# Sensor Planning for Reactive Robotic Systems

Gregory D. Hager and Gerhard Grunwald

Research Report YALEU/DCS/RR-925
October 1992

# Sensor Planning for Reactive Robotic Systems

Gregory D. Hager
Department of Computer Science
P.O. Box 2158 Yale Station
Yale University
New Haven, CT 06520

Gerhard Grunwald
German Aerospace Research Establishment
Institute of Robotics and Systems Dynamics
Münchenerstr. 20
D-8031 Oberpfaffenhofen

**Abstract**

Recently, there has been increased emphasis on employing reactive actions in robot task planning. The principle reasons for this change are to increase the robustness of robot actions by making them sensor controlled, and to accommodate dynamic, unpredictable environments. However, in many cases, supporting reactive mechanisms requires choosing task level sensor inputs for the reactive procedure. This paper addresses the issue of planning the task level sensing required to carry out a reactive robot program. A preliminary framework for planning is presented, and sensor planning is illustrated for the problem of replacing a mechanically attached plug in a space environment.

## 1 Introduction

*Reactive robotics* is an umbrella term for the philosophy that robotic tasks should be carried out using direct, task level sensor feedback. The efficacy of using task level sensor feedback is indisputable: task level feedback increases both the range and reliability of robot task performance. Unfortunately, the vagaries and complexities of processing data from sensors capable of task level sensing (most notably vision, dense and sparse range sensors, tactile sensors, and force sensors) has limited progress in this area. Thus far, most interesting systems employ rudimentary, "hard-wired" sensor inputs to accomplish simple, unstructured tasks such as wandering through corridors or collecting rocks on the surface of Mars [2, 6].

More recently, there have been attempts to employ reactive procedures in more structured tasks by posing task level planning as the configuration of generic reactive modules to accomplish specific objectives [5, 13, 14]. In principle, planning for reactive behavior can be decomposed into the following two subproblems:

- Analysis of the task and its decomposition into sensor-relative actions. This is, in many ways, similar to "classical" planning, however, it is important to deal with problems of realtime operation and concurrency [14].
- An analysis of the structures in the world that can be recognized and used to guide actions. The goal to reduce the required sensor information to the minimal, most reliable sensor inputs needed to perform the action.

We refer to the former as *task level planning* and the latter as *sensor level planning.* Intuitively, task level planning is concerned with the operational decomposition of the

task into more basic actions, and sensor level planning is concerned with satisfying the information needs of these actions. Although we have separated these two activities, there are clearly a number of interactions between these two modules that need to be understood.

In this paper, we further define the problem of sensor level planning, we propose a general procedure for accomplishing it, and we illustrate the planning algorithm on a benchmark problem. We assume an environment structured to the point that the sensor planning system can use simulation as a reliable means of testing and debugging sensor plans. We note that the amount of structure needed by sensor planning is still somewhat unclear. It appears to depend heavily on the type of actions that must be planned, as well as the sensor information required to support them. In our case, we are mostly interested in sensor planning to support space teleoperation. Thus, we attempt to infer or elicit sensor-based actions from user input, and build the sensing configuration required to support it. In general, space environments are well-structured, and as described below, it is often possible to build simulations of high-fidelity.

The remainder of this article is structured as follows. In the next section, we explore the basic structure of the planning problem and introduce our test environment and example problem. In Section 3, we describe the structure of the planning system and discuss how it would plan a sensor-guided motion. Finally, we conclude with a discussion of our current work.

# 2 Task Level Planning and Sensor Level Planning

For the purposes of this paper, we will refer to actions that are executed without any task level sensing as *open-loop*, and actions using sensing as *closed-loop*. The role of task level planning is to construct a robot program composed of primitive, open-loop and closed-loop actions that successfully achieves some objective. Henceforth, we will assume that the task level planner is able to recognize that some actions cannot be executed open-loop with enough precision to succeed, and that it can infer task-specific geometric goal conditions that must be satisfied in object relative coordinates. Any objects referenced by a task description are known to the system *a priori* and geometric models are available. Furthermore, depending on the structuring of the environment, the approximate position or pose of object may be known. Finally we assume that the task level planner has enough geometric information to decompose closed-loop actions into approximately straight line motions. That is, if moving from position $A$ to position $B$ involves moving around an intermediate object, then the planner should define intermediate positions (or sets of positions) so that the total motion is a series of approximately straight segments. This type of decomposition can be done using relatively coarse, nonparametric geometry such as the maps described by Elfes [4].

As stated above, the goal is to rely on as little calibration as possible, and to emphasize object-relative positions rather than robot-relative positions. So, for example, a robot motion to some point $A$ in the workspace involves describing the position $A$ *relative to some features of the environment* rather than in a global robot coordinate system. Hence, closed-loop actions lead planning away from a classical "robot-centered" definition of motions and toward one that is *object-centered.* This is similar in conception to the indexical-functional representation proposed by Agre and Chapman [1].

In order to carry out sensor-based actions, a robotics system must recognize and/or verify the existence and pose of the "landmarks" used to define relative positions, locate

specific features that can be tracked to control the manipulator, perform the action, and at the same time monitor for sensor inputs that indicate the desired goal position has been reached. We initially classify the type of sensor operations needed in each of these steps as a *static* sensor task or a *dynamic* sensor task. Static sensor tasks perform one-time measurements and calculations. These actions are purely informatory, and are usually added to a program to instantiate variables needed by later actions. For example, recognizing an object addresses the problem of matching sensor information with a model. This information can be used to relativize an action to a particular object, to verify expected initial conditions, or to aid in the search for specific object features needed to control an action. The sensor planning problem is to match the available sensor information with a matching or recognition procedure. Dynamic sensor tasks support control of *performatory actions*: actions that lead to actual motion of the robot. For example, approaching an object requires tracking lines or patterns that can control the attitude and relative position of the robot with respect to the goal position. The sensor planning problem is to choose the type of sensing and the sensor patterns to be used in the feedback loop. It must ensure that the entire action, from initial state to final state can be carried out using some combination of sensing, inference, and servoing.

The connection between actions proposed by the task level planner and the operations planned at the sensor level takes two forms. For informatory actions, the action generated by the task planner corresponds directly to a task to be planned by the sensor planner. In some cases, the task level planner is allowed to specify accuracy requirements or other conditions that must be met. For example, a pose estimate may need to meet accuracy requirements in order to support a particular fine motion strategy. Performatory actions that require sensing support are "decomposed" into sensing tasks by the sensor planner. This decomposition is governed by the contextual constraints surrounding the action. For example, approaching an object requires recognition, pose estimation, and tracking to support control. The sensor planner also has information on the type of control that is taking place so that it can choose the features to be tracked based on control stability criteria. We discuss these issues in greater depth in the next section.

## 2.1 Experimental Planning Environment and Example

Our experimental environment consists of the ROTEX (RObot Technology EXperiment) robot, sensors and workcell. ROTEX is a German space telerobotic experiment that will fly with the next German space lab mission (D2) in 1993. It consists of a six-axis robot equipped with a sensorized gripper and an external fixed pair of video cameras providing a stereo image of the robot workspace. The gripper is provided with a number of sensors: two six axis force-torque wrist sensors, two tactile arrays, grasping force control, an array of 9 laser-range finders and a tiny pair of stereo cameras. ROTEX is designed to operate autonomously, to be teleoperated by astronauts, or to be teleoperated from a ground station. A more detailed description of the ROTEX environment can be found in [9] or [10]. Although the ROTEX environment was primarily designed for telemanipulation, there are additional interfaces in the laboratory version of ROTEX that support other modes of operation. We note that, for the moment, the configuration used for sensor planning experiments only uses a single hand-camera, the hand-mounted range sensors, and a single global-camera: no stereo information is available.

Our sensor planning framework will be illustrated using one of the D2 mission experiments: the task of finding, grasping and removing the *ORU* (Orbit Replaceable Unit), an

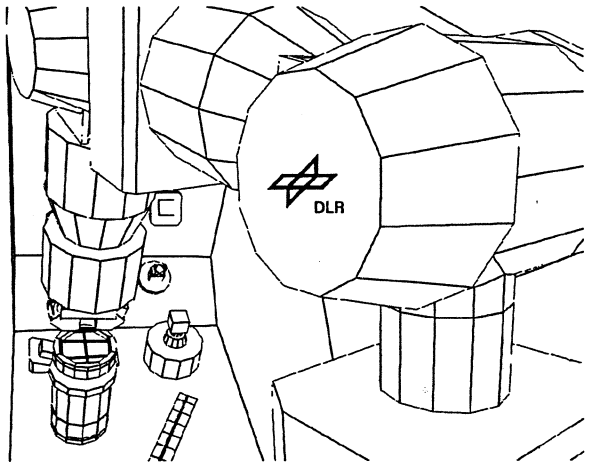


Figure 1: The ROTEX workcell.

electro mechanical plug (See figure 2). The *ORU* is removed by expanding a grip on the top, executing the proper twisting motion, and then pulling away from the attachment surface. We will consider planning for the recognition and sensor-guided approach to the ORU.

We note that experiments with the system in teleoperated mode have shown that this task is extremely difficult to carry out with no sensor feedback. The tolerances on the hand position and orientation are very small, and the sensor information available to the human operator is often difficult to interpret. The system permits the operator to use the four laser distance sensors pointing out of the fingertips to control gripper orientation near the *ORU*. This simplifies the task greatly. We have recently developed a vision-based controller that can control gripper position parallel to the surface of the *ORU*. Combining both range and vision leaves the operator the simple task of getting the robot near the *ORU*, turning on sensor-based control, and then controlling the $z$ axis motion until the gripping position is reached.

# 3 The Sensor Models and Planning Architecture

As noted in the previous section, requests to the sensor planner are either basic sensor requests or higher level actions that are decomposed by the sensor planner into a sequence of more basic sensing actions. For example, the high level action *approach(ORU,final_pos)* is decomposed into the following operations: *approach(ORU,final_pos)* :=[ *recognize(·); pose(·); tracking(·)*]. The first two operations are static, informatory actions, while the latter is used to support performance of a dynamic closed-loop motion. The sensor planner constructs a plan for these actions using information available on the *ORU*, the available sensors, and final position criteria supplied in the request.

Information on objects and sensors comes from object models, sensor models, and sensor simulations. The object models are from the CATIA CAD modeler which provides us with a boundary representation from which physical edges and planar surfaces can be

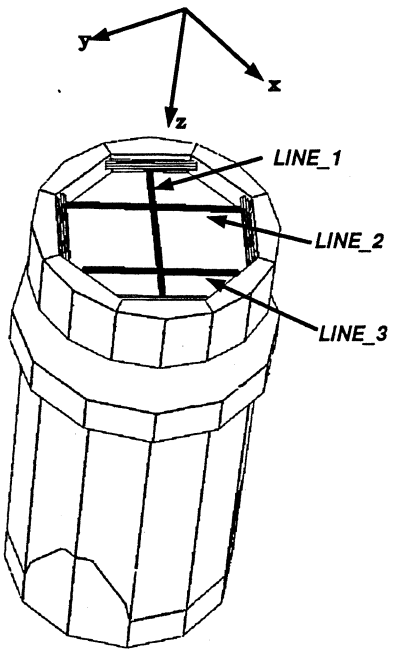


Figure 2: The Orbit Replaceable Unit.

easily extracted. In addition, we separately represent artificial markings such as the guide lines on the ORU. Using this information, we have implemented and tested simulations for the distance sensors, a laser range sensor, and a 6 dof force/torque sensor. Further we have also implemented a primitive camera simulation which is currently being extended to deal with more sophisticated features and imaging models.

## 3.1 Logical Sensors

The basic representation in the system is a structure similar to the notion of Logical Sensors [8]. The basic form of the structure is:

| | |
|---|---|
| **Logical Sensor** | <name> |
| **Input_List** | {<Logical Sensors>} |
| **Data_Proc** | {<return> = <data_proc(...)>} |
| **Type** | **static** \| **dynamic** |
| **Controller** | {<ctrl>} |
| **Valid** | {<cond>} |
| **Characteristic** | <pose>; <fixed\|movable>; <field of view>; <...> |

At planning time, these logical sensors are instantiated and organized into a network structure based on various choices of input sources from the *Input_List* and data processing procedures from the *Data_Proc* list. Each input process has a return value which is used as input to a data processing algorithm. The *Type* slot indicates whether this sensor provides information just once, or over an interval of time. The *Valid* slot represents constraints that must be satisfied by the input to the chosen data procedures in order for them to function correctly and reliably. The *Controller* must check the sensor data with respect to *Valid* and, in case of a dynamic operation, coordinate and synchronize the instantiated data processes. If the Logical Sensor represents a physical sensor it may also

control the device. Physical sensors use the *Characteristic* slot to represent information like the position, the field of view, work-criterions and very important if its a fixed sensor or its mounted on a movable device.

## 3.2 Sensor Planning Architecture

Planning is viewed as instantiating and refining the logical sensor associated with an action based on information about the sensing task. Each action of this type has three parameters: the information the task planner expects from the sensor system; the subject; and, the goal conditions for the entire request. Goal conditions on sequence requests like *approach* are extended to apply to each atomic sensor request and form a common condition the sequence must satisfy. For example, the *approach* action supplies a final position and positioning accuracy that must be attained during the course of executing the three basic actions. The first step in sensor planning is to perform a geometrical analysis of the *ORU* model to find task relevant geometric information that is observable by the available sensors from the set of initial sensor positions. For example, the *recognize()* request requires discriminating geometric features which uniquely identify the *ORU*. To this point, the features used in our environment are defined by hand and include lines, faces, special surface markings and combinations thereof. The geometrical analysis results in a set of geometric classes which are detectable by the sensors. In case of *ORU* the geometric classes are line, face and the relation of the three markings on the top.

The next step is to build a tree of logical sensor which can observe the object features, and to ensure that the execution of the associated data processing routines will be successful. The root of the tree is the logical sensor corresponding to the sensing task. Tree growth is constrained by knowledge of the feature classes computed in the previous geometrical analysis step. Starting from the root for each geometric entity all suitable pairs of Logical Sensors of the Input_List and data processes of Data_Proc define successor nodes. This process is continued recursively until the level of physical sensors is reached.

The class tree is then pruned and instantiated by using task information and output from the sensor simulation system based on the configuration expected when the plan will be executed. For each physical sensor a simulated measurement is performed and the observable geometries are evaluated with respect to the different data processes. At each node the validity conditions are tested, and if a test fails the node is pruned. So the result of the instantiation is a tree containing only reliable information. In our example, the camera simulation will find all lines and faces seen in the Figure 2. The filtered geometry is based on the sensor observation model and hence represents only the visible parts of it.

The instantiated tree is then analyzed with respect to the goal criteria for completeness and redundancy. Completeness addresses the problem if the expected sensor information is sufficient to attain required accuracy, stability, or uniqueness conditions. In case of *recognize* the simulated information is checked if it uniquely identifies the *ORU*. Redundancy is an important consideration as it increases the robustness of sensor-based action, but it also increases the complexity of data processing. In some case, the sensor planner may need to choose a compromise between safety and performance of the system. The failure of any of the above steps causes the sensor planner to reject the request as unsatisfiable.

## 3.3 Constructing Plans for Reactive Actions

Just as *recognize(·)* has a corresponding logical sensor, so do dynamic actions such as tracking. We view the problem of planning sensing to move between two points as the problem of constructing the observer for a series of *adaptive regulators* that attain the final desired position. These regulators are "constructed" by configuring pre-defined generic regulators using the appropriate choice of sensor inputs and goal patterns. More precisely, we are considering the following problem.

**Given:**

- a geometric object description;
- a model of sensor feature detection;
- a range of initial positions defined relative to the object; and
- a goal position defined relative to the object.

**Determine:**

- a collection of "regulators" and sensor inputs;
- the range of stable regulation for each; and
- a switching behavior.

**So That:**

- All plausible paths from the initial state to the goal position are covered by a stable regulator.
- The system deviates as little as possible from the straight line path from the initial state to the goal state.

We note that, although there is a great deal of discussion of the local and global stability of regulation systems [15], we do not know of any computational methods for synthesizing a regulator and analyzing its region of stability. Moreover, while this synthesis problem could, in principle, be formulated analytically, we postulate that any completely analytic solution would be computationally infeasible to carry out within the time constraints imposed by practical planning.

Consequently, we have begun to explore approximate solutions that involve decomposition of general motions into combinations of specific motions for which the regulation problem is well understood. Specifically, we note that if a goal pattern is in view of the camera, then servoing forward while holding a point on the pattern in the center of the image will lead the system to translate to the point regardless of orientation. Consequently, one way of positioning the gripper is to approach a pattern until a distance is reached where other visual patterns or visual and range information can be used to control orientation as well as position. Furthermore, the gripper can rotate about an object by "linking" together visual patterns that contain enough information to control distance (via scale) as well gripper/object relative orientation. Thus, an effective motion plan consists of: 1) finding a pattern on the same surface as the goal that is visible; 2) proceeding toward that pattern; 3) rotating or navigating about the surface until the goal is visible; 4) orienting and translating to the goal position.

For the gripper-mounted camera, the available information sources are various types of gradient-based pattern tracking as well as feature-based tracking. As the camera moves, these patterns distort and scale. It is relatively straightforward to determine the range of visual angles and scales over which particular features or patterns are detectable [16]. In general, stability degrades as the camera moves away from objects.[1] This leads us to consider a representation of the stable region of control in terms of sections of a sphere. Recalling that stability at a point can be determined by examining the Hessian of the objective function describing the controller [12, 15], we can find the boundaries of the region of stable control by testing points on the surfaces of a spheres.

This basic idea leads us to consider a sensor planning algorithm using methods similar to the backprojection technique employed by Donald's motion planner [3, 11]. Essentially, the gripper is simulated at the goal position and a stable set of sensor inputs is chosen. The stable point is expanded to a stable region. Regulators are chosen on the boundary of this region and extended further until the set of goal configurations is encompassed.

More formally, if $s$ is a logical sensor, let $D_s$ denote the spherical section (essentially two intervals of angles and one interval of distance) from the goal over which the sensor detects the chosen features. If a regulator $r$ relies on sensors $A_r$ let $\mathcal{D}_r = \cap_{s \in A} D_s$. Hence, $\mathcal{D}_r$ is the range over which $r$ will receive the measurements that it expects. Let $\mathcal{S}_r$ denote the region of stability of $r$. Then $\mathcal{O}_r := \mathcal{S}_r \cap \mathcal{D}_r$ represents the operating range of the regulator $r$. If $I$ is a set of initial positions expressed relative to the goal point $g$, then let $\mathcal{I}$ denote the minimal spherical section such that $\mathcal{I} \supseteq I$. Let $A(\cdot)$ denote the range of angles of a spherical section. Our proposed algorithm then proceeds as follows:

1. Given a goal position $g$, synthesize a regulator $r_g$ such that $r_g$ is stable at $g$. Compute $\mathcal{O}_{r_g}$. If $\mathcal{O}_{r_g} \supseteq \mathcal{I}$ then stop.

2. Let $L$ be the upper bound on diameter in $\mathcal{O}_{r_g}$. Synthesize regulators on orientation and distance $r_1, r_2, \ldots r_n$ such that $A(\cup_{i=1}^{n} \mathcal{O}_{r_i}) \supseteq A(\mathcal{I})$ and $L \in \cap_{i=1}^{n} \mathcal{O}_{r_i}$.

3. Create a queue $Q$ of these regulators ordered by increasing maximum stable distance.

4. Iteratively:

   - Remove the first regulator $r$ with operating range $\mathcal{O}_r$ from the queue.
   - Compute a set of regulators $r_1, r_2, \ldots r_n$ on horizontal and vertical translation such that $A(\cup_{i=1}^{n} \mathcal{O}_{r_i}) \supseteq A(\mathcal{O}_r)$. Add them to the queue and sort it. If no such set exists, exit with *failure.*
   - Exit when the first element in the queue has an operating range with a maximal distance component larger the largest distance in $\mathcal{I}$.

Once a collection of regulators is found and the algorithm exits, the corresponding logical sensing network of feature trackers is instantiated and information for their initialization is planned. The final result is a tracking "program" that provides information for stable regulation during the approach operation.

[1] We note that perspective projection causes scale ambiguity in camera images so that camera control must be implemented using adaptive methods.

# 4 Discussion

The basic elements of the system we have described is currently being implemented within the ROTEX environment of Section 2. As noted above, many of the sensor simulations exists, as well as an adaptive vision-based regulator and a range-based regulator. We have a primitive camera simulation which we are planning to augment with a detection model similar to that described in [16]. The point stability analysis has been implemented for the monocular camera and range sensor combination. We are currently extending that analysis to spherical regions as described in Section 3.

Our analysis has been based on a reasonably known, structured environment. In particular, we are able to use the available environment models to constrain the initial conditions for most sensor planning operations. This facilitates efficient offline sensor planning. It appears that working in an unstructured environment requires much more online evaluation of execution conditions as well as a more general framework for task planning. In future work, we hope to consider the issues involved in working in unstructured environments.

**Acknowledgements:** This research was supported by DARPA grant N00014-91-J-1577, by National Science Foundation grants IRI-9109116 and DDM-9112458, by NATO Collaborative Research Grant CRG-910994, and by funds provided by Yale University.

# References

[1] P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proc. of AAAI-87*, pages 268–272. Seattle, WA, 1987.

[2] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[3] B. R. Donald. A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence*, 31(3):295–353, 1987.

[4] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265, June 1987.

[5] J. Firby. An investigation into reactive planning in complex domains. In *Proc. of AAAI-87*, pages 202–206. Seattle, WA, 1987.

[6] E. Gat and D. Miller. Modular, low-computation robot control for object acquisition and retrieval. Jet Propulsion Lab Technical Report, 1990.

[7] C. Hansen and T. Henderson. CAGD-based computer vision. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11(11), 1989.

[8] T. Henderson and E. Shilcrat. Logical sensor systems. *Journal of Robotics Systems*, 1(2):169–193, 1984.

[9] G. Hirzinger. The telerobotic concepts of ROTEX—Germany's first step into space robotics. *39th I.A.F., Bangalore, India*, 1988.

[10] G. Hirzinger, G. Grunwald, B. Brunner, and H. Heindl. A sensor-based telerobotic system for the space robot experiment ROTEX. *2. International Symposium on Experimental Robotics*, 1991. Toulouse, France.

[11] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1):3–24, 1984.

[12] D. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, New York, N.Y., second edition, 1984.

[13] D. M. Lyons and M. A. Arbib. A formal model of computation for sensory-based robotics. *IEEE Journal on Robotics and Automation*, 5(3):280–293, June 1989.

[14] D. McDermott. Planning reactive behavior: A progress report. In K. Sycara, editor, *Innovative Approaches to Planning, Scheduling and Control*, pages 450–458. Kaufmann, San Mateo, CA, 1990.

[15] K. Narendra and A. M. Annaswamy. *Stable Adaptive Systems.* Prentice Hall, Englewood Cliffs, N.J., 1989.

[16] K. Tarabanis, R. Tsai, and P. Allen. Automated sensor planning for robotic vision tasks. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 76–82. 1991.