Appears in the Proceedings of the 1992 IEEE Conference on Robotics and Automation (ICRA-92), pp. 2719–2724.

SSS: A Hybrid Architecture Applied to Robot Navigation

Jonathan H. Connell

IBM T.J. Watson Research Center, Box 704, Yorktown Heights NY 10598

Abstract

This paper describes a new three layer architecture, SSS, for robot control. It combines a servo-control layer, a “subsumption" layer, and a symbolic layer in a way that allows the advantages of each technique to be fully exploited. The key to this synergy is the interface between the individual subsystems. We describe how to build situation recognizers that bridge the gap between the servo and subsumption layers, and event detectors that link the subsumption and symbolic layers. The development of such a combined system is illustrated by a fully implemented indoor navigation example. The resulting real robot, called “TJ", is able automatically map office building environments and smoothly navigate through them at the rapid speed of 2.6 feet per second.

1: Introduction

In the “SSS" architecture (an acronym for “servo, subsumption, symbolic" systems) we have tried to combine the best features of conventional servo-systems and signal processing, with multi-agent reactive controllers and state-based symbolic AI systems.

For instance, servo-controllers have trouble with many real-world phenomena which are not understood well enough to be modelled accurately or which are non-linear. Behavior-based or subsumption systems (e.g. [2, 5]), on the other hand, do not impose as many modelling constraints on the world and are good at making rapid, radical decisions. Yet such systems often yield jerky motions due to their slow sample rate and their discrete view of the world. This shortcoming can in turn be easily rectified by adding appropriate servo-systems which are particularly good at making smooth motions.

Behavior-based systems also have problems with world modelling and persistent state. Since behavior-based systems are often implemented in a distributed fashion, there is no good place to put a world model. Indeed, many of the adherents of this school claim that this is a beneficial feature of such systems [3]. However, for some tasks, such as navigation, it is certainly *convenient* to have higher-level centralized representations. This is the forté of standard hierarchical symbolic programming languages. The usual stumbling block of such systems, real-time control, can be finessed by delegating tactical authority to the subsumption and servo control layers.

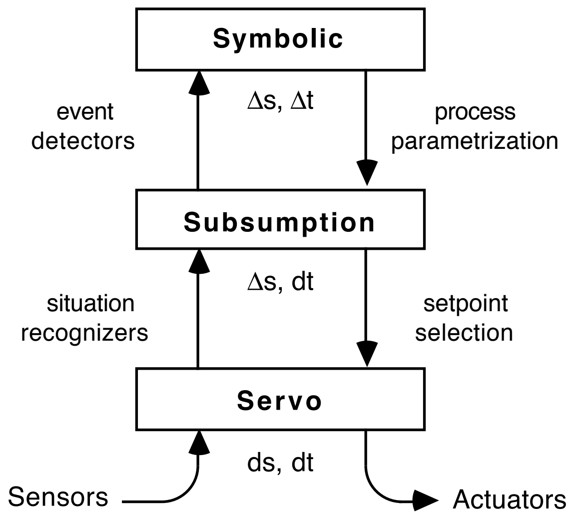

Figure 1 - The SSS architecture combines 3 control techniques which can be characterized by their treatment of time and space. Special interfaces allow the layers of this system to cooperate effectively.

The three layers in our system come from progressively quantizing first space then time. As shown in figure 1, the servo-style system basically operates in a domain of continuous time and continuous space. That is, these systems constantly monitor the state of the world and typically represent this state as an ensemble of scalar values. Behavior-based system also constantly check their sensors but their representations tend to be special-purpose recognizers for certain types of situations. In this way behavior-based systems discretize the possible states of the world into a small number of special task-dependent categories. Symbolic systems take this one step further and also discretize time on the basis of significant events. They commonly use terms such as “after X do Y" and “perform A until B happens". Since we create temporal events on the basis of changes in spatial situations, it does not make sense for us to discretize time before space. For the same reason, we do not include a fourth layer in which space is continuous but time is discrete.

In order to use these three fairly different technologies we must design effective interfaces between them. The first interface is the command transformation between the behavior-based layer and the underlying servos. Subsumption-style controllers typically act by adjusting the **setpoints** of the servo-loops, such as the wheel speed controller, to one of a few values. All relevant PID calculations and trapezoidal profile generation are then performed transparently by the underlying servo system.

The sensory interface from a signal-processing front-end to the subsumption controller is a little more involved. A productive way to view this interpretation process is in the context of “matched filters" [16, 6]. The idea here is that, for a particular task, certain classes of sensory states are equivalent since they call for the same motor response by the robot. There are typically some key features that, for the limited range of experiences the robot is likely to encounter, adequately discriminate the relevant situations from all others. Such “matched filter" **recognizers** are the mechanism by which spatial parsing occurs.

The command interface between the symbolic and subsumption layers consists of the ability to turn each behavior on or off selectively [7], and to **parameterize** certain modules. These event-like commands are “latched" and continue to remain in effect without requiring constant renewal by the symbolic system.

The sensor interface between the behavior-based layer and the symbolic layer is accomplished by a mechanism which looks for the first instant in which various situation recognizers are all valid. For instance, when the robot has not yet reached its goal but notices that it has not been able to make progress recently, this generates a “path-blocked" **event** for the symbolic layer. To help decouple the symbolic system from real-time demands we have added a structure called the “contingency table". This table allows the symbolic system to pre-compile what actions to take when certain events occur, much as baseball outfielders yell to each other “the play's to second" before a pitch. The entries in this table reflect what the symbolic system expects to occur and each embodies a one-step plan for coping with the actual outcome.

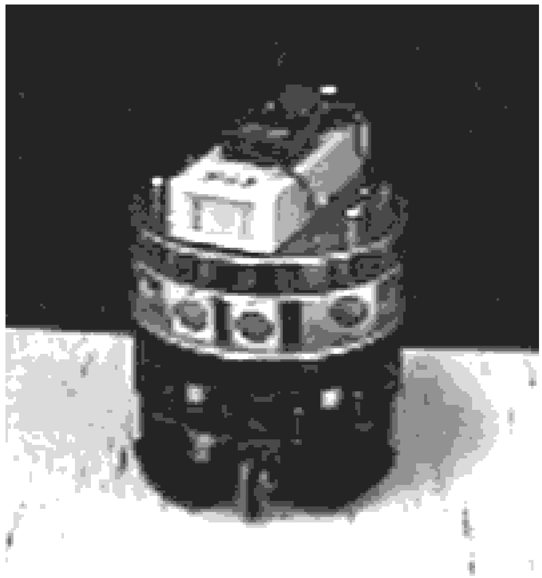

Figure 2 - TJ has a 3-wheeled omni-directional base and uses both sonar ranging and infrared proximity detection for navigation. All servo-loops, subsumption modules, and the contingency table are on-board. The robot receives path segment commands over a spread-spectrum radio link.

2: The navigation task

The task for our robot (shown in figure 2) is to map a collection of corridors and doorways and then rapidly navigate from one office in the building to another. To accomplish this we had to address two basic navigation problems. The first of these is compensating for the variability of the environment. There are often people in the halls, doors open and close by random amounts, some days there are large stacks of boxes, and the trash cans are always moving around.

We solve the variability problem by restricting ourselves to very coarse geometric maps. We record only the distance and orientations between relevant intersections. The actual details of the path between two points are never recorded, thus there are never any details to correct. However, to make use of such a coarse map the robot must be able to reliably follow the segments so described [4]. Fortunately, behavior-based systems are particularly good at this sort of local navigation.

The other basic navigation problem is knowing when the robot has arrived back in a place it has been before. The difficulty is that, using standard robot sensory systems, an individual office or particular intersection is not distinguishable from any other. One approach is to use odometry to determine the absolute position of the robot.

However, it is well known that over long distances such measurements can drift quite severely due to differing surface traction and non-planar areas.

We solve the loop problem by exploiting the geometry of the environment. In most office buildings all corridors are more or less straight and meet at right angles. Therefore we measure the length of each path segment and treat this as a straight line. Similarly, when the robot switches from one path to another we force the turn to be a multiple of 90 degrees. This is essentially an odometric representation which is recalibrated in both heading and travel distance at each intersection. In this way we maintain a coarse *(x, y)* position estimate of the robot which can be compared to the stored coordinates of relevant places.

2.1: Tactical navigation

Tactical, or moment-to-moment control of the robot is handled by the servo and subsumption layers of our architecture. The servo layer consists of two velocity controllers, one for translation and one for rotation, on the robot base. These proportional controllers operate at 256 Hz and have acceleration-limited trapezoidal profiling.

Built on top of these servo-controllers are various reactive behaviors which run at a rate of 7.5Hz. One of the more important of these is wall following. For this we use a carefully arranged set of side-looking infrared proximity detectors. The method for deriving of specific responses for each sensory state is detailed in [4, 6] for similar systems. While some researchers have tried to fuse sensory data over time and then fit line segments of it, most “walls" that we want to follow are not really flat. There are always gaps caused by doors, and often junk in the hall that makes the walls look lumpy. This is the same reason we did not try to implement this activity as a servo-controller: it is very hard to directly extract angle and offset distance from the type of sensory information we have available. The matched filter approach lets us get way with only partial representations of the environment relative to the robot.

There are also two tactical navigation modules based on odometry. The first of these looks at the cumulative travel and slows or stops the robot when the value gets close to a specified distance. A similar setup exists based on the average heading of the robot. The average heading is computed by slowly shifting the old average heading value toward the robot's current direction of travel. If the robot is only turning in place the average heading does not change, but after the robot has travelled about 5 feet in a new direction the value will be very close to the actual heading. A special behavior steers the robot to keep this “tail" straight behind, which in turn causes the robot to remain aligned with its average heading. This is very useful for correcting the robot's direction of travel after it has veered around an obstacle. If the detour is short, the average heading will not have been affected much.

The average heading signal provides an interesting opportunity for the symbolic system to deliberately “fake out" the associated behaviors. For instance, the symbolic system can cleanly specify a new direction of travel for the robot by yanking around the robot's “tail". This method is better than commanding the robot to follow a new absolute direction, especially for cases in which the robot was not aligned precisely with the original corridor or in which the new corridor seems to bend gradually (in actuality or from odometric drift). Instead of forcing the robot to continually scrape its way along one wall or the other, the average heading will eventually adjust itself to reflect the direction along which progress has been made and thereby allow the robot to proceed smoothly.

Although they sound like servo-controllers, the two odometric behaviors were put in the subsumption layer for two reasons. First, they do not require fine-grained error signals. The alignment behavior is quiescent if the robot is “close" to the right heading, and the travel behavior only slows the robot when it is “near" the goal. Second, and more importantly, we wanted these behaviors to interact with other subsumption behaviors. For example, the alignment behavior takes precedence over the part of wall following that moves the robot closer to a surface, however it is not as important as collision avoidance. Many of these other behaviors are necessarily cast in a subsumption style framework because of the limited quality of sensory information available. Thus, to accommodate the appropriate dominance relations, the alignment and travel limiting behaviors were also included in this layer.

2.2: Strategic navigation

The strategic part of navigation – where to go next – is handled by the symbolic layer. To provide this information, our symbolic system maintains a coarse geometric map of the robot's world. This map consists of a number of landmarks, each with a type annotation, and a number of paths between them, each with an associated length. The landmarks used in the map are the sudden appearance and disappearance of side walls. These are detected by long range IR proximity detectors on each side of the robot. Normally, in a corridor the robot would continuously perceive both walls. When it gets to an intersection, suddenly there will be no wall within range of one or both of these sensors. Similarly, when the robot is cruising down a corridor and passes an office, the IR beam will enter the room far enough so that no return is detected.

Once a map has been created, an efficient route can be plotted by a spreading activation algorithm [10] or some other method. To traverse this route, the symbolic system enables whatever collection of subsumption modules it deems appropriate for the first segment and parameterizes their operation in appropriate ways. The symbolic system does not need to constantly fiddle with the subsumption layer, it only has to reconfigure the layer when specific events occur. In our navigation example, this typically happens when the robot has reached the end of one path segment and needs to have its course altered to put it on the next segment. In this case, the alteration of subsumption parameters must be swift or the robot will overshoot the intersection. To relieve this real-time burden from the symbolic system we create a “contingency table" such as shown in figure 3 (in pseudo-LISP code). This structure is similar to the “event dispatch" clauses in the original subsumption architecture [2].

```
(do-until-return
  (setq recognizers (check-situations))
  (cond ((and (near-distance? recognizers)
              (aligned-okay? recognizers)
              (left-opening? recognizers))
         (inc-heading! 90)
         (new-travel! 564)
         (return recognizers))
        ((beyond-distance? recognizers)
         (inc-heading! 180)
         (new-travel! 48)
         (return recognizers))
        ((no-progress? recognizers)
         (disable! stay-aligned)
         (enable! scan-for-escape)
         (return recognizers))
        (t nil)))
```

Figure 3 - The “contingency table" continuously monitors a collection of special-purpose situation recognizers. When a specified conjunction occurs, this “event" causes a new set of permissions and parameters to be passed to the subsumption system. After this, the symbolic system builds a new table.

The contingency table allows the symbolic system to specify a number of events and what response to make in each case. As suggested by the code, this contingency table module continually checks the status of a number of special-purpose situation recognizers. When one of the listed conjunctions occurs, the module performs the specified alterations to the subsumption controller then returns the triggering condition to the symbolic system. If two or more events occur simultaneously, the action for the one listed first is taken. After this one burst of activity the old contingency table is flushed and the symbolic system is free to load an entirely new table.

For the navigation application, the contingency table includes a check for an opening in the correct direction at an appropriate path displacement, along with commands to reset the orientation and travel distances for the next path segment (see figure 3). Notice that the robot also checks to make sure it is aligned with the average heading when it observes an IR opening to the left. This is to keep the robot from mistakenly triggering the next subsumption configuration just because the IR signal happened to vanish as the robot was veering around some obstacle in its path.

3: Experimental results

Our first experiment was aimed at validating the claim that environmentally constrained odometry allows us to solve the loop navigation problem. For this we provided the robot with a rough path to follow of the the form: (($travel_1$ $turn_1$) ($travel_2$ $turn_2$) ...). Travel was specified as an integral number of inches and turns were specified as an integral number of degrees. The top half of figure 4 shows the path the robot took according to its odometry. The circles with small projections indicate where the robot observed openings in the directions indicated. The circles are the same size as the robot (12 inches diameter) to provide scale. Notice that neither of the loops appears to be closed. Based on this information it is questionable whether the corridor found in the middle of the map is the same one which the robot later traversed.

The symbolic map, which appears in the lower half of figure 4, correctly matches the important intersections to each other. In this map, nodes are offset slightly to the side of the path segments. Each node's type is denoted iconically as a corner – two short lines indicating the direction of the opening and the direction of free travel. This symmetry reflects the fact that the robot is likely to perceive the same corner when coming out *through* the marked aperture. Corner nodes are placed at the robot position where they are sensed; no adjustment is made for the width of the corridor. This is partially compensated by a wide tolerance in matching nodes.

When a new opening is detected the robot compares it with all the other nodes in the map that have similar opening directions. If there is a match which is no more than 6 feet from the robot's current position in either x or y, the new opening is considered to be a sighting of the old node. In this case we average the positions of the robot and the node and move both to this new location. This merging operation is why the corridors do not look perfectly straight in the symbolic map. However, when the robot is instructed to follow the same path a second time and update the map, the changes are minimal.

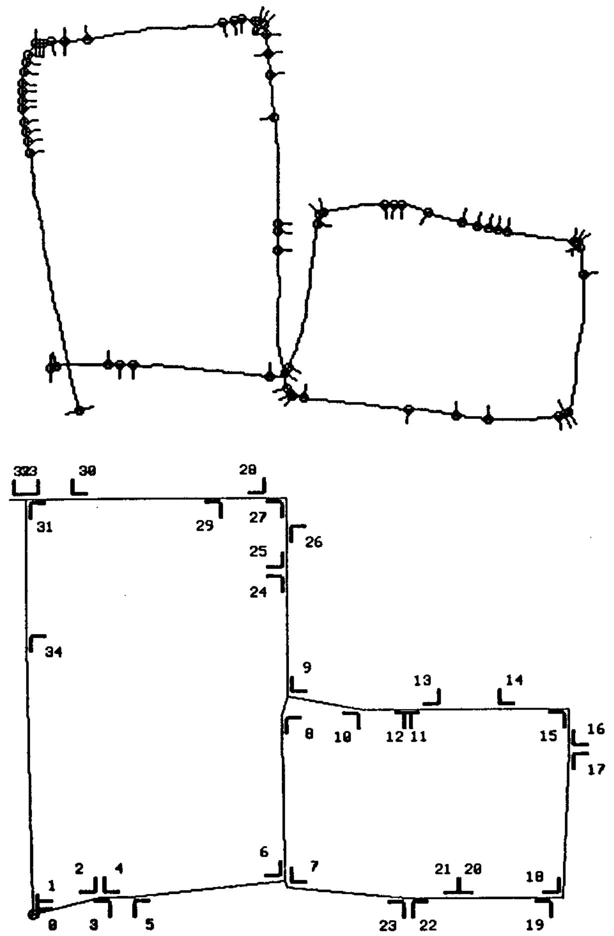

Figure 4 - Geometric constraints of the environment allow the robot to build maps with loops despite poor absolute odometry. The top picture shows the path the robot thought it took. The hair-like projections indicate openings it sensed. The lower picture shows the map the robot built during this exploration.

The second experiment shows that the subsumption system is sufficiently competent at local navigation to allow the use of a coarse geometric map. In this experiment we had the symbolic system plan a path (from node 10 to node 5) using the map it generated in the first part. We then told it to configure the two lower layers of the architecture to follow this path. The result of five consecutive runs is shown in figure 5. The displayed paths are based on the robot's odometry and hence do not accurately reflect the robot's true position in space over large displacements. However, the local details are qualitatively correct (some of the angularity is due to the time lag between successive position readings). On different runs we started the robot in slightly different directions. We also altered the positions of various obstacles along the path and changed the states of some of the doors on the corridor. This lead to slightly different forms for each of the runs. Despite these variations in initial heading and the configuration of the environment, the symbolic system was able to successfully navigate the robot to its goal in all 5 runs.

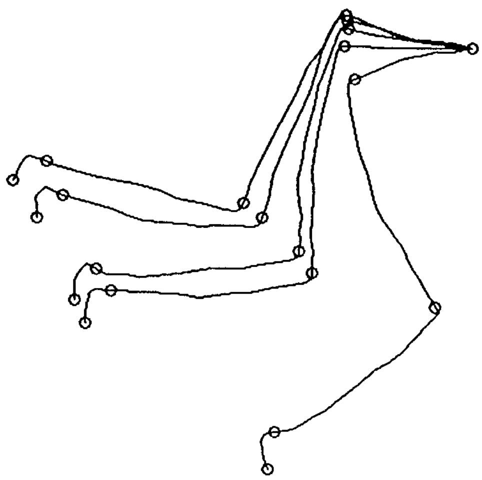

Figure 5 - These traces illustrate the odometrically perceived path of the robot on 5 consecutive runs along the same path. Notice that the wiggles along each segment are different since obstacles were moved and doors were altered between runs. To the symbolic system, however, all runs seemed identical.

4: Discussion

The SSS architecture is related to a number of recent projects in robot control architectures. The greatest degree of similarity is with the ATLANTIS system developed at JPL [11, 9]. In this system there is a subsumption-style “control" layer, an operating system-like “sequencing" layer, and a model-building “deliberative" layer. As in our navigation scheme, this system delegates the task of following a particular path segment control to a behavior-based system while using a rough topological map to specify the turns between segments. A mobile robot developed at Bell labs [15] also uses this same task decomposition as does the AuRA system [1] from UMASS Amherst.

However, in ATLANTIS (and in RAPs [8], its intellectual predecessor) each of the behaviors in the subsumption controller is associated with some "goal" and can report on its progress toward achievement of this objective. A control system developed at Hughes [13] also requires a similar signalling of process "failures". The problem with this type of system is that detecting true failures at such a low level can be difficult. In our system, external occurrences, not the internal state of some subsidiary process, determine when the behavior-based system is reconfigured.

ATLANTIS also places much more emphasis on the sequencer layer than we do. In ATLANTIS the deliberative layer essentially builds a partially ordered "universal plan" [14] which it downloads to the sequencer layer for execution. Similarly, one of the control systems used on HILARE [12] generates off-line a "mission" plan which is passed to a supervisor module that slowly doles out pieces to the "surveillance manager" for execution. In such systems, the symbolic system is completely out of the control loop during the actual performance of the prescribed task. This exclusion allows for only simple fixes to plans and makes it difficult to do things such as update the traversability of some segment in the map. In contrast, the contingency table in the SSS system only decouples the symbolic system from the most rapid form of the decision making – the symbolic system must still constantly replan the strategy and monitor the execution of each step.

In summary, with our SSS system we have attempted to provide a recipe for constructing fast-response, goal-directed robot control systems. We suggest combining a linear servo-like system, a reactive rule-like system, and a discrete-time symbolic system in the same controller. This is not to say a good robot could not be built using just one of these technologies exclusively. We simply believe that certain parts of the problem are most easily handled by different technologies. To this end we have tried to explain the types of interfaces between systems that we have found to be effective. To summarize, the upward sensory links are based on the temporal concepts of situations and events, while the downward command links are based on parameter adjustment and setpoint selection.

The SSS architecture has been used for indoor navigation and proved quite satisfactory. Developing a robot which moves at an *average* speed of 32 inches per second (the peak speed is higher), but which can still reliably navigate to a specified goal, is a non-trivial problem. It required using a subsumption approach to competently swerve around obstacles, a symbolic map system to keep the robot on track, and a number of servo controllers to make the robot move smoothly.

We plan to extend this work to a number of different navigation problems including the traversal of open lobbies, movement within a particular room, and outdoors patrol in a parking lot. We also intend to use a similar system to acquire and manipulate objects using a larger mobile robot with an on-board arm.

References

[1] Ronald C. Arkin, "Motor Schema Based Navigation for a Mobile Robot", *Proceedings of the IEEE Conference on Robotics and Automation*, 264–271, 1987.

[2] Rodney Brooks, "A Layered Intelligent Control System for a Mobile Robot", *IEEE Journal Robotics and Automation*, RA-2, 14-23, April 1986.

[3] Rodney Brooks, "Intelligence without Representation", *Artificial Intelligence*, vol. 47, 139–160, 1991.

[4] Jonathan H. Connell, "Navigation by Path Remembering", *Proceedings of the 1988 SPIE Conference on Mobile Robots*, 383–390.

[5] Jonathan H. Connell, *Minimalist Mobile Robotics: A Colony-style Architecture for a Mobile Robot*, Academic Press, Cambridge MA, 1990 (also MIT TR-1151).

[6] Jonathan H. Connell, "Controlling a Robot Using Partial Representations", *Proceedings of the 1991 SPIE Conference on Mobile Robots*, (to appear).

[7] Jonathan H. Connell and Paul Viola, "Cooperative Control of a Semi-autonomous Mobile Robot", *Proceedings of the IEEE Conference on Robotics and Automation*, Cincinnati OH, 1118–1121, May 1990.

[8] R. James Firby, "An Investigation into Reactive Planning in Complex Domains", *Proceedings of AAAI-87,* 202–206, 1987.

[9] Erann Gat, "Taking the Second Left: Reliable Goal-Directed Reactive Control for Real-world Autonomous Mobile Robots", Ph.D. thesis, Virginia Polytechnic Institute and State University, May 1991.

[10] Maja J. Mataric, "Environment Learning Using a Distributed Representation", *Proceedings of the IEEE Conference on Robotics and Automation*, 402–406, 1990.

[11] David P. Miller and Erann Gat, "Exploiting Known Topologies to Navigate with Low-Computation Sensing", *Proceedings of the 1991 SPIE Conference on Sensor Fusion*, 1990.

[12] Fabrice R. Noreils and Raja G. Chatila, "Control of Mobile Robot Actions", *Proceedings of the IEEE Conference on Robotics and Automation*, 701–707, 1989.

[13] David W. Payton, "An Architecture for Reflexive Autonomous Vehicle Control", *Proceedings of the IEEE Conference on Robotics and Automation*, 1838–1845, 1986.

[14] Marcel Schoppers, "Universal Plans for Reactive Robots in Unpredictable Environments", *Proceedings of IJCAI-87*, Milan Italy, 1039–1046, August 1987.

[15] Monnett Hanvey Soldo, "Reactive and Preplanned Control in a Mobile Robot", *Proceedings of the IEEE Conference on Robotics and Automation*, Cincinnati OH, 1128–1132, May 1990.

[16] Rüdiger Wehner, "Matched Filters - Neural Models of the External World", *Journal of Comparative Physiology*, 161:511-531.