SAUSAGES: Between Planning and Action

Jay Gowday

CMU-RI-TR-94-32

Carnegie Mellon University

The Robotics Institute

Technical Report

Technical Report

SAUSAGES: Between Planning and Action

Jay Gowdy

CMU-RI-TR-94-32

Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

September, 1994

This research was partly sponsored by DARPA, under contracts "Perception for Outdoor Navigation" (contract number DACA76-89-C-0014, monitored by the US Army Topographic Engineering Center) and "Unmanned Ground Vehicle System" (contract number DAAE07-90-C-R059, monitored by TACOM), and partly sponsored by NSF under NSF Contract BCS-9120655, titled "Annotated Maps for Autonomous Underwater Vehicles."

Carnegie Mellon

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3890

15 December 1994

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314

RE: Report No. CMU-RI-TR-94-32

Permission is granted to the Defense Technical Information Center and the National Technical Information Service to reproduce and sell the following report, which contains information general in nature:

Jay Gowdy
SAUSAGES: Between Planning and Action

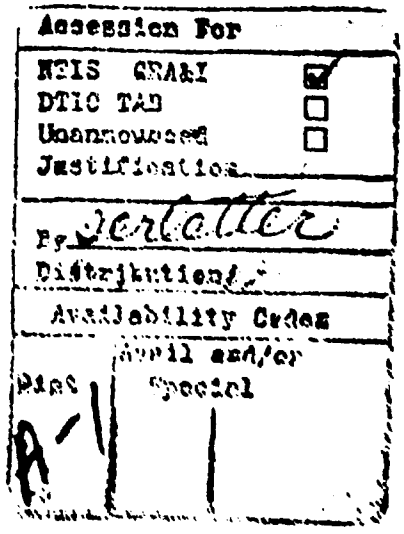
Yours truly,

Marcella L. Zaragoza

Marcella L. Zaragoza
Graduate Program Coordinator

enc.: 12 copies of report

Table of Contents



List of Figures

Abstract

SAUSAGES stands for System for AUtonomous Specification, Acquisition, Generation, and Execution of Schemata. SAUSAGES provides a framework for specifying, running, monitoring, and altering plans for mobile robots, and is ideal for building complex real-time systems that need to operate outdoors. SAUSAGES bridges the gap between planning and action, between the world of symbols and propositions and the world of pixels and actuators. On the one hand, system designers whose primary interest is perception can easily build a SAUSAGES system capable of carrying out complex missions that showcases their perception modalities and actuation modules without a lot of resources devoted to higher level task management and control. On the other hand, system designers who are primarily interested in planning can use SAUSAGES to provide a layer of abstraction for high level planners so that they do not have to worry about the implementation details of each plan step. This allows for the natural development of a robot that can act as an intelligent agent in the unstructured outdoor environment rather than just in simulation or in a highly structured laboratory environment.

1 Alternatives to SAUSAGES

Several years ago, it became apparent that some form of mission execution and monitoring was needed to integrate the capabilities of the perception systems developed at Carnegie Mellon for the Navlab, a computer controlled Chevrolet van[10]. We had road followers that robustly followed roads[6], object detectors that avoided obstacles[9], landmark recognizers that localized the vehicle position in a map[3], but we had no consistent architectural glue to join them together. A robust road follower is impressive on its own, but a road follower alone has no way to know which way to turn at an intersection, no way to know when to speed up or slow down for important events, etc. A system that can execute a complex mission cannot simply be the sum of its perceptual modalities, there needs to be a "plan" which uses high level knowledge about goals and intentions to direct the behaviors of the low level perception and actuation modules.

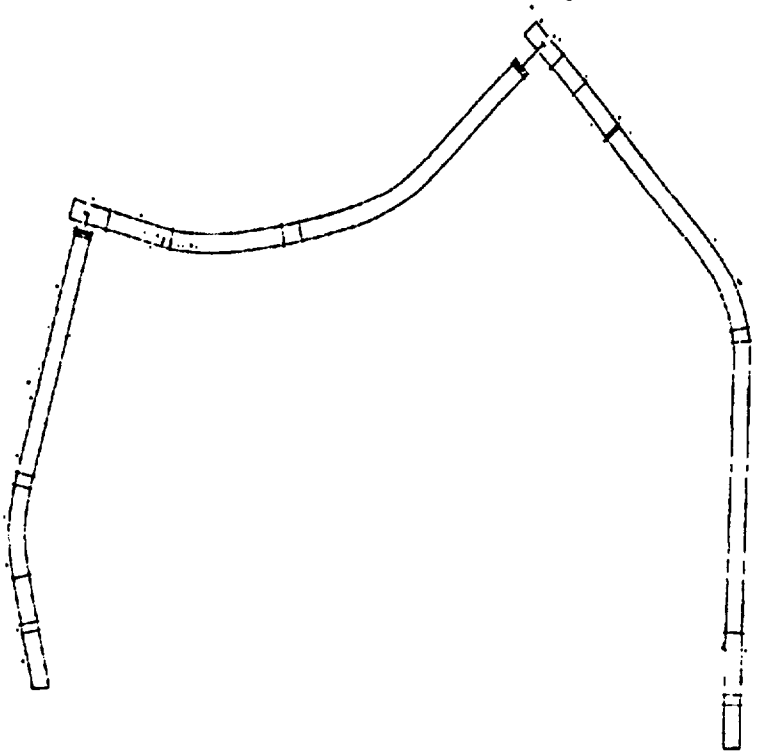
An intuitive way to represent a plan and execute it is through events, e.g., "Follow this road until the second intersection, turn left, and stop at a mail box." This plan representation works well for simulated and indoor mobile robots, which operate in a highly structured and controlled environment where every event necessary to the plan execution is easily perceived. Unfortunately, once a robot system leaves the laboratory it is in a much less structured, less controlled, and less predictable environment. The events that an outdoor mobile robot system needs to detect to execute a plan are very difficult to discriminate from the background environment. Reliable and accurate acquisition of perceptual cues, such as intersections or signs, is an almost impossible task. A system that relies solely on these perceptual events to direct its plan is not feasible.

To get around this problem and still have a system that could execute interesting missions we developed Annotated Maps[11]. An Annotated Map system reduces the problem of monitoring for arbitrary events, such as intersections or signs, to the problem of monitoring the vehicle position on an a priori map. The plan in an Annotated Map is essentially a positional production system, with productions like "at position x, perform action y" instead of productions like "if event x then perform action y." This alternate representation of plans vastly reduces the computational resources necessary to perform a mission, while still allowing complex scenarios.

We have extensively tested this system. For example, we used it to drive the Carnegie Mellon Navlab from one house in a suburb north of Pittsburgh to another approximately one kilometer away. The system used neural networks to drive the vehicle along the roads and used a laser range finder to detect obstacles and landmarks. The system successfully performed the mission, speeding up and slowing down at the appropriate places, navigating through several intersections, and coming to a stop at the proper mailbox.

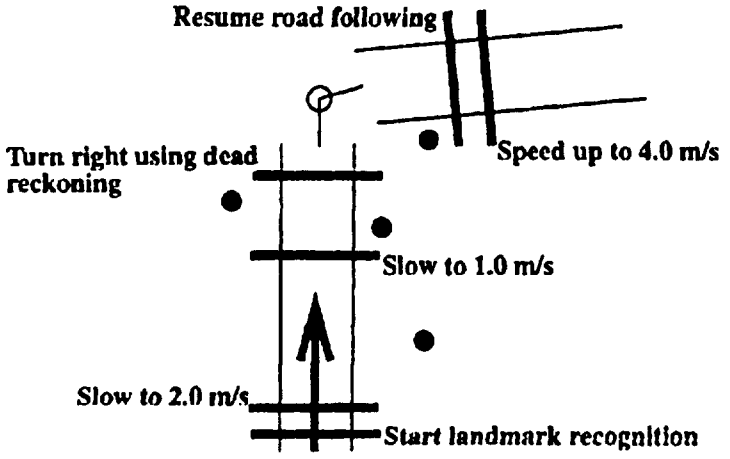
Figure 1: shows the annotated map we used in this system. We built this map by driving the vehicle over the course, storing the road position received from the controller and the landmarks detected by the laser range finder. A human expert then went through and added "annotations," or trigger lines. When the vehicle crosses these lines various actions are taken, such as setting speed, or turning modalities on or off. Thus, these trigger lines implicitly specify the plan.

Figure 1: A sample annotated map



The process of navigating through an intersection shows how this simple geometric implicit plan specification works. We had no perception system that allowed us to navigate an intersection, so it had to be done using only the map, and thus the vehicle's position had to be accurate. Figure 2: gives an example of this process. As the vehicle's position is tracked going down the road, the first trigger line that it crosses will turn on landmark based position updating to pinpoint the vehicle's position on the map for precise navigation. Then the vehicle will cross a series of lines that will slow it down in stages. Once the vehicle actually enters an intersection, it crosses a line which turns off road following and turns on dead reckoning from the map to actually make a left turn. Once the vehicle leaves the intersection it crosses lines that turn road following back on, turn landmark based position updating off, and sets the vehicle speed back to maximum again. Each step in this "plan" is sequenced implicitly by the positioning of the trigger lines and assumptions about the motion of the vehicle.

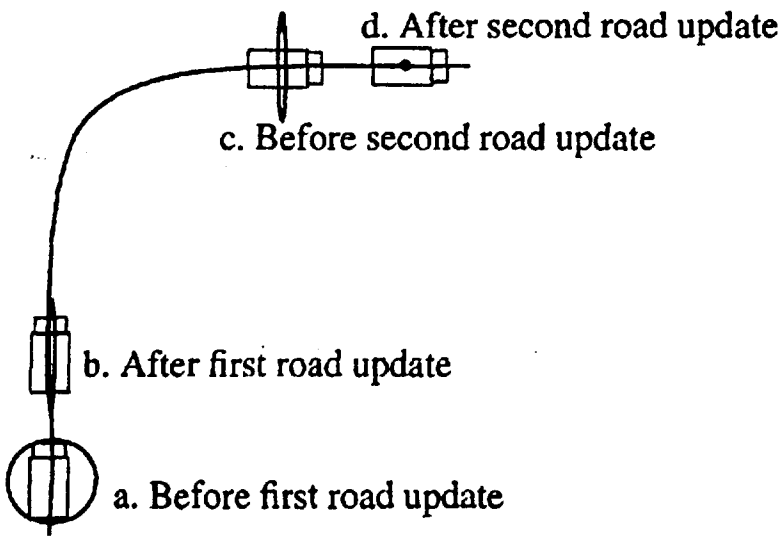
Figure 2: Navigating an intersection with an Annotated Map



There are several possible objections to using a global coordinate system with an absolute map. What if the map is wrong? What if the positioning system drifts? We have found that these are not valid objections if there are landmark recognition systems that can guarantee that the map is locally correct during the execution of the mission. It does not matter if the map is wrong globally as long as the region currently around the vehicle is roughly correct. What is important is that the control information encoded by the trigger lines is tied geometrically to actual features in the map, such as roads or intersections. If the position of the vehicle relative to a local landmark is accurate, then the position of the vehicle relative to the control information will be accurate, and therefore the plan will be accurately executed.

Having these landmark recognition systems is not much of an additional expenditure of resources since the perception systems needed to navigate the vehicle can double as sources of information about landmarks. For example, obstacle avoiders can double as discrete landmark detectors. Even road followers can be used to pinpoint the vehicle's position in a map with surprising ease and precision. As Figure 3: shows, a road follower can reduce the uncertainty in the vehicle's position perpendicular to the axis of the road. If the vehicle rounds a corner the cumulative updates in the vehicle's position on the map will reduce the uncertainty of the vehicle's position dramatically[12].

Figure 3: Using a road follower to update position on a map.



Our philosophy in building outdoor mobile robot systems is that perception is the primary task, i.e., the task that the most computational and developmental resources will be expended on. Planning has been considered secondary, since it is a much "easier" problem. Annotated maps are an example of this philosophy taken to an extreme. Annotated Maps allow us to showcase our perception and actuation modules in complex scenarios with a minimum of resource investment in the event detection and planning issues.

This approach is in direct contrast to much of the previous work in the field in that many researchers consider the interesting part of their system to be the planning, monitoring, and error recovery capabilities. The perception systems are an afterthought to showcase the abilities of the planning system. This can be seen in the plethora of indoor mobile robots and simulations found in the literature in which the environment can be structured and controlled to the advantage of the plan generation, execution, and monitoring systems.

Either extreme has obvious limitations. A system that reasons very robustly and intelligently about events that it can't robustly detect in the real world will be an interesting system with no real application. At the other extreme, a system that only uses Annotated Maps for mission planning will rigidly perform one preplanned mission, but won't react well to unexpected changes in the plan, goals, or environment.

For example, while an Annotated Maps system can handle repeated, well established scenarios, such as mail delivery or a daily commute, it cannot begin to execute a simple, non-geometric plan for the first time, such as, "go to the first corner and turn left." For every mission there has to be an a priori, detailed map and a priori expert knowledge of how to navigate around in that map. Another major limitation of an Annotated Maps "plan" is that it is only implicit, i.e., the sequencing of the "steps" of the plan are implicit to the geometric placement of the annotations. The Annotated Map manager itself has no concept of "plan", just of the map and the vehicle position. This makes it difficult to execute missions with loops in the plan without bending, if not breaking, the paradigm. It also makes it hard to monitor for failures of the plan. How is the map manager supposed to detect a failure in the plan when it doesn't even explicitly know what the plan is?

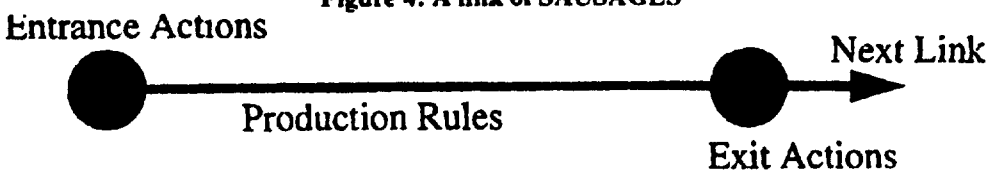
Our experience building real outdoor systems shows that we need a high level plan executor/monitor to integrate our various perception and actuation modules into coherent systems. We also found that any such high level knowledge system needs to reflects the realities of building an outdoor mobile navigation system: What the system can do is driven primarily by what it can perceive, not what it can plan. At the same time any plan executor/monitor has to be much more flexible and reactive than Annotated Maps. To satisfy these needs we built SAUSAGES.

2 Description of SAUSAGES

SAUSAGES stands for System for AUtonomous Specification, Acquisition, Generation, and Execution of Schemata. SAUSAGES is designed to be a bridge between the worlds of planning and perception. It can be used to integrate perception modules together into useful systems with a minimum of planning work, and it is easy to use with external planners that can generate, monitor, and adjust SAUSAGES plans to create intelligent and reactive behavior in the face of changing environments and goals.

A SAUSAGES plan is made of discrete semantic units that we call "links." While in an Annotated Map the sequencing of the plan is implicit, in a SAUSAGES plan the sequencing is explicit. Where an annotated map plan is a geometric map with annotations, a SAUSAGES plan is a directed graph of links.

Figure 4: A link of SAUSAGES

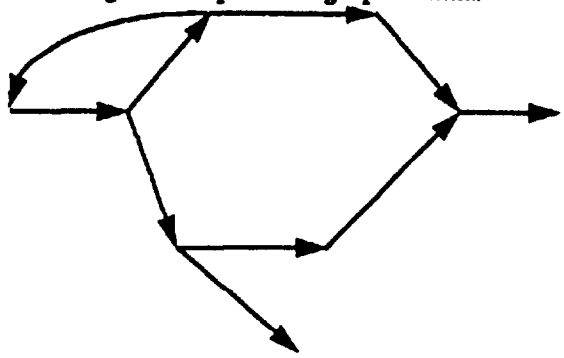


A link can be thought of as a single step in a plan. A link has entrance actions that get invoked when the link starts up, and exit actions to clean up when the link exits. The link has a set of production rules associated with it that are added to a global production rule system. These production rules have actions that specify when the link is finished, when it has failed, side affects of the link, or whatever the system designer wants them to specify. These production rules are only in effect while the link is active.

SAUSAGES uses a Link Manager process that manages the execution of links. It is the Link Manager that maintains the production rule system and starts and stops links. Most interaction with a SAUSAGES plan is done through the Link Manager by setting state variables that fire various production rules. The Link Manager also lets external modules remotely add links, run them, delete them, and even change them.

A SAUSAGES link also has a function that returns the next link to execute when it is finished. This is how a planner hooks together a series of links into a plan. This function can be as simple as returning another link, which might be part of a serial chain of links, or it could be conditional, where if one condition holds activate one link and if another holds activate another. This allows a plan writer to construct plans that are directed graphs of links rather than just a serial chain. Thus a planner can create a plan that has some intelligence and decision making built into it to handle common contingencies that might be foreseen.

Figure 5: A plan as a graph of links

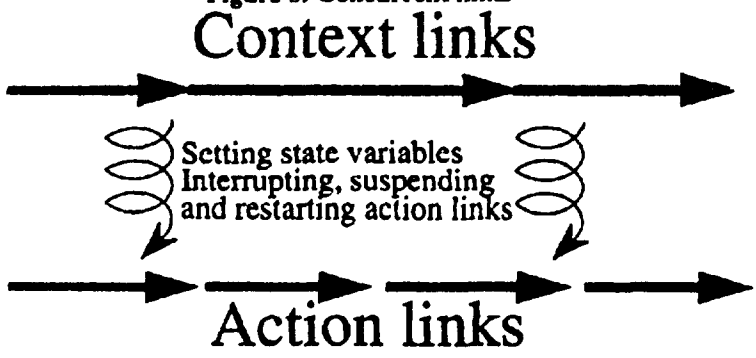


Links can be composed of sub-links, to form hierarchical plans. This is a common abstraction in plan descriptions. A link which implements "go to airport", will decompose into several sub-links such as, "get in car," "drive to airport," and "park at airport." Each of these links could have their own production rules with completion and failure conditions attached. The plan designer can specify that any failure be handled entirely at the level of the link that failed, or that failures get propagated up the link hierarchy.

SAUSAGES plans do not have to be "plans" in the traditional sense. They can simply be algorithms. An external planner can produce an algorithm for a situation, complete with looping or recursion, download it to SAUSAGES, and run it. We get this flexibility because the "next link" function is arbitrary, and instead of producing a predetermined link it could be a function that generates a new link based on the current link or any condition that is relevant. Thus SAUSAGES "algorithms" can be recursive and dynamic rather than simply iterative and static.

The plan graph can have more than one link active at a time. This way the Link Manager can simulate multiple threads of control. One common example for mobile robots is to have two threads of control, an action thread and a context thread. The action thread will control the robot motion and perception, while the context thread will monitor for global errors or major changes in the plan. The context thread can affect how the action thread transitions. The context thread can start and stop links in the action thread, or replace links in the action thread entirely.

Figure 6: Concurrent links

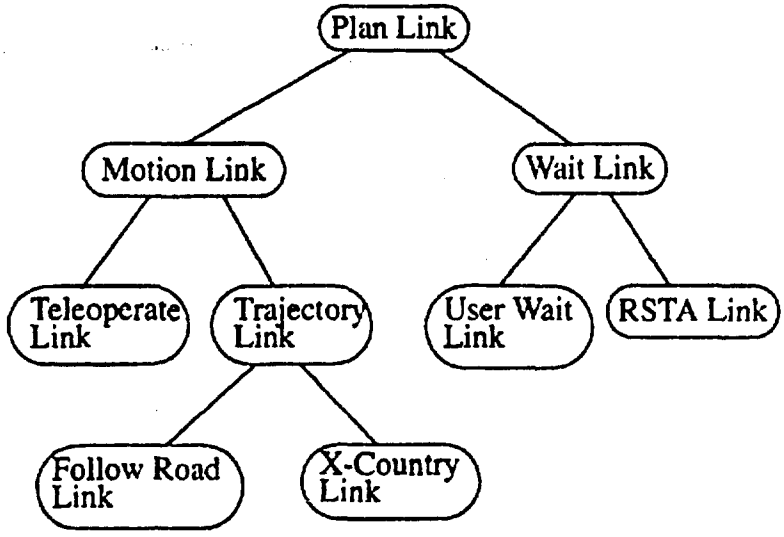


So, SAUSAGES provides a planning language that is well suited for specifying and executing mobile robot plans. These plans can range in complexity from a simple linear linking together of different steps in a plan to actual algorithms including looping and recursion. A great advantage of the SAUSAGES approach is that plans are specified in semantically meaningful units which can be easily manipulated by external planners and monitors in useful ways.

3 Implementation of SAUSAGES

Although SAUSAGES is written in C++, SAUSAGES plans, links, and interface functions are specified using a C++ lisp interpreter called JayLisp. The most notable variation in JayLisp is that it has a simple class system which can be especially useful in specifying types of "links" for the Link Manager. For example, there can be a "trajectory" link class that follows a trajectory and contains much of the administrative work for following trajectories. Then subclasses of the trajectory class can be specified that follow roads or go cross country. At present, JayLisp only allows for inheritance of data and methods from a single parent class, but Figure 7: shows how even single inheritance can create a useful class hierarchy.

Figure 7: The UGV link class hierarchy



Another key feature of JayLisp is that it is trivial to write new C++ functions and to invoke them from JayLisp. This extensibility supports our philosophy of using JayLisp as an interface and doing any computationally expensive work in C++. This plays to the run-time strengths of both languages: JayLisp's flexibility and C++'s efficiency.

Using a Lisp language as the interface to SAUSAGES allows great flexibility in specifying a plan or algorithm. Link entrance and exit actions are naturally specified by JayLisp expressions. Arbitrarily complex Lisp expression's can be used in specifying "rule" criteria. Classes can be used as link templates, so a "follow-road" link is just an instance of the follow road link class with the appropriate data slots filled in and the remaining data slots filled with default values.

The most important module in SAUSAGES is the Link Manager, which actually manages the execution and monitoring of links. Link networks, i.e. plans, that the Link Manager will run can be read in from files or transmitted from other modules via a TCP/IP connection.

In order to run links and affect their behavior the Link Manager maintains a simplified production rule system. Each rule in the system has a list of tags, a criteria expression (the "left hand side" of the rule), and an action expression to evaluate when the criteria expression evaluates to non-nil. A rule's criteria expression is only evaluated when one of its tags is marked as changed. This is not a complete forward propagating rule system, but it has proven sufficient for the purposes of mobile robot planning. If a complete production system is needed it will be easy to add and integrate. The Link Manager also maintains a global timer table which maintains a list of expressions to be evaluated at regular intervals

These global systems for productions and timers are the means by which a SAUSAGES plan can seem to be multi-threaded. When a link is run, the Link Manager adds the link's rules and timers to the global sys-

tems and when the link finishes they are removed. If two links are active "concurrently" both links' rules and timers are active in the global systems.

External modules can easily send the Link Manager expressions to be evaluated, and thus can easily affect the firing of rules in the global production system. The remote expression evaluation provides one method for an external module to create and run links remotely. The Link Manager also provides methods for exporting references to the links themselves so that external modules can modify them as needed. The Link Manager can also send messages to external modules just as easily as it gets them. These are just some of the ways a SAUSAGES system can interact with external perception modules and plan monitors.

The only kind of events monitored in an Annotated Maps systems were based on the position of a vehicle on a map. We could write these map monitoring capabilities directly into the Link Manager using JayLisp, but we find it more convenient to delegate this processing to a separate module called the Navigator. The Navigator has two jobs:

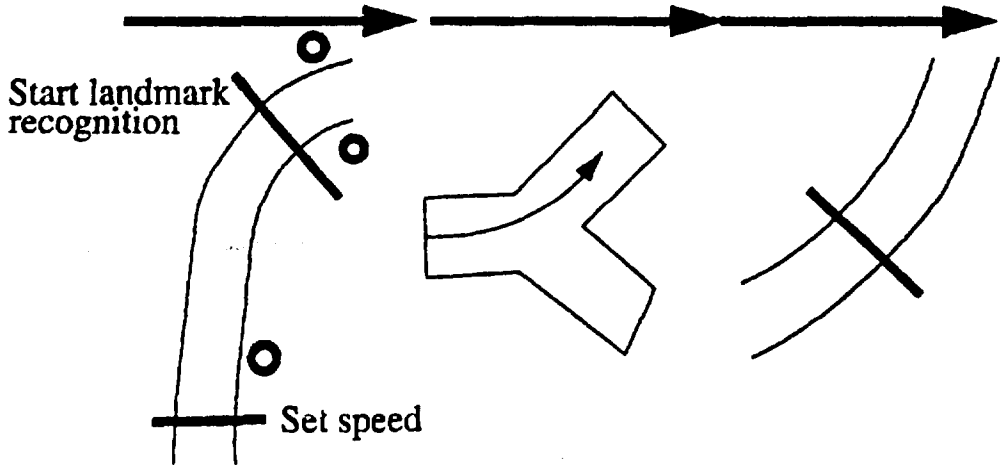
1. Maintaining a transform between the position that the vehicle controller reports and the position on a map.
2. Handling client queries about the vehicle's position relative to objects in the world. Clients can register a geometric query and the Navigator will inform the client when the vehicle has moved to a position that triggers that query. This query mechanism can be used for such purposes as monitoring the crossing of trigger lines and checking to make sure the vehicle stays within a tolerance distance of a trajectory.

3 SAUSAGES and Annotated Maps

It is obvious in SAUSAGES how to represent event based plans: Each link terminates on the occurrence of some event, such as "intersection detected," or "Fred's mailbox found." Unfortunately, this brings up the original problem with this type of plan representation for outdoor mobile robots: Detecting the arbitrary events necessary to proceed from plan step to plan step. We can easily solve this problem by using segments of Annotated Maps as individual links.

In the simplest possible annotated map link, the only production rule would be, "If vehicle X crosses the line (x1, y1), (x2, y2), end the link and go to the next one." The next link would just be the next segment of the Annotated Map. Stringing links of this type together exactly simulates the capabilities of our current Annotated Map system, with the plan made more explicit.

Figure 8: Using Annotated Maps as SAUSAGES links

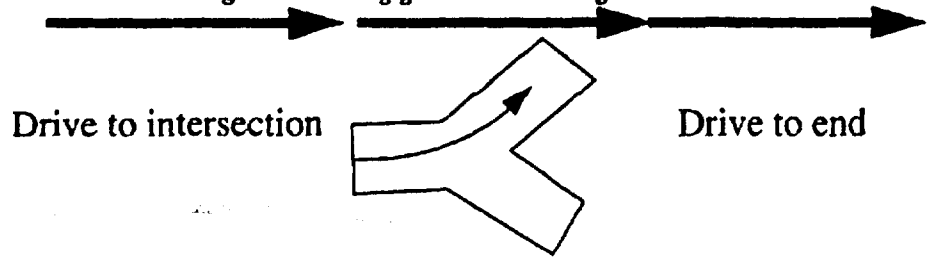


Using SAUSAGES to make an Annotated Maps plan explicit frees a system from many of the limitations of Annotated Maps while retaining the central advantage of Annotated Maps: the reduction of the event detection problem to a position monitoring problem. For example, retrotraversing a trajectory was difficult with Annotated Maps since the system had to discriminate between the annotations relevant to the journey out and the annotations relevant to the journey back. SAUSAGES easily solves this problem by allowing two annotated map links with the same geometric information about the world, but with different annotation data for each leg of the traversal. So each link has the same road and landmark data but with different control information.

Since each step of the plan is an explicit link, the user can attach explicit and different failure conditions to each step. Thus we can easily and flexibly specify the conditions under which a plan fails and what to do when the plan fails. For example, in a road following annotated map link we would want a failure condition to be "if the vehicle position is too different from the road position, fail." For a cross country annotated map link we would want the same condition, but with a higher tolerance. This flexibility in specifying the failure conditions is trivial to implement in a SAUSAGES plan because each class of link can have a different set of production rules that indicate errors. With Annotated Maps alone implementing even this simple error monitoring requires adding code that does not fit the design well.

All links are equivalent to SAUSAGES, so it is trivial to mix "geometric" or map based links with "cognitive" or event based links. A SAUSAGES plan is not locked into either paradigm and provides extensive support for both. This allows planners to easily generate plans that can both retraverse known territory using map based links and explore new territory using event based links, assuming that event detectors can be built.

Figure 9: Mixing geometric and cognitive links

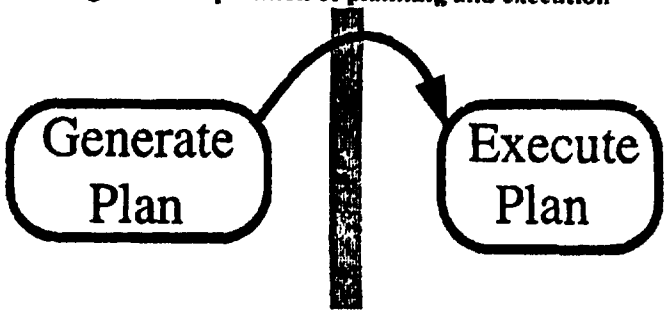


4 SAUSAGES and planners

SAUSAGES is not a planning system. It is a plan executor and monitor that stands between a planner and the real world, between the world of symbols and propositions and the world of pixels and actuators. It depends on external sources for its plans, whether that be a human hand coding a mission plan for a specific demo or a sophisticated AI planner generating plans as it responds to changes in environment and goals.

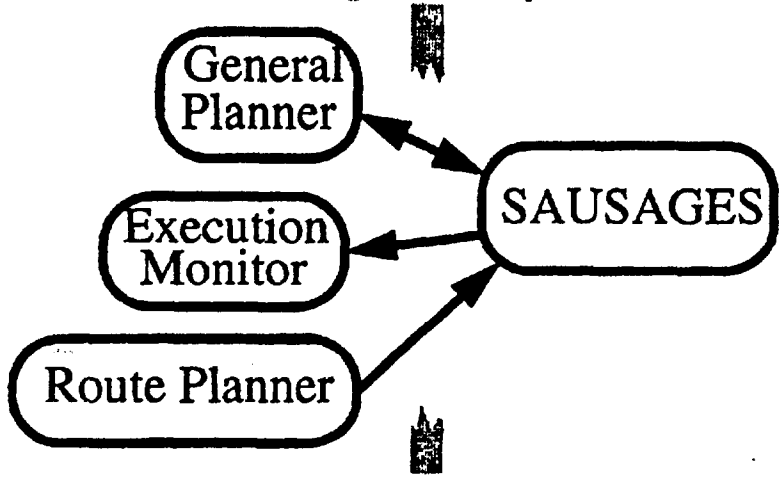
One of the historical models of planning involves a separation of planning and execution[5]. A planner comes up with a complete plan, sends it to an executor which does its best to perform the plan, and reports success or failure to the planner. In this paradigm there is a conceptual wall between planning and execution, and many researchers have shown the limitations of this approach. Is SAUSAGES a step backwards to this paradigm?

Figure 10: Separation of planning and execution



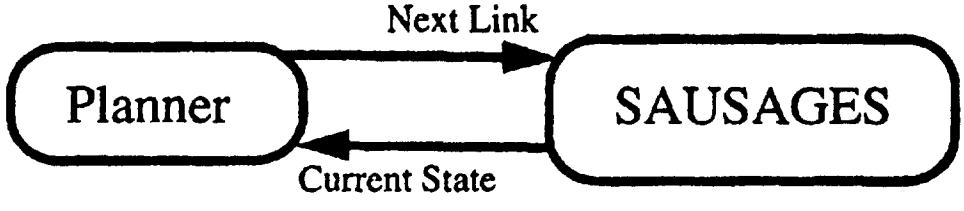
It is easy to construct SAUSAGES systems using this paradigm, but SAUSAGES is not limited to it. SAUSAGES was designed for heavy interaction with any number of external planners, not just one omniscient planner. External modules can create, edit, and run links. Links can be designed to report status back to external modules and can have data associated with them that is not necessary for plan execution, but which do encode information necessary to manipulate the plan.

Figure 11: SAUSAGES interacting with external planners and monitors



As we have seen, a SAUSAGES plan does not have to be a passive, static graph, since the "next link" function can create new links at run time. The relevant case for planning is a next link function that generates a call to an external planner asking what to do now. The most extreme form of this approach would be to make every link's next function report status back to an external planner and request that the planner generate the next link.

Figure 12: A SAUSAGES system that completely interleaves planning and execution



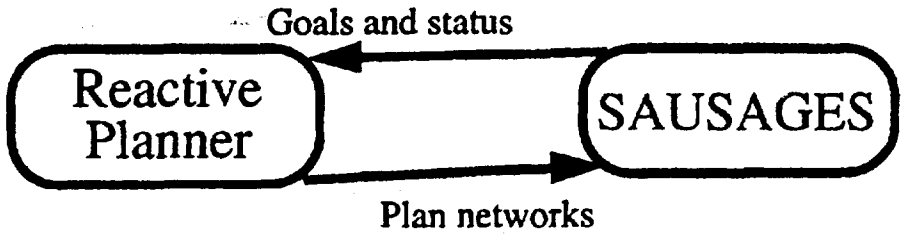
This approach, which totally interleaves planning and execution, can be used to interface a SAUSAGES system with incremental planners such as the Procedural Reasoning System (PRS)[2] or the universal planner approach[7]. At each step in a plan a incremental planner examines everything it knows about the situation before applying the next action. This planning paradigm lets the system easily respond to errors, system failures, goal changes, and other unexpected changes in the world. It also lets the "plan" be mostly

incomplete and underspecified. Using this planning/execution paradigm in a SAUSAGES system essentially reduces the role of SAUSAGES to a "mode manager," i.e., the planner spits out a high level command such as "follow the road using system X" which SAUSAGES translates into actual commands to perception and actuation modules. SAUSAGES is also used in such a system to translate failures of perception and actuation modules into symbology that the incremental planner can process.

While this approach is robust and has been extensively tested, we do not feel that it is the proper approach to use in a SAUSAGES system. It throws away many of the abilities of SAUSAGES in order to have everything controlled by a single monolithic planner. We find in building systems that putting too much of an emphasis on general purpose modules results in degraded real time performance. It is much better to have multiple specialized systems, with each system doing its job efficiently and robustly. SAUSAGES is not a general purpose module: its specialization is as a plan executor and monitor, and any computationally expensive job should be delegated to an external module. The "incremental plan" scenario just does not take advantage of the things that SAUSAGES does well.

A better approach for the ultimate SAUSAGES system is to still use something very much like PRS, but instead of generating single plan steps the system would generate algorithms and plan sequences that are appropriate for achieving the goals of the mission. This plan graph sent to SAUSAGES would incorporate the necessary information to monitor and to detect failure. Any failures would be handled by reporting to the Planner the new state of the world so that it could generate a new algorithm or plan sequence to achieve the goals of the system with the new knowledge of the failure.

Figure 13: The ultimate SAUSAGES intelligent agent



This approach takes advantage of the "reactive" abilities of an incremental planner like PRS while still exploiting the efficiency and flexibility of SAUSAGES. The question of which system is dominant, the incremental planner or the SAUSAGES plan executor, is irrelevant. It doesn't really matter whether there is a SAUSAGES plan network that invokes the incremental planner which in turn generates SAUSAGES programs or if there is an incremental planner which generates a SAUSAGES program which calls out to the planner which in turn generates more SAUSAGES programs.

A human problem solver executing a task does not typically think about each step. A human will see a goal to achieve and will try and find a schema or algorithm that achieves that goal. Typically, a human problem solver will only realize that an algorithm is not appropriate when it fails, and only then will the slow, thoughtful step by step "planning" to achieve the goal or work around it be tried. This behavior, known as Einstellung effect[4], corresponds well to a system that uses an incremental planner to choose SAUSAGES plan networks. The incremental planner has a goal, and to achieve that goal it generates a SAUSAGES network. The SAUSAGES network runs until it finishes or fails. If the network successfully finishes, the planner can go on to working on the next goal. If the network fails, the planner can then go into a more intensive step by step incremental planning mode, or it can take whatever other course of action it deems necessary.

5 Conclusions

SAUSAGES allows incremental development of real outdoor mobile robot systems. Groups working primarily in perception and actuation can use powerful constructs such as annotated maps to quickly generate

systems that perform interesting missions with a minimum of work in high level planners. Groups working primarily in planning and artificial intelligence can use SAUSAGES to interact with a real robot on the symbolic level rather than the pixel and motor level.

It is easy to see how SAUSAGES allows the development of a complete outdoor mobile robotic system that is driven by what the robot can perceive rather than what the planners want to perceive. As the perception and actuation modules improve, the developers create SAUSAGES links and algorithms to perform various tasks. When a planning system matures to the point of being integrated with a real robot it can use these various SAUSAGES links and algorithms that have been heavily tested in the field as building blocks for plans.

Another advantage of SAUSAGES is that it does not lock the system designer into any single planning or architectural paradigm. It is simply a bridge between planning and action, and does not specify how either is to be accomplished, or if either is to happen at all. SAUSAGES lets the system designer vary the level and extent of involvement of a general planner in the execution of a plan.

For example, suppose that one leg of a mission has a pair of vehicles performing a "bounding overwatch." A bounding overwatch is a maneuver in which two vehicles mutually support each other as they travel under the possible observation of an "enemy." A bounding overwatch cannot be completely specified as a set of trajectories and mode transition points, since the exact execution depends on a large number of environment variables and constraints that can not be determined a priori. A SAUSAGES system could implement a bounding overwatch by having a general purpose reactive planner keep track of the current goals and states of the vehicles and enemy. This planner would generate new links one by one as the bounding overwatch progresses. Alternatively, if a system designer can design a bounding overwatch algorithm it would be more efficient to have a specialized bounding overwatch module implement this algorithm. In this case, the higher level planner just generates a link, "do a bounding overwatch," which SAUSAGES interprets as "ask the bounding overwatch planner to do a bounding overwatch." If the algorithm for doing a bounding overwatch is simple enough, the bounding overwatch module could be completely dispensed with, and the bounding overwatch could be done through an algorithm implemented as a SAUSAGES network. With SAUSAGES, any of these could easily be implemented, and it is up to the system designer to decide which is best.

Our philosophy in building SAUSAGES systems is to avoid generalization. Instead of having one monolithic general planner that handles all situations completely, it is better to set up a planning system that has a general purpose planner that generates links that call domain specific planners that are much more efficient at planning in their domain. There is still a place for a general high level planner, but it should not be involved at all levels of decisions. It is a much better approach to use the most efficient representation and solution methods to achieve a goal, and then have a well defined way of combining the results of these methods. SAUSAGES is that well defined way.

This philosophy comes from a desire to avoid bottlenecks in knowledge and communications. Simmons claims that having an omniscient central module that controls everything and that all data must flow through makes monitoring and control easier[8], but it reflects unrealistic assumptions about bandwidth and real time response to environmental changes. SAUSAGES appears to reside in the center of a system, between planners and perception modules, but it does not suffer from the bottleneck problem. In a properly designed SAUSAGES system, the modules that keep the mobile robot alive and that depend on real time responses are only being given advice by SAUSAGES and are not depending on SAUSAGES for real time commands.

For example, SAUSAGES will be giving commands like, "start road following," and "try to follow this trajectory." The primary job of the road followers and cross country navigators is to keep the robot safe, the secondary is to follow the advice. It is up to SAUSAGES to monitor the progress of the modules and to give them better advice if the advice turns out to be bad. The last things these modules should depend on for real time response is the SAUSAGES planner and its ancillary high level planners. This approach is similar to Agre and Chapman's idea of plans as communication rather than programs[1].

Systems can be designed in which SAUSAGES is a bottleneck, but that never has to happen. For example, if there is a route planner that needs feedback from the terrain, and thus needs high resolution terrain maps from a perception system, this information should not be transmitted through SAUSAGES. The route planner could use SAUSAGES to set up a direct connection between the planner and the perception module, and then use that connection to communicate this high-bandwidth information. SAUSAGES is designed to act as a system traffic cop, and not the information highway itself.

SAUSAGES provides a module that can be the middle ground between the extremes of the robotics world. It allows a researcher working on a high level planner to avoid the awful implementation details of a mobile robot system by abstracting the operation of the lower level perception and actuation modules. It allows a perception researcher to easily build a system that showcases the perception work in a complex scenario without worrying too much about issues of planning and error recovery that are irrelevant to the task at hand. Finally, SAUSAGES facilitates the joining of these two extremes into a complete, intelligent, and robust outdoor mobile robot system.

References

[1] P. E. Agre and D. Chapman. What are plans for?. *IEEE Transactions on Robotics and Automation.* 617-34, 1990.

[2] M. P. Georgeff and A. L. Lansky. Procedural knowledge. *Proc. IEE Special Issue on Knowledge Representation*, pages 1383-1398. 1986.

[3] M. Hebert. Building and navigation maps of road scenes using an active sensor. *Proc. Of Intern. Conf. on Robotics and Automation*, pages 1136-1142. IEEE Computer Society, 1989.

[4] A. S. Luchins. Mechanization in problem solving. *Psychological Monographs.* 54(248), 1942.

[5] N. Nilsson. *Shakey the Robot.* Technical Report Tech. Note 323, SRI, Menlo Park, CA, 1984.

[6] D. Pomerleau. ALVINN: An Autonomous Land Vehicle In a Neural Network. *Advances in Neural Information Processing Systems I.* In D. Touretzky, Morgan Kaufmann, 1989.

[7] M. J. Schoppers. Universal plans for reactive robots in unpredictable environments. *Proc. Int'l Joint Conf. on Artificial Intelligence*, pages 1039-1046. 1987.

[8] R. Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation.* 10(1):34-43, 1994.

[9] T. Stentz. *The NAVLAB System for Mobile Robot Navigation.* Ph.D. thesis, Carnegie-Mellon University, 1989.

[10] C. Thorpe. *Vision and Navigation: the The Carnegie Mellon Navlab.* Kluwer Academic Publishers, 1990.

[11] C. Thorpe and J. Gowdy. Annotated Maps for Autonomous Land Vehicles. *Proceedings of DARPA Image Understanding Workshop.* Pittsburgh PA, September, 1990.

[12] C. Thorpe, O. Amidi, J. Gowdy, M. Hebert, D. Pomerleau. Integrating position measurement and image understanding for autonomous vehicle navigation. *Second International Workshop on High Precision Navigation.* 1991.