapped onto visual recognition, this time by using active vision techniques. The strategic system might start by considering a number of possible object models. It would then direct the tactical component to explore a prospective candidate for some subpiece of one of the objects. By observing the trajectory followed by the focus of attention, the strategic component should be able to winnow its collection of models. Based on these initial observations, the focus could then be shifted to whichever other area was most useful for discriminating between the remaining models. Once the possibilities are narrowed down to a single object, and there is reasonable support for this interpretation, the system could stop exploring.

Figure 7: The reflectional symmetries of a figure define a number of perceptual paths. Using heuristics to traverse these paths gives rise to an object segmentation. The characteristics of each path, along with a description of their intersections, can be used to generate a semantic network representation of the object.

An attention-shifting approach such as this is crucial for systems, such as foveated sensors, which employ a serial assignment of resources. In head-eye systems this is squarely a robotics problem since shifting the focus of attention requires moving the whole imaging assembly. Even if physical motion is not necessary, this type of scheme can lead to efficient recognition procedures. For instance, Bolles' *feature-focus* method [2] has a similar flavor to it. Also, using a related technique, Burt [4] shows how to generate and match variable resolution *pattern tree* models of playing cards. These trees are basically a collection of iconic templates linked by their relative visual displacements.

Conclusion

Mobile robot navigation requires addressing some very fundamental issues. These include extracting symbols from signals, effectively using varying levels of abstraction, and integrating tactical control with strategic planning. Our solution to these problems is to use a

Figure 5: The strategic component only specifies that a can is somewhere out and down from the starting location. The hand can use its local sensors to move toward the can and grasp it despite environmental variation.

being directed under position control toward a waiting spatula. When the correct approximate position is achieved, the fingers are closed under force-sensitive behaviors. A stable grasp occurs when all the fingers have passed certain force and position thresholds. This event then triggers resumption of the position control mode to bring the spatula to the frying pan. The robot lowers the spatula until it detects contact with the pan. The spatula is then rocked back and forth to level it, after which it is extended forward until the far edge of the pan is detected by a force threshold. Hopefully this scoops up the egg. Finally, the hand executes a ballistic flipping motion to turn it over. This system has all the aspects of the mobile robot navigation problem: sensory pre-processing and servoing, local reactive path following, event triggered swaps of behavior sets, and strategic monitoring and interpretation of trajectories.

In this domain it is also possible to form an analogy to the “map” subproblem. This is best illustrated by the early work of Stansfield [13]. In her system there is a simple vision system that finds an object of interest and specifies its rough location. A finger equipped with a touch sensor is then sent in this direction to investigate. When the finger hits the object, the tactile sensor classifies the resulting contact as either extended, linear, or point. Each variety of contact basically defines

a “path” and has an associated Exploratory Procedure. For extended contacts the finger is moved around on the surface for a while. As the finger moves, its position is monitored in order to determine whether the surface is flat or curved in some direction. After this, the finger is slid toward the visual boundary of the object until it eventually senses a linear contact. The finger is then directed to follow this ridge in one direction. When a point contact is detected, the finger wobbles around to try to establish a new linear contact and hence find a new edge to follow. During all of this, a background system is recording the finger’s trajectory as it traces the contour of the object.

This is a prime example of local tactical processes controlling the motion of a robot while a strategic system watches for certain key events or qualitative properties of the path. The Exploratory Procedures would be part of the subsumption layer of the SSS architecture, while the monitoring facility and representation builder belong in the symbolic layer. Notice that the strategic system occasionally redirects the robot motion or changes which behaviors are running based on sensory events (such as the transition to a new type of contact). These policy shifts fit neatly into the contingency table structure. Thus, navigation provides useful tools for manipulation and haptic modelling.

Visual Description

A similar analogy can be used to transform the “map” problem into the visual description domain. The specific problem we address is the generation of semantic network representations from gray scale images of isolated objects. Once again, the first step is to look for pre-existing “paths” in the environment. One promising candidate is Brady’s *smoothed local symmetry* [3]. As shown in Figure 6, the SLS is a reflection symmetry between two segments of the object’s contour. This is related to the medial-axis or grass-fire transform, except that it does not require the inscribed circle to lie completely within the outline of the object. The reflectional symmetries of an object are potential axes of generalized cylinders and therefore can be used to describe many types of objects.

The top half of Figure 7 shows the intrinsic symmetries for the contour segments of a claw hammer and a heuristic segmentation produced by grouping various symmetries [5][9]. The bottom of the figure shows a semantic network that can be derived from this partitioning. While these results were not computed using the navigation metaphor, it is relatively simple to see how this might be done.

To build such a network representation we could start by moving the focus of attention to the center of the object. There would then be a tactical navigation behavior that chose two contour fragments and travelled along the “corridor” defined by their SLS. A method similar to Arkin’s potential fields [1] might be used for this. When the end of one symmetry was reached, the

intervals but divide the input state space into a small number of qualitatively equivalent *situations*. The top “Symbolic” layer takes this one step further and typically only reevaluates its surroundings when significant *events* occur (such as the transition from one situation to another). In terms of control, the symbolic layer influences the subsumption layer by switching on and off different groups of behaviors and changing parameters which influence their operation (such as the desired travel direction). The subsumption layer in turn generates commands which alter the set-points and gains of the underlying servos or adjust the thresholds of sensory pre-processing routines.

While all the layers are constantly a part of the sensor-action control loop, sometimes events require a more rapid response than is typically provided by most symbolic systems. For this reason, we add a structure called the *contingency table* to mediate the interface between the layers. The symbolic layer loads this table with advice about what to do should any of a number of anticipated events occur. When external stimuli trigger one of these patterns, the lower subsumption layer is immediately reconfigured as directed and the symbolic is notified of the event. At this point the symbolic system usually rewrites the table entries. However, it is free to modify the contingency table at any time – the table’s only purpose is to remove the hard real-time constraint. This particular mechanism is also convenient since many times the bulk of strategic control can be encoded as a finite-state machine. The contingency table can then be viewed as holding just the transition arcs emanating from the current node in the FSA controller.

Manipulation

Manipulation has a lot in common with navigation: both move a physical object through a cluttered environment toward some goal position. Therefore many of the same control ideas can be employed in this domain. As an example task, consider picking up a soda can from the top of a table. In a robot system developed at MIT [7][6] a laser-striping system finds potential cans and aligns the robot’s arm with the perceived location of the target. However, the vision system does not specify the exact location of the can nor does it plan a path for the hand. The sensory and motor system employed were basically too primitive and unreliable to allow this. Instead, just the existence of a target was communicated to the manipulation controller. Aside from roughly aligning the plane of the arm with the can, the location of the can is specified only implicitly by the urge to grope downwards and extend the arm.

Despite this lack of information, the hand is still able to acquire the can in most cases. This success rate is due to a number of end-point sensory systems (see Figure 4) which allow the hand to follow certain pre-defined “paths” in its environment. The hand initially starts out by descending until it hits a supporting surface. This event is detected by the two tip switches. After

Figure 4: The hand has a number of sensors on it to aid in local navigation.

this, the hand knows it is probably in the vicinity of horizontal surface so it lifts upward and progresses forward and down. If it fails to hit the table again within some interval, it reverts to the descending mode. Otherwise, it bounces along the table until the crossed proximity detectors in the front sense an obstruction. Then, as long as the obstruction is still sensed, the hand lifts up in an attempt to surmount the barricade. Typically the proximity signal vanishes just short of the top, at which point the hand proceeds forward again. If the obstruction was a vertical cliff, the hand will ram into it and activate the wrist overload sensor. This in turn causes the hand to back off slightly then rise a fixed distance to get it past the edge. On the other hand, the obstacle might have been the can it was looking for. In this case, when the hand extends forward the can will pass between its finger and break a light beam. This triggers the hand’s closure reflex.

This combination of behaviors is quite effective at grabbing cans. Figure 5 shows the path of the fingertips during two attempts. Notice that even if the can is elevated above the table or secluded behind a ridge, the hand is able to find and grasp it. As with our solution to the “route” subproblem in mobile robot navigation, the strategic component only needs to supply a minimal amount of information about the goal. The specific location and a suitable trajectory for getting there are determined autonomously by the behavior-based tactical component.

Pook [12] describes a similar system for flipping eggs. Her scenario starts with a multi-fingered robotic hand

Global Mapping

As the robot progresses along a path segment it constantly looks for openings to either side. As mentioned before, these are sensed by the long range proximity detectors on each side of the robot. When traversing a particular commanded route, the robot expects to travel a specific distance and then turn left or right. Thus, it normally filters the detected opening “events” by ignoring all of them which are not at the desired distance, or which are in the wrong direction. However, to build a map we can instead have the robot record all such events along with their displacements from the start of the path segment. Such a map is shown in Figure 2. The rough distances between corner nodes and the approximate orientation of halls are derived from onboard odometric measurements. These estimates are then refined by applying the constraint that all halls are basically straight and that they generally intersect at right angles.

Figure 2: The robot monitors certain sensors during path traversal. It then combines these observations with geometric constraints from the environment to build symbolic maps.

The important part of this construction procedure is that the symbolic system builds up a map by the noting certain features along the trajectory autonomously generated by reactive local navigation behaviors. While our system does look for explicit landmarks, it does not require any precise physical configuration to be present nor does it record much geometric detail about its world. It only remembers the direction and position of the openings since this is the only information it will need to communicate to the behavior-based system in order to successfully navigate to this place again.

This is an effective approach and similar systems have been used by a number of other researchers as well. For instance, Mataric [11] developed a robot which employs a distributed map for navigation. In her scheme the nodes in the map correspond to regions in which the

robot’s heading is constant and there is a wall consistently to the right, left, or both sides. Although in the same spirit, this representation is a bit more qualitative as it does not record the length of any of the segments nor the rough positions of intersections. Kuipers [10] has also proposed a scheme based on following the local minima of certain functions defined by the perceived sensory data. The intersections of such paths are used to define distinctive places. These nodes are then stored in a graph representation based on relative position estimates, and the links between them are marked with process that was in control during each segment.

Control Architecture

Our system divides the navigation problem into a tactical piece and a strategic piece. Each of these is most naturally programmed in a particular style. The tactical component can be easily encoded by a number of concurrently running propositional rules. The strategic component, on the other hand, is more conveniently coded in a symbolic fashion. Underlying both of these are some basic signal processing (e.g. sonar range determination) and servo-control (e.g. base velocity regulation) routines which are normally implemented in a functional or procedural manner. These observations form the basis for the SSS architecture shown in Figure 3. This architecture is generally applicable to any system which makes the tactical/strategic distinction.

Figure 3: The SSS architecture combines three control techniques which can be characterized by their treatment of time and space. Special interfaces allow the layers of this system to cooperate effectively.

In the SSS architecture, the bottom “Servo” layer operates in the continuous time and continuous space domains. That is, the processes here run continuously and usually deal with vectors of real numbers. The routines in the next level up, “Subsumption”, also run at regular

in useful ways. One such cue is reflectional symmetry. Each symmetry generates an axial “path” and therefore defines a generalized cylinder which can be entered as a node in the semantic network. We can associate features with this node by observing the cross-sectional characteristics of the area during traversal of this path. When the current axis ends, the strategic component can determine the length of the most recent object part and also record its relations to previous parts as links in the network.

We will describe both these scenarios in more detail later, as well as some other applications. Let us now examine a real mobile robot navigation system and its supporting architecture to see what insights can be gleaned.

Navigation

The primary task for our robot, TJ, is to navigate from room to room along hallways in an office building [8]. It is able to do this fairly reliably in an unmodified environment (no stripes or beacons) at a relatively high speed (32”/sec on average) with a large number of locations (approximately 100 offices). As mentioned above, the robot does not store all the geometric details of its environment. Rather, it has a number of reactive routines for avoiding collisions, veering around obstacles, following walls, and entering doors.

Supervising this tactical component is a strategic component that provides guidance of the form “go 51 feet then take the first left”. This symbolic system uses a coarse geometric map consisting of a number of nodes (the corners of doors and hallways) connected by links specifying the distance and rough direction between them. This is the only information in the map. Features such as the widths of corridors, positions of objects affixed to the walls, locations of piles of boxes, and status of door openings are not recorded. The strategic system implicitly trusts the tactical system to cope with such variations when they arise.

Local Routines

Most of the time the robot is controlled by a collection of behaviors. All these behaviors base their control suggestions on the current state of a number of sensory systems specially crafted for a particular task. For instance, Figure 1 shows the sensor arrangement and control rules governing one aspect of wall following. On each side of the robot there are three short range proximity detectors that return a “1” (thick line) when the robot is close to a wall, and a “0” (thin line) otherwise. There is also another central sensor with a longer range (about 7 feet) that is used for detecting doorways and intersections. While this array was developed specifically with flat walls in mind, it also works well in circumnavigating boxes and rounding corners.

In operation, the three bits produced by the short range array form the basis for controlling the steering of the robot. The activation distances of the front and

back diagonal sensors are set to 16”, while the center sensor is set to 10”. Thus, when the robot is from 10” to about 11” away from the wall, just the two diagonal sensor bits will be high. This is the desired state and no corrective steering is required. However, if none of the bits is on (a) the robot needs to turn toward the wall. If both of the diagonal sensors is on and the center proximity detector is also on (b) the robot is too close to the wall and should steer away. Once the correct wall offset has been established, the robot uses these same three bits to adjust its following angle. If it starts angling toward a wall (c), the rear sensor will go to zero. This signals the robot to turn away. Similarly, if the robot is slowly veering off (d), the front sensor will go low and cause a turn in the reverse direction.

Figure 1: Wall-following depends on recognizing a number of local situations. Several of these classes can be discriminated using just four binary proximity sensors (the solid lines). The robot travels in the direction of the internal arrowhead; its past path is dotted.

The point of this example is that by carefully selecting a few local sensory features, most points in the resulting state space can be mapped directly to instantaneous control actions. The paucity of the sensory information required is a product of relying on a number of constraints. Some of these are environmental regularities, such as the fact that there are long straight corridors. Others are contextual in nature. For instance, after executing a turn around a corner under the control of another set of behaviors, the robot will already be roughly aligned with the axis of the new hall and at approximately the correct offset.

With a surprisingly small set of such local behaviors, the robot can automatically deal with the stereotyped portions of its environment, even though they are not geometrically identical. It is this ability which allows the strategic component to provide only vague instructions, such as the rough travel distance and qualitative turn direction, for each path segment.

Extending the Navigation Metaphor to Other Domains

Jonathan H. Connell
IBM T.J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598
jhc@watson.ibm.com

Abstract

Navigation is one of the standard problems in the mobile robot domain. Here we describe how this was solved by one particular system developed at IBM. We then show how the same basic decomposition idea, as well as the specific control architecture developed for this task, can be ported to the domains of object manipulation and visual description. There are a number of advantages to viewing these two new tasks as variations of the navigation problem.

Introduction

The complete navigation problem can be broken down into the related subproblems of route traversal and map generation. A major issues in both of these is: At what level of detail should the external environment be represented? For moment-to-moment steering it seems the robot would need as much detail as possible. However, at the user interface level, one does not want to deal with coordinate frames or voxels. The obvious answer to this conundrum is to have multiple types of representations each specialized for a particular task. Yet translating between representations and maintaining consistency across levels is difficult.

These problems can be largely side-stepped as long as one does not insist on keeping a detailed, global map of the world. In many cases, the robot only needs to know the fine-grained details of its world locally. After the robot passes the area, the associated data can be discarded. This is especially true since many pieces of information, such as the position of people in the hall or how widely opened some door is, are likely to change before the robot returns to this location again. Generally, it is not useful for the robot to call up a blueprint of an area when the human user refers to the intersection of the E and F corridors. Items such as the location and sizes of all door knobs, floor moldings, trash cans, fire extinguishers, etc. are usually irrelevant. Thus, we do not try to reconstruct the world in its entirety.

Our solution is to instead have the robot under the control of a number of mostly local, mostly stateless tactical *behaviors* for the majority of the time. Each of these behaviors attends to just those features that are important for its particular function, and then only for as long as such information is necessary for the immediate task. Of course, for both of the navigation subproblems there is usually a small amount of data that does need to be maintained permanently. This information, such as a rough floor plan of the building, is kept in a more centralized strategic component. While we do not worry about being able to translate freely between the representations used by different behaviors, it is necessary for this strategic component to detect the occurrence of certain key events. Fortunately, we can do this indirectly by observing specific aspects of the trajectory generated by the local behaviors.

The “route” half of this solution can be extended to manipulation by driving around the robot’s hand instead of its body. Consider the basic operation of reaching toward and grabbing an object. Once the user sets up a strategic goal by specifying a rough desired position, the hand can use a tactical control system based on end-point sensing to circumvent obstructions on the way to this location. After it arrives at the destination, other types of local sensing can be used to establish a firm grasp on the object. This is in contrast to approaches that uses a detailed CAD-type object models to devise a stable grasp. Similarly, we do not try to build up an accurate internal reconstruction of the target area with some sort of 3-D sensor and then plan a path for manipulator in configuration space. While these other techniques are very general and work even in cluttered spaces with peculiar shaped objects, the commonly encountered situations are much simpler and can usually be solved by a combination of local methods.

The “map” half of our approach is particularly useful for building semantic network descriptions of visual objects. Here, it is the focus of attention (or a high-resolution fovea) which is being driven around instead of the robot’s body. Once again we can use spatially local features of the image to guide the robot’s attention