3

A Survey of Reactivity[1]

Srinivas S. Ravela

Laboratory for Perceptual Robotics
Department of Computer Science
University of Massachusetts

Technical Report #92-61
September, 1992

Abstract

This report is a survey of reactive systems. Horizontally decomposed architectures employing the classical planning paradigm confront problems associated with timeliness, flexibility, robustness, and are sensitive to non-stationary, uncertain, and incompletely modeled environments. In contrast, reactive architectures attempt to provide a timely response, do not rely on detailed or complete world models and generate actions triggered by the current perceptions of the agent. However, reactive architectures are often "short-sighted." This letter examines reactive architectures, as well as architectures that integrate reactivity and classical planning paradigms and presents the salient features from a subset of currently available literature. Papers are drawn from the domain of robotics, specifically robot navigation, path planning, and grasping in order to provide an analysis of AI methodology in real world situations. An attempt is also made to maintain a consistent definition of various terms, such as *reflexive, reactive, and deliberative behavior, behavior composition, and world models*, among others, used widely in the realm of planning and control in AI research.

[1] This study was conducted as part of an independent study under the supervision of Prof. Rod Grupen

Contents

List of Figures

1 Introduction

This letter is a report on a study in reactivity. The aim of this study is to understand the construction methodology, the advantages and disadvantages of reactive systems. The papers discussed here are chosen from the domain of robotics, because this domain provides an ideal platform to study the application of AI methodology in the real world. The real world is an interaction of multiple agents. For this reason it is dynamic and unpredictable. Also, the real world cannot be completely observed due to the uncertainty in the environment and the limitations of sensors. Agents operating in the real world should cope with these limitations. Classical planning approaches are known to be time consuming, require a relatively complete world model and assume the state of the world changes only due to the actions of the agent under control. This is not true as the study of various papers indicate.

In contrast reactive control offers the advantage of guaranteed response and hence the ability to quickly react to a changing environment. The control system is modeled as reacting directly into the current situation. The behavior of the agent is specified in such a system as situation action rules that are evaluated at frequent intervals. Thus, *Reactivity* may be defined as actions that are a function of current percepts. This definition is qualified by the following properties:

1. Reactive behavior is declarative.

2. Is dynamically composable.

3. Is context activated.

4. Filters bounded noise in the environment.

Response to temporal constraints imposed by the environment on the agent is one of the factors that necessitates design of reactive systems. This study strongly suggests that actions based on local knowledge alone is prone to failure in a complex world. Most of the architectures described in this letter employ more than local knowledge. Architectures that do not employ more than local knowledge have been observed in this study to be inadequate, i.e these architectures do not cater to issues pertaining to the real world.

In this study we will observe the above mentioned properties of reactive systems. The letter is organized in three sections. Section 2 is the most significant part of this study and contains a literature survey of papers that deal with reactivity. Section 2.1 gives us a brief overview of the environment, describes a characterization of the environment along three dimensions and the relation between these dimensions and the types of internal state that an agent may be endowed with. Section 2.1.1 is an overview of architectures for intelligent systems that employ a reactive component. Section 2.2 studies reactive architectures, models of interaction and behavior composition. Section 2.3 considers the issue of merging reactive and deliberative (planning) approaches to building intelligent systems. We sum up with our observations in section 3 and a conclusion.

2 Literature Survey

2.1 An Overview

The real world may be considered as a complex interaction of multiple of agents. For this reason the real world is considered dynamic and unpredictable. The world is never completely observable. These aspects are described in Chrisman's paper [Chrisman,1991], who then argues that internal state is essential to compensate for perceptual incompleteness. He also characterizes the environment along three dimensions, namely, complexity, predictability and density of choice points and ascribes a relation between these dimensions and the type of internal state that an agent might require. This paper is discussed below:

Chrisman et al. in their paper [Chrisman,1991], argue that perception is incomplete when at any instant relevant features of the world cannot be observed. Features are relevant when the appropriate choice of action for a situation depends upon those features. Therefore for perception to be complete, the world must be completely observable at every instant of time.

Perceptual incompleteness arises due to a number of reasons including sensor limitations, physical obstructions, monetary costs of sensing, computational costs of sensing, mutually exclusive sensing and uncertainty, [Chrisman,1991]. An example from their paper illustrates perceptual incompleteness at several places. There is a broken object in the middle of a room near a toolshed. An agent whose job is to repair broken objects enters the room and discovers the object. To repair the object the agent must go to the toolshed select an appropriate tool and return to the object. The object is invisible the moment the agent turns away from the object. If actions were based on current percepts only, then the agent would not be able to pick the right tool. To compensate for perceptual incompleteness the authors propose the following types of internal state:

1. *Model Reactive Agent:* A model reactive agent records only information that represents some aspect of the current world state. In the tool shed example, the agent memorizes the state description of the object and uses this description to fetch a tool.

2. *Expectation utilizing agent:* An expectation utilizing agent uses .the current percepts to select a plan to act through spans of incomplete perception. That is it selects a schedule of behaviors and reschedules when any future percept indicates a failure of the currently chosen schedule.

3. *Contingency Anticipating Agent* maintains a contingency state. This state changes as a result of the current percepts and has the effect of triggering particular decisions in the future. Such an agent does not make expectations about the future. In the tool shed example when the agent sees the broken object it records the fact that whenever it sees a phillips screwdriver it should pick it up.

Next, they characterize the environment under three mutually orthogonal dimensions. These dimensions are:- Complexity, Predictability , Density of critical choice points. Complexity of the world state relates to the length of enumeration required to describe all pertinent aspects of the world state. If enumerations are long complexity is high. The environment is predictable when the agent's expectations about the future are reliable. Density of choice points in the face of all possible future execution paths, are those points where an actual decision impacts the effectiveness of the agent. A model-reactive agent is sensitive to the world state complexity since it must maintain longer descriptions as the world gets complex. The contingency anticipating agent is sensitive to the density of choice points since it must make potential decisions at a time when relevant observations are available. In the tool shed example, the world is not complex because it would require very little memory to represent the fact that the object is broken . The world is predictable because it is assumed static within this example. Given the example, the density of choice points is low, because the problem does not define the presence of obstacles or other processes.

The salient features of this paper are:

- Perception is incomplete in the real world.

- Internal state is required to compensate for perceptual incompleteness.

- The three types of internal states are: Model reactive, Expectation utilizing and Contingency anticipating states. Note that such an explicit classification is not always possible. In the papers studied here, as in [Brooks, 1986], such classification is not possible. I believe it is the notion of state that is important. The classification presented here is a good explanation for the notion of state and the significance of 'state' in the context of this study.

- The environment can be characterized along the following dimensions. Complexity, Predictability and Density of Critical choice points.

- Model reactive agents are sensitive to the world state complexity, Expectation utilizing agents are sensitive to the predictability of the world and Contingency anticipating agents are sensitive to the density of critical choice points.

The observations from this paper are: model reactive agents primarily are reactive in nature. This goes by the definition since model reactive agents generate actions based on their current perceptions only. Expectations about the future are plans. Therefore, a reactive agent is sensitive to a complex world. This conjecture at this point is supported in all the papers studied. Expectation utilization or planning is sensitive to the predictability of the environment. The real world is unpredictable, therefore uncertain. The presence of uncertainty causes plans to fail. Therefore we can conclude that since the real world is unpredictable, planning alone may not generate solutions. Since the real world is complex a model reactive agent may not be sufficient. This letter addresses precisely these issues and we study the domains in which reactive approaches are suitable and if not, why not.

2.1.1 Intelligent Architectures and Reactivity

This section gives us an overview of intelligent systems that operate in the real world. All the papers described in this section pertain to architectures for mobile robot navigation which employ a reactive subsystem. Further, the study of reactivity in the context of planning gives us important information about the advantages of reactivity. The papers described in this section help us understand the scope and construction methodology of reactive systems. These papers also give us sufficient information to come up with a working definition of the term *behavior* widely used in AI research. The papers that are described in this section are [Brooks, 1986],[Payton,1986] and [Arkin,1988, Arkin,1987].

- *The subsumption architecture*

Brooks, in [Brooks, 1986] proposes an alternative to the classical horizontal decomposition of architectures for intelligent systems. Classical systems are horizontally decomposed into a series of functional units. They are: 1. Perception, 2. Modeling, 3. Planning, 4. Task Execution and 5. Motor Control. This architecture is fraught with several disadvantages, that are listed below:

1. *Classical architectures do not support timely action.* This is because information is assimilated from several sensors to construct a composite representation of the world, before any planning or execution takes place (Sensor fusion problem). This process is computationally expensive and time consuming.

2. *Classical architectures are not flexible.* This is because, any modifications to a functional unit, would cause significant changes in the interface. That is, this decomposition method does not support incremental development of a system.

3. *Horizontal decomposition is not robust.* This is because, the process of planning is based on the availability of a complete world model. Since the real world cannot be modeled completely(This is because the real world is never completely observable. See discussion on [Chrisman,1991]), planned-action may not result in changes in the world state as expressed in a plan. That is, plans do not account for the uncertainty in the real world, that results from the presence of unmodeled processes in the environment. For example consider the mobile robot navigation issue. It is possible that the robot, while following a supposedly safe path (a plan), may encounter obstacles (unmodeled processes) which are not accounted for in the world model. In this sense the plan fails and may require plan modification or replanning.

Brooks develops a robust, flexible, real-time software architecture for mobile robot navigation called the *Subsumption architecture*, that is constructed as a hierarchy of behaviors. This paper states criteria for evaluating an architecture, describes the control hierarchy, the construction procedure for individual control layers and an inter-layer communication protocol. All these aspects are studied and certain observations are made.

The requirements for an autonomous system in this paper are stated as follows:

1. *Multiple Goals:* The robot might have multiple (possibly conflicting) goals to satisfy. For example to stay on path and avoid an obstacle on path.

2. *Multiple Sensors:* A robot will have multiple sensors. Errors occur both due to lack of observability and noise. A robot must cope with these limitations.

3. *Robustness* A robot must be robust. When some sensors fail it should be able to adapt and cope by relying on those that are still effective.

4. *Extensibility:* As more capabilities are added to a robot, more processing power would be required; otherwise the original capabilities of the robot will be impaired with time.

The control hierarchy depicted below, corresponds to levels of competence. A level of competence is an informal description of a desired class of behaviors of a robot over all the environments it will encounter. These levels are:

- **LEVEL 0:** Avoid contact with other obstacles.

- **LEVEL 1:** Wander aimlessly without hitting things.

- **LEVEL 2:** See places at a distance and head for them.

- **LEVEL 3:** Build a map of the environment and plan routes from one place to the other.

- **LEVEL 4:** Notice changes in the static environment.

- **LEVEL 5:** Reason about the world in terms of identifiable objects and perform tasks related to certain environments.

- **LEVEL 6:** Formulate and execute plans that involve changing the state of the world in some desirable way.

- **LEVEL 7:** Reason about behavior and modify plans accordingly.

Each level of competence includes as a subset an earlier level of competence. The construction of the hierarchy proceeds in a bottom-up manner described below:

Layer zero is constructed to implement a level 0 competence. Layer one control can examine the data from layer zero and can suppress the normal data flow of layer zero. With the aid of layer zero, layer one achieves level 1 competence. By suppressing the data flow of layer zero, layer one can subsume the role of layer zero. Thus, higher layers can subsume the role of lower layers and the architecture is called the subsumption architecture. Figure 1 shows layer zero. Brooks argues that in the subsumption architecture:-

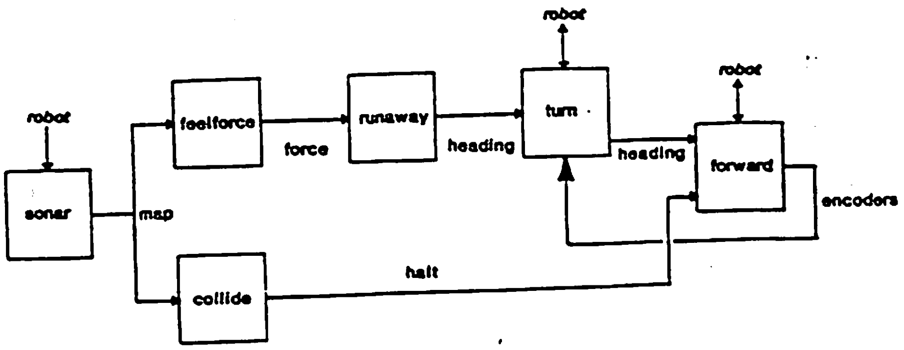


Figure 1: Layer zero control

- Multiple goals can be satisfied with different layers working simultaneously on different goals. Suppression mediates action and the ultimate decision may be made by using the results of pursuing all of them to some level of conclusion.

- Each layer uses only those sensors that determine the output of the layer. Perception is shared among behaviors. Only those sensors which are identified by the perception processing unit to be extremely reliable are used.

- The system is robust because lower levels which have been debugged thoroughly continue to run when higher levels fail to generate an output in time, but at a lower level of competence.

- Each layer can be made to run on an independent processor making the system extensible.

Brooks then describes the internal structure of a layer and a protocol for communication between different layers. Each module is a finite state machine. Input lines are single element buffers; one input line holds the perception datum, and the other functions as a reset line. The machine begins in the *nil* state and either of the following events causes a transition to another state.

1. Output: The result of computation of a function of the input, instance variables and buffers.

2. Side-effect: Instance variable being set to a new value.

3. Conditional dispatch: A predicate computation based on the module's instance variables.

4. Event dispatch: A monitor that monitors the input line and triggers a transition on a change in input.

For example consider the avoid module shown in Figure 2.

The force input line inputs a force with a magnitude and direction found by treating each point found by the sonar as a site of repulsive force decaying as the square of distance. Function select direction takes this and combines it with the input on the heading line considered as a motive force. It selects an instantaneous direction by summing up the forces acting on the robot. If this resulting force is above a threshold,(which is the force value that will result in a motion that takes more than a second) follow force is activated and converts the desired direction and force magnitude into velocity commands. If the force is less than the threshold the dispatch logic ignores it.

Communication between different layers is done asynchronously over a low bandwidth channel. A higher level control can either suppress the input line or inhibit the output. A lower level can also be reset by a higher control layer. The lines are held for a certain amount of time, typically is the time required for the higher layer of control to initiate motor control. Figure 3 shows layer zero augmented with layer one control.

- *Definition and types of behavior*

This paper demonstrates that an intelligent control system may be decomposed in terms of behaviors. In this paper behavior is observed as action generated by a functional unit called an agent in response to perception of the environment. In the subsumption architecture, each layer of control is a behavior because it generates action. For example layer zero is a behavior that determines output solely in response to the perception of the world, therefore it reacts. Layer one and layer two similarly react to their environments. Layer three and above are behaviors that generate action after deliberation. For example the function of layer three is to build maps and plan routes from one place to another. Therefore in part layer three plans i.e. deliberates. Therefore action, in this architecture, results both from reaction and deliberation. Thus, the subsumption architecture is decomposed into a hierarchy of deliberative and reactive behaviors. Note that in this sense the classical architecture is a purely deliberative behavior.

- *Reactive behavior can be used to attain real time performance*

But why is such a decomposition needed ? As discussed in the beginning of this section, classical architectures, may not generate timely action. This is because of the computational requirements associated with sensor fusion. Further the process of planning may not generate action (i.e. a complete executable plan) , within the temporal constraints imposed by the environment. Consider the following simple example. The mobile robot approaches an unexpected obstacle in the course of following a trajectory. Planning, with the associated sensor fusion problem may not generate action in time to avoid this obstacle. This demand for real time performance in general is not addressed by the classical decomposition method. In the subsumption architecture, real time performance is generated by the low level behaviors, or as we classify them, the reactive behaviors. This is because, of two reasons. First, the problem of sensor fusion is avoided by predetermining the sensory requirements associated with each behavior. Second, reactive behaviors are procedural units that generate action from the perception of the immediate surroundings. No planning is performed.

- *Hierarchical behavior decomposition is robust and flexible*

The decomposition also provides a mechanism to build robust and flexible systems that facilitate incremental development of a system. This decomposition technique is robust because in the event of failure of a module action is still generated by the other modules, possibly at a lower level of competence or at a slower speed. Since each layer exhibits behavior independently, addition of new behaviors does not alter the existing behaviors in any significant way.

- *Behavior composition is achieved via subsumption*

This paper also suggests a mechanism for composing behaviors. This composition is necessary in order to generate unique actions. A control layer with a higher degree of competence can subsume a behavior of lower competence. Thus, behavior composition is achieved through a priority based arbitration, in which a behavior with a higher degree of competence has greater priority. This paper discusses issues pertaining to levels 0,1 and 2 only. This paper does not show how deliberative behavior may be achieved.

- *Vertical Decomposition: Payton*

Payton in [Payton,1986] proposes an architecture
based on vertical decomposition of control, different from [Brooks, 1986]. This paper is different from [Brooks, 1986] because, instead of considering levels of competence, this architecture is based on a hierarchy of levels of abstraction. This architecture uses plans at a symbolic level to constrain the choice of reflexive behaviors that implement the plan. These behaviors guarantee real time performance. However we will observe a coarse correspondence between both these architectures.

Payton in this paper develops an architecture for an Autonomous Land Vehicle(ALV).The author specifies the requirements of the system, the various design tradeoffs and presents an architecture with an emphasis on local and reflexive planning. Together these two modules constitute a reactive subsystem.

This architecture is designed to handle diverse terrain requirements through a library of reflexive strategies. Payton argues that since, the vehicle is in continuous motion, therefore sensor data that were to result in an action based on a detailed assimilation and planning process, would be obsolete by the time action is produced. Hence a short reaction time is essential. Secondly, unlike simple road following, the vehicle needs to tackle diverse terrain and adversary threats. Thus, a wide variety of planning strategies may be required. This means a variety of sensing capabilities are required. The problem of assimilation of sensor data or sensor fusion to produce plans, for a real world would lead to an overwhelming computational load. Therefore, the process of assimilation may have to be sacrificed in order to achieve immediacy required for real time response.

Assimilation of data is advantageous because it gives a relatively complete view of the world. This is of value to planning because, features that are critical to plan execution may not be otherwise discovered. However, this may not be timely. For example as a sample of data is going through analysis, the robot might have already run into an obstacle. Immediacy on the other hand as Payton argues is of value to control. The perception action cycle may be viewed as a feedback control unit, with the environment providing the feedback. Greater stability results if there are fewer delays between sensing and action. Greater immediacy should therefore result in a more stable system. However, data that may not have gone through sufficient assimilation may not provide the system with critical information.

Payton also discusses the advantages of a vertical decomposition based on levels of abstraction over horizontal decomposition. In this paper, Payton addresses the issues of flexibility and sensor fusion. In a horizontally decomposed system fusion of data is performed through the generation of a composite representation of the local environment. This would cause considerable delay between sensing and action. This delay can be reduced by increasing the throughput of the system or by bypassing certain modules, in which case the generality of the approach is lost. As discussed in [Brooks, 1986] this is inflexible because it would require changes at the interfaces. The vertical decomposition scheme proposed by Payton is shown in Figure 4. Sets of sensors generate specialized representations for use by the behaviors. The task then is to arbitrate among these behaviors or compose them. This is termed by Payton as command fusion. He argues that although command fusion needs to be resolved, this scheme is much better since specialized pathways for action have already been generated, to ensure a quick vehicle response. The organization of the system in terms of behaviors makes the system flexible because addition of modules does not significantly alter the other interfaces.

In Payton's architecture the planning hierarchy consists of mission planning , map-based planning, local planning and reflexive planning. The mission module translates abstract goals into a set of geographic goals and has a response time of a few minutes. The map based planner translates geographic goals into route plans, also has a response time of a few minutes. The local planning module ensures that the route plan is executed properly. The reflexive planning module has the task of maintaining real time vehicle control. This requires a response of less than a second. Note that assimilation is provided by higher levels and immediacy by the lower two. While commands are passed downwards failure and other status reports are passed upwards. Each individual module is organized as follows:

A module consists of expert sub-modules or agents. New agents can be added without affecting the performance of the system significantly. However, simultaneous activation of a number of agents may not be meaningful , due to redundant or conflicting actions that can be generated. Therefore, the agents are partitioned into activation sets, with each activation resulting in a meaningful action. Payton's architecture is different from [Brooks, 1986] because it is not organized as a hierarchy of behaviors. However, in the subsumption architecture higher levels of competence require an increasing degree of assimilation. Hence there is a correspondence between both the architectures. Payton also organizes each module differently. In the subsumption architecture each module is organized horizontally, i.e. as a perception action cycle, where action is a function of the perception. Payton organizes each module laterally.

Each module is a collection of expert agents that communicate through a common blackboard. Each expert agent receives some combination of processed sensor data. An agent constitutes a sequence of ; 1. A perceptual element (called virtual sensor) and an action component called the behavior. Behaviors are procedural specifications and an activation set is a group of behaviors. An activation set is selected by the local planning module during task execution from a pool of reflexive behaviors. When an activity is initiated each of the individual behaviors are parameterized and initiated as independent processes. These processes as stated above communicate through a black board. Now, each behavior places a mötor command with an integer priority. Command fusion is done via arbitration based on the priority values. If a particular behavior fails, control is transferred to another behavior which acts as a failure handler. Figure 5 shows an activation set, Figure 6 shows a behavior.

The local planning module based on the map based planning module's output and local perception generates a sequence of activities to be performed. It is responsible for selecting and instantiating activities, instantiate behaviors, detect and handle failures. Figure 7 shows the functional elements of the local planning module.

- *Observations*

In Payton's paper a noticeable feature is that the definition of behavior (expert agents producing action) is consistent with the definition of the term introduced in our discussion on [Brooks, 1986]. In Payton's paper action is produced by expert agents. These behaviors are reflexive, for the following reason. Expert agents generate actions in response to the immediate perception. These are based on simple computations on the inputs in the nature of the slow-for-obstacle shown in figure 6. Alternately, they are highly procedural units. Actions generated by these behaviors independently cannot execute a task In a broad sense, these behaviors have a *metabolic commitment* in preserving the constitution of the agent. This means that activity is generated only to avoid undesirable states that may arise as a result of the current interaction of the agent with the environment. For example slow-for-obstacle is a reflex essential to prevent the agent from running into the obstacle. Reactive behavior however, employs the current local perception as the basis for a choice of action to pursue a goal. The line of difference is very thin and frequently this difference arises due to the difference in the time required for local assimilation. Reflexes may not assimilate sensory data at all. They may be thought of as triggers which are activated when a sensor values cross certain predetermined bounds . See [Grupen,1991d].

- *Reactive behavior makes plans robust*

In this architecture, action is generated by these behaviors. The choice of behaviors is made by the local planning module, in response to the local environment. Further, this choice is constrained by the plan specified by the map-based planner. But why should a map plan not be implemented directly. The reason is because, the success of plans depends on the completeness of the world model. Since a complete world model cannot be maintained for the real world, plans may fail in the presence of unmodeled objects. These behaviors therefore, make the plans robust by accounting for uncertainties in the environment. Thus reactive behavior filters out *bounded noise*, where noise arises due to the presence of unmodeled processes , and in this sense make plans robust.

- *redundant behaviors makes the system robust*

Payton also argues that since the system is required to handle diverse terrains, redundant behaviors in the repertoire would make the system robust and generate actions that can handle a wide variety of situations.

- *composition via arbitration*

Payton also suggests a mechanism for composing behaviors, that is by arbitration. Each agent outputs an integer priority and the behavior with the highest priority is chosen as the successor. One problem with this scheme is the resolution would be required in case two behaviors, that may possibly result in conflicting behaviors put out the same priority values. Such problems have been resolved using random selection methods for example. However Payton does not discuss this issue.

- *Command fusion is better than sensor fusion*

Payton also suggests that the command fusion above is better than sensor fusion. Sensor fusion is a time consuming process. Instead each behavior has an associated set of sensors, that are determined from the action requirements of the agents. The issue then is to arbitrate among the behaviors, and this is better because pathways for action have already been generated.

- *Reactive navigation*

The next set of papers that are discussed in this section is work done by Arkin in [Arkin,1988], [Arkin,1987]. The architecture is very similar to Payton's architecture. We know that a behavior is an interaction of the world based on local perception. This requires a transformation of the world model into a domain that facilititates actuator commands. This transformation defines a model of interaction with the world. Both Brooks and Payton did not deal with this issue in detail. Arkin employs artificial potential fields as a mechanism for interacting with the environment. The transformation as we call it, is done by representing the objects in the world by their ascribed attributes. For example an obstacle is represented by a repulsive field. In this sense this model represents both the local environment and plans (stay on path for example) in a domain that is conducive to generate actuator commands.

Arkin [Arkin,1988, Arkin,1987] describes an architecture for reactive navigation for an autonomous vehicle. The system is decomposed vertically into five major components: the planning, cartographic, perception, motor and homeostatic subsystems. The planner consists of a mission planner, navigator, pilot and motor-schema mangers. The cartographer, whose task is to maintain the information stored in long term and short-term memory, provides this information on demand to the planning and sensory modules. Long-term memory (LTM) contains *a priori* knowledge about the world, while Short-term memory contains an acquired perceptual model of the world overlaid in the LTM context. The perception subsystem structures information in a coherent and consistent manner from sensory inputs available to the cartographer and motor schema manager. The motor subsystem is the means by which the vehicle interacts with it's environment. This interaction is in response to sensory stimuli and the high level plans. The homeostatic control is concerned with the maintenance of a safe internal environment for the robot. The concentration in this paper is on a behavior based planning subsystem for robot navigation and is the issue of interest to this study.

The planning subsystem has two distinct levels of information for path planning; the map based in the long term memory and sensor data. Planning is decomposed vertically as a hierarchical control system. The Mission planner takes mission commands in conjunction with the cartographer and the homeostatic subsystems and directs an output to the navigator via a blackboard. The output contains particular mission goals that must be satisfied. The navigator, uses this information in conjunction with the LTM and constructs vertex graphs, that represent the free space. It also issues status reports to the mission planner. The world at this point is modeled geometrically and objects are represented by polygonal approximations. The navigator moves the appropriate local *a priori* information available into the STM, based on the robot's current world position. The output of the navigator is a point to point path and is directed to the pilot. The pilot, uses this path and instantiates suitable behaviors from a repertoire of behaviors and passes them to the motor schema manager. It is important to note that the pilot operates based on the current perception of the robot. It is guided by the point to point path and monitors the execution by determining the success or failure of the motor schemas instantiated. The pilot is responsible for producing timely action. If the motor schemas to perform the desired activity, the pilot is invoked to compose (or instantiate) alternate behaviors. The motor schema manager maintains a model for interaction with the environment. Control for the execution path actually occurs within the confines of the motor schema manager. Figure 8 shows the decomposition of the planning hierarchy.

The local world is represented as a potential field to provide the appropriate velocity commands to the robot. This field is constructed from the schemas that are instantiated by the pilot, in the context of the Short-term memory. This potential field is constructed by composing the individual fields that are ascribed to individual behaviors. This is possible because of the property of superposition holds in potential fields. The potential field concept is dealt in detail in [Khatib,1985]

Figure 9 shows the various schemas and their definitions. Figure 10 gives a snapshot of multiple instantiations.

We begin our observations on these papers by identifying the reactive subsystem in this architecture. The reactive subsystem, comprises of the pilot and motor-schema manager. The pilot is effectively a monitor of plans and a composer of behavior. Behavior in this paper is a motor schema.

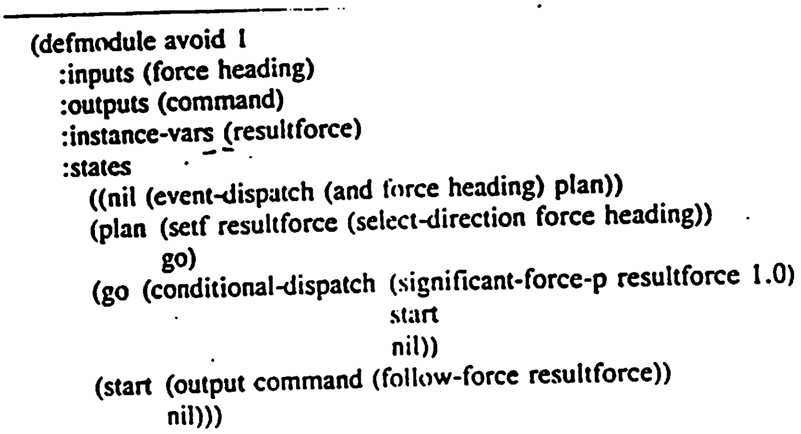
The salient features in Arkin's work are:

- The use of potential fields to model the interaction of an agent with the world.

- Composition of behavior as a superposition of primitive behaviors.

- The potential field representation may fail to generate the desired trajectory. This is because in the presence of unmodeled obstacles that form a cluttered environment, local minima may arise in the composite field [Khatib,1985]. This would require recomputation of the vertex graph to include these new set of obstacles and replan, i.e specify a path based on global observation. In Khatib's paper we note that this method of potential fields to generate reactive behavior therefore fails in the presence of local minima, in the surface that represents the interaction of the agent with the world. Also, I refer the reader to our discussion on work done by [Connolly,1992] who uses harmonic functions to resolve this problem.

With this broad overview of reactive systems, that have been employed in architectures for intelligent behavior, we now proceed to study the issues in reactivity. Section 2.2 describes reactive architectures, described in [Grupen,1991a, Yamauchi,1991], and models of interaction to achieve reactive control, described in [Khatib,1985], [Connolly,1992] . One of the conclusions that result from this study is that reactive behaviors alone may not find solutions. They require a certain amount of global information to be effective.



```
(defmodule avoid 1
   :inputs (force heading)
   :outputs (command)
   :instance-vars (resultforce)
   :states
     ((nil (event-dispatch (and force heading) plan))
     (plan (setf resultforce (select-direction force heading))
           go)
     (go (conditional-dispatch (significant-force-p resultforce 1.0)
                               start
                               nil))
     (start (output command (follow-force resultforce))
           nil)))
```

Figure 2: Avoid obstacle module

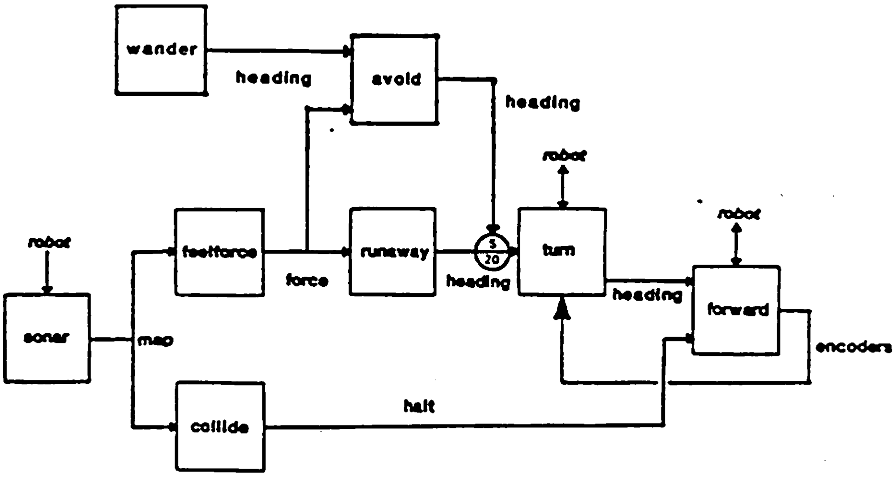


Figure 3: Layer zero augmented with Layer one

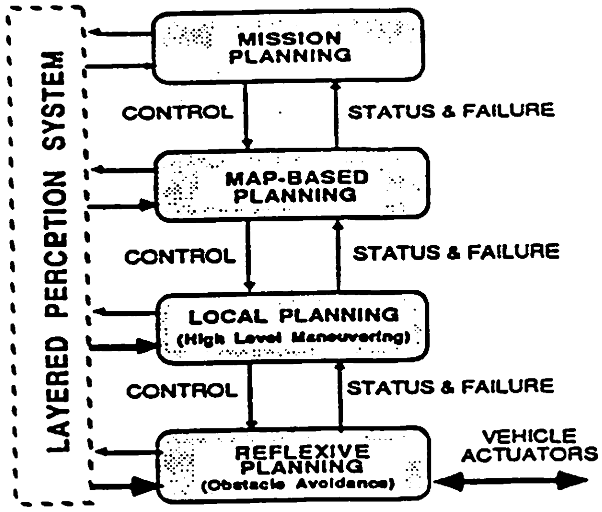


Figure 4: Payton's vertical architecture

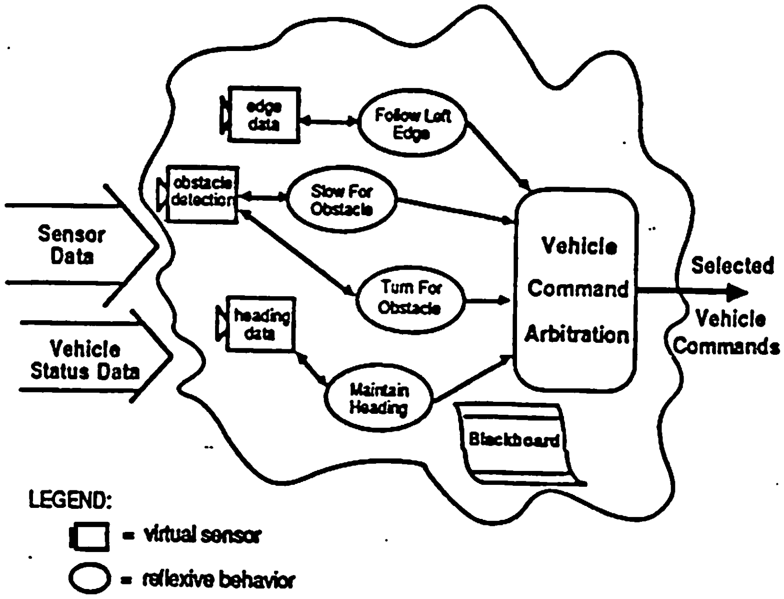
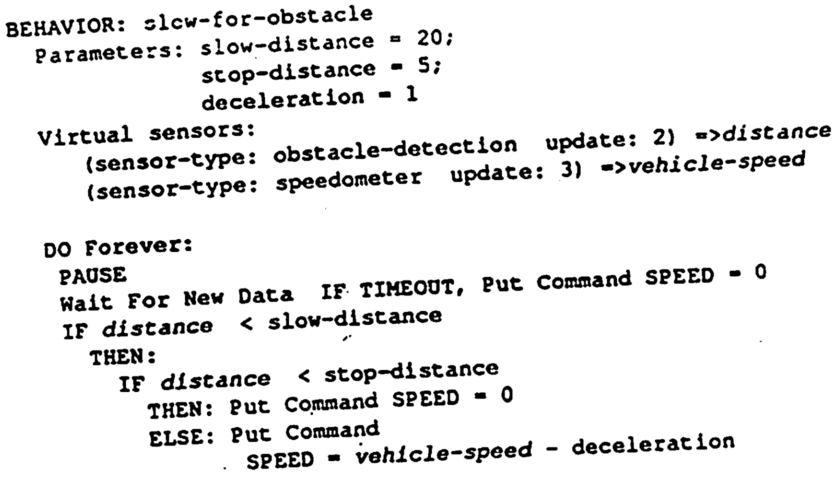


Figure 5: An activation set behavior



```
BEHAVIOR: slcw-for-obstacle
  Parameters: slow-distance = 20;
              stop-distance = 5;
              deceleration = 1
  Virtual sensors:
     (sensor-type: obstacle-detection  update: 2) =>distance
     (sensor-type: speedometer  update: 3) =>vehicle-speed

  DO Forever:
   PAUSE
   Wait For New Data  IF TIMEOUT, Put Command SPEED = 0
   IF distance  < slow-distance
     THEN:
       IF distance  < stop-distance
         THEN: Put Command SPEED = 0
         ELSE: Put Command
                 SPEED = vehicle-speed - deceleration
```

Figure 6: A behavior

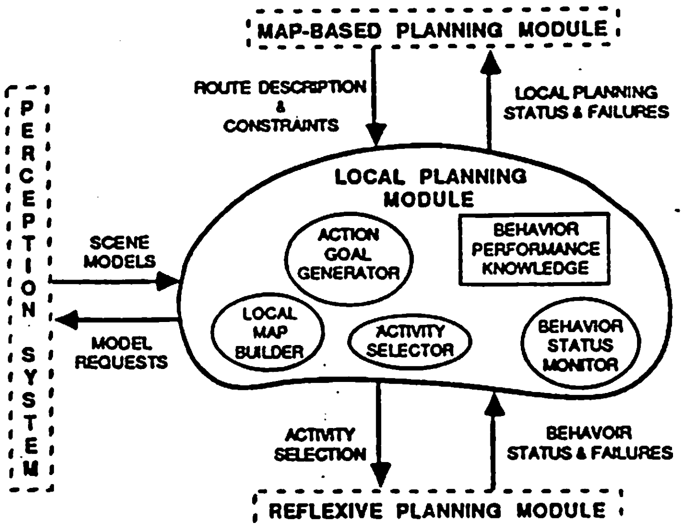

MAP-BASED PLANNING MODULE
ROUTE DESCRIPTION & CONSTRAINTS
LOCAL PLANNING STATUS & FAILURES
PERCEPTION SYSTEM
SCENE MODELS
MODEL REQUESTS
LOCAL PLANNING MODULE
ACTION GOAL GENERATOR
BEHAVIOR PERFORMANCE KNOWLEDGE
LOCAL MAP BUILDER
ACTIVITY SELECTOR
BEHAVIOR STATUS MONITOR
ACTIVITY SELECTION
BEHAVOIR STATUS & FAILURES
REFLEXIVE PLANNING MODULE

Figure 7: The local planning module

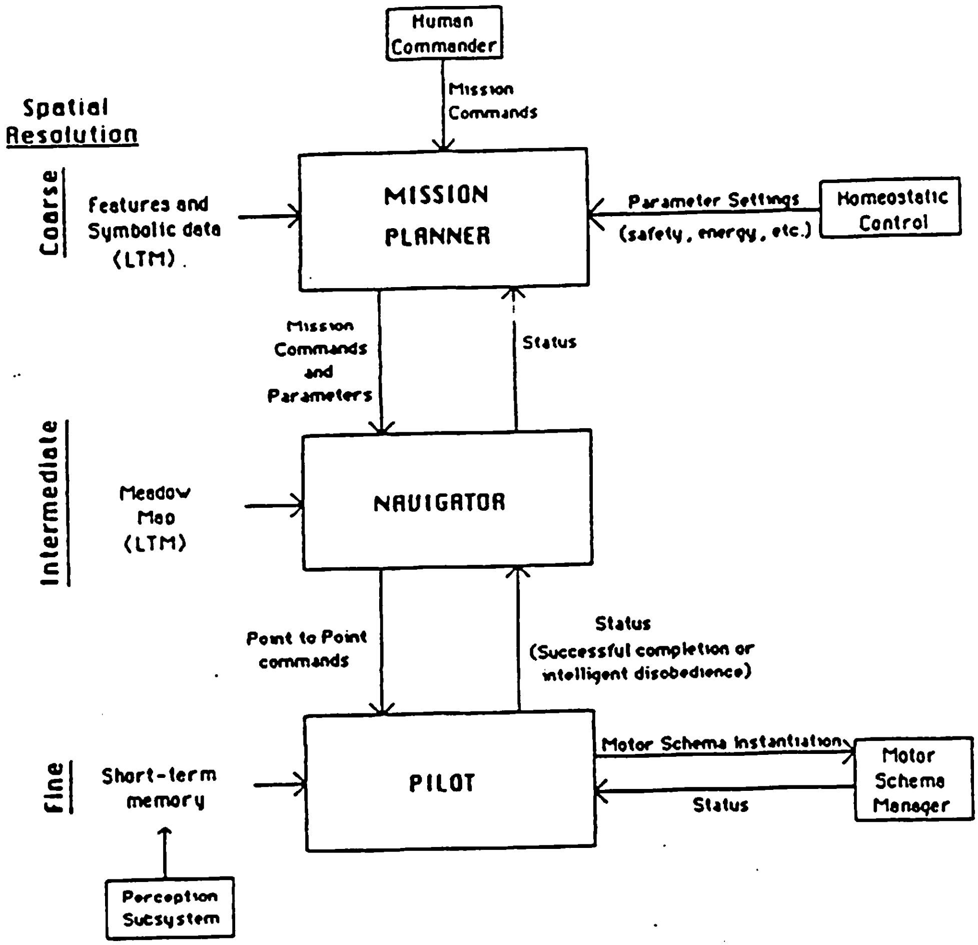


Figure 8: Planning Hierarchy-Arkin

- Avoid-obstacle

$V_{magnitude} =$

$$\begin{array}{l} 0 \;\; for \;\; d > S \\ \frac{S-d}{S-R} * G \;\; for \;\; R < d \leq S \\ \infty \;\; for \;\; d \leq R \end{array}$$

where:

S = Sphere of Influence (radial extent of force from the center of the obstacle)
R = Radius of obstacle
G = Gain
d = Distance of robot to center of obstacle

$V_{direction}$ = along a line from robot to center of obstacle

- Stay-on-path

$V_{magnitude} =$

$$\begin{array}{l} \text{P for } d > (W/2) \\ \frac{d}{(W/2)} * G \text{ for } \;\; d \leq \frac{W}{2} \end{array}$$

where:

W = Width of path
P = Off path gain
G = On path gain
d = Distance of robot to center of path

$V_{direction}$ = along a line from robot to center of path heading toward centerline

- Move-ahead

$V_{magnitude}$ = Fixed gain value
$V_{direction}$ = A specified compass direction

- Move-to-goal

$V_{magnitude}$ = Fixed gain value
$V_{direction}$ = In the direction towards perceived goal

- Noise

$V_{magnitude}$ = Fixed gain value
$V_{direction}$ = In a random direction for a specified persistence (time interval)

Figure 9: Motor-schema definitions

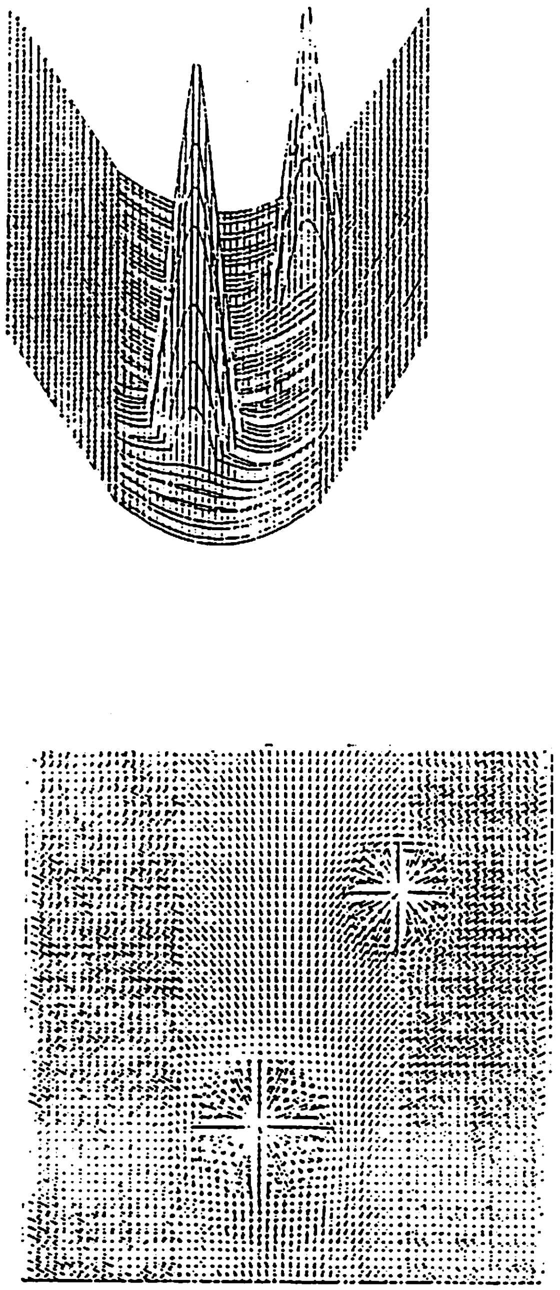


Figure 10: Potential field from multiple instantiations

2.2 Reactive architectures

In this section we study the construction methodology of reactive systems. Reactive behavior as stated earlier uses it's current perceptions to generate action. In [Yamauchi,1991] we observe that the dynamics of the real world are not modeled to determine behavior. In [Grupen,1991a], [Grupen,1991b] we observe that an object model is refined to employ reactive behavior. [Khatib,1985] and [Connolly,1992] describe methods of modeling interaction with the world to generate reactive behavior. Each method has been possible due to some observable feature of the environment. In [Yamauchi,1991] for example the dynamics of the environment are slow enough to employ qualitative models, in [Grupen,1991a] the object is stable. We will also observe that reactive behavior alone may not be sufficient to find solutions to complex problems.

- *The juggler: a reactive system*

Yamauchi [Yamauchi,1991], describes a reactive architecture,
called the Independent agents architecture. The behavior based architecture is shown if figure 11.

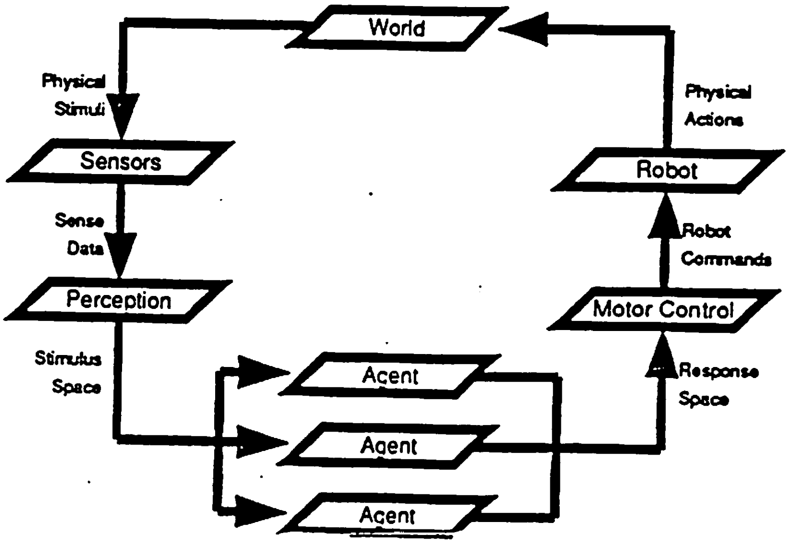


Figure 11: An independent agents architecture

The task is to implement a juggler that juggles a balloon off a paddle fixed to a Puma 761 robot arm. Sensing is done through a pair of cameras.

Perception uses sensor data to generate a point in stimulus space. The behaviors use this point in stimulus space to compute a point in a response space. The balloon is observed through two cameras and their horizontal centroids are computed. This is given by:

$$x_l = \frac{\sum_{x,y \in T_l} X}{|T_l|}$$

$$x_r = \frac{\sum_{x,y \in T_r} X}{|T_r|}$$

where, T_l and T_r are pixels above the intensity threshold of the lefty and right cameras respectively. The range is given by

$$y = \frac{k_{range}}{x_l - x_r}$$

where k_{range} is a constant relating image disparity to balloon range. The stimulus space is a 3-tuple (x, y, ρ_{old}), where

- x: is the centroid from the left camera $[x_{min}, x_{max}]$
- y: is the range of the balloon from the robot $[y_{min}, y_{max}]$
- ρ_{old} : is the old robot arm extension. $[\rho_{min}, \rho_{max}]$.

The behaviors are :

1. Rotational Tracker: Rotates the robot so it faces the balloon.
2. Extensional Tracker: Extends arm so that the paddle is located directly under the balloon.
3. Hitter: Swings the arm when the balloon is directly over the paddle.

They independently determine a response in the response space. The response space is a 3-tuple $(\Delta\theta, \rho_{new}, h)$, where

- $\Delta\theta$: is the rotation about the base $[\Delta\theta_{min}, \Delta\theta_{max}]$
- ρ_{new} : is the new robot arm extension. $[\rho_{min}, \rho_{max}]$.
- h: is the hit command $[0, 1]$

Each behavior results in a change in one of these three dimensions and the behaviors are mutually independent. Secondly, all three behaviors are required to determine the response at any given point in time. Action then is stated as follows. The rotational tracker tracks the balloon and causes the robot to face the balloon. Simultaneously, the extensional tracker, extends the arm so that the paddle is under the balloon. The hitter hits when the balloon is close to the paddle.

These behaviors are highly procedural and the current perception determines the output. The architecture is reactive and does not incorporate a planning component. Yamauchi argues that he uses a vertical decomposition strategy. I think this is incorrect. The decomposition is horizontal. Since perception uses sensor data to generate a point in stimulus space and the behaviors (independently) use this point to generate a point in response space. the decomposition is effectively horizontal. In terms of the architectures in [Brooks, 1986],[Payton,1986] , Yamauchi has presented a low level of control.

- *Pure reactivity is sufficient if the environment is completely observable*

In this paper Yamauchi argues that he does not model the dynamics of the world (balloon) to generate effective action. This is right but has been possible only because the balloon is filled with helium. Therefore, the juggler operates in a domain where the dynamics of the environment can be observed at all times. I presume the cameras continuously track the balloon. In this paper Yamauchi has primarily described a reactive architecture, that uses current perception to guide actions.

- *parallel composition of behaviors*

Behavior composition is achieved through parallel execution, because the behavior is decomposed into three mutually exclusive and collectively inclusive behaviors. The model of the world used (using the notation in this paper) is qualitative, and this is possible only because the dynamics of the world have been slowed down to make the world observable at all times. I believe in case the balloon were to drop faster, a more rigorous mathematical model would have to be maintained. An argument however against this could be that faster rate of drops can be accounted for, if greater computational speed is available. I think this argument would place us at the same disadvantage that classical planning does by assuming a static world. The aim in part, in building an intelligent system is to achieve the best possible results, given a real world and limitations in the present computational and sensory powers. Clearly, Yamauchi in this paper develops an architecture that reacts to the environment. However, the functions of the perception inputs which he terms the stimulus space, to determine the reactive response of the juggler have been possible because the environment has been slow enough to observe the motion of the balloon.

- *Grasping*

In [Grupen,1991a],[Grupen,1991b] , however, the environment is static. The uncertainty arises due to the lack of a predefined model for the object to be grasped. At no stage is the object completely observable. This paper is discussed below:

Grupen in [Grupen,1991a] , [Grupen,1991b] , [Grupen,1991c] , describes a reactive grasp formation strategy for an unknown object. He describes an incremental technique for grasp formation from an incomplete world model. He describes a multiple resolution model of the environment, a model for the grasping task and a model for an agent that represents the manipulator, which complies with the changing contact geometries that occur during task execution.

The Jacobian is a linear transform that describes the cartesian tip velocities resulting from joint velocities. If a unit sphere from R^4 (describing the joint space) into R^3, the result is a manipulability ellipsoid. The manipulability index is a scalar metric describing the conditioning of the manipulator. This is proportional to the volume of the manipulability ellipsoid. Since the volume of the ellipsoid increases as it becomes spherical, there is correlation between high manipulability and isotropic conditioning. Grupen then presents a weighting coefficient which penalizes extreme joint configurations. The product of this coefficient over all joint configurations in the manipulator's work space is a scalar field. The product of this field with the manipulability index is a weighted manipulability isogram. Hence given a contact geometry we know the configuration for which the manipulability metric is maximized. The Manipulability index for the hand is empirically given by the following formula.

$$M_{hand} = \frac{\Pi_i M_i}{\sum_i M_i}$$

This is analogous to the parallel resistance equation. Thus the manipulability of the hand tends to associate itself with the smallest finger index M_j .

Given contact geometries and the present hand index candidate hand frame movements can be enumerated and the grasp which improves the hand index the most is as the successor state. Let us call this the conditioning behavior, that models the manipulator's response to a task.

Now, the task is specified relative to the object coordinate frame as a set of wrenches

$$\tau = \left\{ \left(\alpha_i^+, \alpha_i^-\right) \cdot \hat{t}_i \right\}$$

where, $\hat{t}_i \in R^m$ (m is the dimensionality of the wrench space) is the task basis and α is the associated magnitude in the positive and negative directions. Grupen initially constructs a geometric model of the object represented by a set of planar facets. The geometric model is transformed in to a force domain model. This model is the model for interaction that determines the effector commands. The geometric object is maintained to ensure consistent interpretation of the sensor data. Given a local planar surface and an associated contact model, the forces transmitted by the surface are enumerated at the contact locations, using unit normal and frictional forces. i.e.

$$F = \left\{ \vec{f} | \vec{f} \in R^n \right\}$$

These forces map onto a set of object frame wrenches

$$W = \{\vec{w} | \vec{w} \subset R^m\}$$

$W \subset R^m$ through a linear mapping $\mathcal{G}$. Assuming no contact torques are applied to the object's surface we have

$$\mathcal{G} : R^n \rightarrow R^m$$

defined as ,

$$\mathcal{G}\left(\vec{f}_i\right) = \left(\vec{f}_i \left(\vec{r}_i \times \vec{f}_i\right)\right)$$

i.e.

$$\mathcal{G}(F) = W$$

$\vec{r}$ is the position vector relative to the task frame. An error is then computed with respect to the task specification τ . This requires that W be projected along the task basis $\hat{t}$, i.e.

$$\mathcal{W} = \left\{ \left(\beta_i^+, \beta_i^-\right) \cdot \hat{t}_i \right\}$$

The error then is a difference in magnitude along each task basis vector i.e.

$$\mathcal{E} = \left\{\left(e_i^+, e_i^-\right) \cdot \hat{t}_i\right\}$$

The grasp solution consists of simultaneously reducing the components of E by modifying the set of contact sites. The trajectory selected is one that minimizes the quadratic metric defined for each contact as

$$\Delta u_{opt} = \frac{\sum_i \left\langle \frac{d\vec{w}}{du}, \vec{e_i} \right\rangle}{\sum_i \left\langle \frac{d\vec{w}}{du}, \frac{d\vec{w}}{du} \right\rangle}$$

where $\frac{d\vec{w}}{d\vec{u}}$ expresses how the contact wrench changes in the neighborhood of the current contact position as a function of the surface coordinate $\vec{u}$. The local force domain surface elements are smoothed and interpolated to obtain a global force domain hypothesis, by projecting the wrench surface fragments onto a finite series of fourier basis functions. The series approximation is given by:

$$g(u) = \frac{A(0)}{2.0} + \sum_{n=1}^{m} \left[A(n) \times cos(nu) + B(n) \times sin(nu)\right]$$

Initially a global model that behaves like a low pass filter is considered by using the first harmonic in the fourier series model. A gradient descent is performed to minimize the above mentioned error term. The first harmonic model is convex and there are no local minima. For objects that are ellipsoidal this is sufficient. This is shown in Figure 12 . However this may not be sufficient. See Figure 13. Therefore, local information is employed to perform reflexive adjustment to optimize the grasp locally. In Figure 13 for instance the contact location of the right location is held fixed and the left contact location is migrated to construct a null grasp. The conditioning behavior is responsible to reposture the hand to suit this contact migration. Grupen's technique can be summed up in the following manner:

1. Describe the task
2. Construct a force domain model.
3. Use a low pass model of the object in the force domain. Compute error w.r.t. task and minimize this error
4. The hand should comply with the finger migrations.

5. If the goal has not been reached, increase the resolution of the force model. Repeat.

In this paper Grupen has implemented a reactive control layer. The control layer is specified by a finite state automaton shown in Figure 14 and has been described above. There are several observations to be made from these papers:

- *The reactive process*

Grupen's architecture demonstrates an important technique to achieve reactive control. This process is described below:

1. Specify a goal. In Grupen's approach, it is the state which minimizes the error with respect to the task. In Arkin's paper this is represented by specifying the stay on path behavior.

2. Present the apriori information as the world model. This is global information. In grasping for example, the first harmonic of the force model serves this purpose. In Arkin's paper this is achieved by instantiating behaviors from the STM.

3. Employ local perception to account for unmodeled processes. In Grupen's approach, this is observed as the local optimization of the contact. In Arkin's paper this is used to instantiate behaviors based on the pilots local perception. Note that the information contained in the STM is the world in so far as the reactive module in Arkin's paper is concerned. Also note that the model of interaction uses some attribute of processes in the perceived or represented local model. In Arkin's approach these are potential fields. In grasping, these are object frame wrenches projected along the task basis.

- *Working approaches to reactive control employ more than local information*

Though reactive behavior is defined as actions based on current perceptions only, from this study we observe that working models employ more than local information. I think there is a reason for this:

In Grupen's approach a purely reactive approach would have terminated by placing the contacts at one of the local minima. In Arkin's approach this problem still remains in the presence of cluttered obstacles in the composite field. By employing a low pass global model Grupen uses the apriori information to construct trajectories free of local minima. In Arkin's approach this is specified by the vertex graphs. I therefore submit that reactive approaches can fail, in the presence of stable local minima that arise due to the complexity(non-convexity) in the environment. This entails the use of a global strategy and apriori information serves to reduce this uncertainty by reducing the complexity of the world. This is done by planning safe trajectories from the knowledge about the world.

- *Reactive approaches may fail*

Reactive behavior is akin to valley descent and is an opportunistic process. Actions based on current perceptions determine a transition to the next best state. This is a potential problem if local minima are present in the state space. Hence, reactive approaches may require global information in the form of plans or safe trajectories.

- *Composition via parallel activation*

It can be said that Grupen's architecture uses primitive behaviors. The reduction in error term results in a contact migration. This is one behavior, Secondly, the hand is adjusted to comply with the contact migration. This is a second primitive behavior which we have termed as the conditioning behavior. No composition is necessary since both these behaviors are necessary to achieve a particular task.

- *Reactive approaches respond to temporal constraints*

Grupen's technique can also respond to temporal constraints. By restricting the model refinement process within the temporal constraints imposed, a solution is obtained albeit at a lower level of competence.

Another approach to modeling the world is described in [Y. Gat,1991], where a generalized Vornoi diagram, using a medial axis transform gives a skeleton representation of the free space. This approach is based on the assumption that exteroceptive description of the trajectory is sufficient and that there is no necessity to describe the trajectory in proprioceptive terms, to produce an executable trajectory. The skeleton model is used to construct a graph in which the vertices are junctions and the edges represents paths. The graph thus encapsulates the free space. A search is performed on the graph to generate an optimal path. However, I do not think this paper is of direct consequence to reactivity as the title of this paper states. By assuming sufficiency of exteroceptive description the authors implicitly ignore proprioceptive requirements which is fundamental to building a reactive system. They do describe path following using their model of the world but based on the assumptions that there are no limitations either in the sensory quantity or quality. A third aspect is that they do not consider presence of dynamic obstacles. I believe this could mean recomputation of the graph which is expensive. They have devised a simple and elegant procedure but Gat's model is applicable for very simple domains where reactive behavior is not required. I have used this paper to bring out certain implicit properties of the the term reactivity, by citing what in general need not be reactivity .

- *Potential fields as a model for interaction*

The technique that Grupen suggested by using models, and task specifications that are consistent with actuator commands is termed as analogic planning [Bizzi,1991]. Khatib's work on artificial potential fields is an example of such planning methods. Khatib's technique has been used as a real time control method for manipulators. This paper is significant from the perspective of reactivity because, of it's utility as a model for interaction in domains of study such as robotics. Khatib's method represents objects by their attributes, i.e. analogicaly. The attribute is a potential field ascribed to obstacles or goals and since potential fields are linearly superposable, composite behavior is determined by by the linearly superposed potential field. The problem is the presence of local minima that may arise. In this sense a reactive approach may fail and global techniques may have to be employed. These issues are studied in Khatib's paper described below. Khatib, [Khatib,1985] develops a method for real time obstacle avoidance for manipulators and variations of this method have been used in a number of papers. Some of them that are also important from the perspective of reactivity have been included in this study such as [Payton,1990, Arkin,1988, Arkin,1987] . Khatib's paper is important to this study because it defines a method for modeling the world and this can be developed, as shown in this paper, into a reactive control strategy.Since, this is a method to model the world, it can be applied as an algorithm that construct trajectories to the goal or one that avoids obstacles or both. A brief description of the paper is given below:

The end-effector equation can be expressed as

$$\frac{d}{dt}\left(\frac{\delta L}{\delta \dot{x}}\right) - \frac{\delta L}{\delta x} = F$$

, Where $L(x,\dot{x})$ is the Lagrangian and

$$L(x,\dot{x}) = T(x,\dot{x}) - U(x)$$

where,

$$T(x,\dot{x}) = \frac{1}{2}\dot{x}^T\Lambda(x)x$$

and $\Lambda(x)$ is the symmetric kinetic energy matrix. U(x) is the potential energy. This is defined in a space that is a set x of m_0 independent parameters describing the end-effector position and orientation in a frame R_0 . Further,

$$\tau = J^T \cdot F$$

, where F is the command vector (in cartesian space) and τ the joint torques. J is the Jacobian matrix. Khatib describes the equivalent form of the decoupled end-effector equation i.e.

$$F = \Lambda(x)F^* + \mu(x,\dot{x}) + p(x)$$

where, $\mu(x,\dot{x})$ represents the centrifugal and coriolis forces and p(x) gravity. F^* is the command vector.

The artificial potential field approach is illustrated as follows:-

> The manipulator moves in a field of forces. The goal is an attractive pole and obstacles are sites of repulsive forces

Now, consider a goal x_d and an obstacle O. Then,

$$U_{art}(x) = U_{x_d}(x) + U_O(x)$$

and then U(x) as used in the Lagrangian is given by

$$U(x) = U_{art}(x) + U_g(x)$$

where, $U_g(x)$ is the potential due to gravity. The command vector F^* therefore is given by

$$F^* = F^*_{x_d} + F^*_O$$

where,

$$F^*_{x_d} = -\nabla U_{x_d}(x)$$

and

$$F^*_O = -\nabla U_O(x)$$

Now,

$$U_{x_d}(x) = \frac{1}{2}K\left(x - x_d\right)^2$$

and K is the position gain. The artificial field is defined as:

$$U_O = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)^2 & \text{if } \rho \leq \rho_0 \\ 0 & \text{if } \rho > \rho_0 \end{cases}$$

Here, η is a constant gain, ρ_0 is the limit distance of influence of the potential field and ρ is the shortest distance to the obstacle O . A point subjected to this potential (called psp) experiences a force

$$F^*_{O,psp} = \begin{cases} \frac{1}{\rho^2}\eta\left(\frac{1}{\rho} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2}\frac{\delta\rho}{\delta X} & \text{if } \rho \leq \rho_0 \\ 0 & \text{if } \rho > \rho_0 \end{cases}$$

and

$$\frac{\delta\rho}{\delta X} = \left[\frac{\delta\rho}{\delta x}, \frac{\delta\rho}{\delta y}, \frac{\delta\rho}{\delta z}\right]^T$$

The joint torques can be computed for the psp using,

$$\tau_{(O,psp)} = J^T_{psp}\Lambda\left(x\right)F^*_{(O,psp)}$$

The objects are modeled using primitives which are convex, such as the point, line, plane, cone and cylinder. Thus for an object, the force acting on it is the sum of the forces acting on all the primitives that are used to compose the object. This is given by

$$F^*_{O,psp} = \sum_p F^*_{p_p,psp}$$

,where p_p is a primitive. for several obstacles we sum the forces for all the obstacles. This is,

$$F^*_{psp} = \sum_p F^*_{O_i,psp}$$

where O_i is an obstacle. There are in general several points to be considered along a link. The joint forces for the obstacles over all these points is given by :

$$\tau_{obstacles} = \sum_j J^T_{psp}\Lambda\left(x\right)F^*_{psp_j}$$

Further, joint limits can be avoided using the same concept. Let $q_{i,min}$ and $q_{i,max}$ be the minimal and maximal bounds of a joint q_i . Then

$$\tau_{q_{i,,min}} = \begin{cases} \frac{1}{\rho^2_{i,min}}\eta\left(\frac{1}{\rho_{i,min}} - \frac{1}{\rho_{0,i,min}}\right)\frac{1}{\rho^2_{i,min}} & \text{if } \rho_{i,min} \leq \rho_{0,i,min} \\ 0 & \text{if } \rho_{i,min} > \rho_{0,i,min} \end{cases}$$

$\tau_{q_{i,max}}$ is defined similarly. This method fits potential barriers at the hyperplanes $q_i = q_{i,min}$ and $q_i = q_{i,max}$. $\rho_{0,i,min}$ and $\rho_{0,i,max}$ are the distance limits of the potential field and $\rho_{0,i,max} = q_i - q_{i,max}$ and $\rho_{0,i,min} = q_i - q_{i,min}$ The joint torque then is given by

$$\tau = \tau_{motion} + \tau_{obstacles} + \tau_{jointlimit}$$

τ_{motion} is given by the applying the Jacobian and the energy matrix to the $F^*_{x_d}$ i.e. matrix product. The coriolis ,centrifugal and gravitational forces also need to be considered for the joints in computing τ . This completes the description of the paper. The following observations can be made from Khatib's paper:

1. Given the shortest distance to an obstacle and from the knowledge of the obstacles range of influence, force and hence the joint velocities can be computed in real time.

2. Given a potential surface a path can be computed by computing the the net joint force (or torque) continuously along the field.

3. This method may fail when the obstacle space is cluttered. Also, local minima can exist in the potential field when a number of obstacles are present. Therefore, the manipulator may not reach the goal.

4. Khatib demonstrates that using a potential field approach an agent can reactively find a path to the goal. This is because the gradients are assumed to direct the manipulator to the goal. However due to the problems mentioned in the last point a result cannot be guaranteed using a purely reactive approach. In order to reach the goal global methods must be applied to find an alternate path. This suggests that reactive approaches can in general fail due to local minima being present in the state space. Thus, there is a requirement for a plan, that is based on a global observation of the world. This point was mentioned in out discussion on Brook's and Payton's work. At the same time the pitfalls of classical planning based on a world model

(which cannot be complete for a real world) are known. [Payton,1990] [Kaebling,89] explain this concept in detail and propose a *plan guided reaction* as a paradigm for intelligent systems.

- *Generating robust models of the world incrementally*

The primary problem with using potential fields to model the world is that the potential field may contain local minima.
Connolly, [Connolly,1992] proposes a technique based on harmonic functions to implement potential field path planning. These functions produce smooth vector fields which are free of local minima. This paper describes the implementation of velocity based control using harmonic functions. This paper shows that paths constructed from models of the world based on harmonic functions will always lead to the goal, since they lack spurious local minima. However this still requires that the environment be modeled completely. If the manipulator encounters an unknown object (not modeled) in this environment then this algorithm may potentially fail, as will be clear from the study of this paper.

Harmonic functions are solutions to the Laplace's equation

$$\sum_{i=1}^{n} \frac{\delta^2 \phi}{\delta x_i^2} = 0$$

The properties of a harmonic functions are:

- Lack of spurious local minima
- Linearly superposable
- Continuity and smoothness of C-space trajectories
- Robustness with respect to geometrical uncertainties
- Completeness up to discretization error

The first property ensures that the gradient vectors or *streamlines* will always terminate at the goal. Linear superposition is suggestive of the nature of composition that may be applied, given a set of harmonic functions. The third property ensures that gradients can be computed. This is true for two reasons : If the world model contains spurious minima, the computation of a harmonic function smoothes them out, secondly in the face of unmodeled processes introduced into the geometrical model (an unknown object) harmonic functions provide a platform for strategies that could still take the manipulator to the goal. This paper concentrates on such a strategy called *equipotential following.* Completeness up to discretization error is an issue of sampling limitation. This means that in the discrete representation of the C-space (as a grid) objects smaller than the sample size cannot be identified, unless the object happens to lie on one of the vertices of the grid. However all objects that are larger than the sample size can be identified and accordingly relaxation is not performed over these objects (harmonic functions are compute via relaxation over the free space in the grid) and hence is complete.

Harmonic functions are computed using relaxation over the free space in a grid. After convergence, the value of the function (representing the C-space) in the free space portion is an average of the values at the grid neighbor. The only extrema occur at the obstacles and the goal point(s). This process is applied when the boundary potential of the surface are fixed i.e.,

$$\phi_{\left(x|_{boundary}\right)} = c$$

The harmonic function used to compute a path is a potential function denoted by ϕ . Streamlines are trajectories obtained by following the gradient of this function, denoted by ψ . Equipotential surfaces are orthogonal to the these streamlines every where. Since there are no local minima, these streamlines always terminate at the goal (the global minima) point. Unmodeled obstacles, which are a source of uncertainty, obstruct the streamlines, there by cutting of the path to the goal. The solution according to this paper is to use the equipotential surface to transfer to an alternate streamline and continue towards the goal. Let $v_\psi = c\nabla\phi$ be the velocity vector along the stream line. Further, the obstacle exerts a contact force *f*, which can be observed from the normal component f_n and a frictional component f_f generated at the contact site. Since we know that the equipotential surface is orthogonal to the streamline, the dot product of the new velocity vector along the equipotential surface to a new streamline v_ϕ and the gradient vector of the potential function is zero. i.e. $\nabla\phi \cdot v_\phi = 0$. This is not sufficient to compute the new velocity command. Connolly, uses the following equation to determine the new velocity command.

$$v_\phi = c\,(v_\psi \times (f \times v_\psi))$$

where c is the speed limiting factor. This produces a velocity such that $v_{phi} \cdot v_{psi} = 0$ satisfying the above constraint. The path indicated by this velocity is followed, till a new streamline is found.

Connolly's paper provides a simple and elegant method to implement path planning using velocity control. The use of harmonic functions rids the world model of local minima and the manipulator can reactively progress to the goal. However this method can fail. It fails when the above cross product returns a zero magnitude, which occurs when f is tangent to the streamline currently being followed (i.e. the surface at the contact site is almost normal to the streamline, assuming a very small frictional component). Since the velocity commands are computed in real-time, Connolly's method provides a way to reactively control the manipulator's motion. This has been possible because of a robust world model and real-time algorithm, to cope with uncertainties introduced by unmodeled obstacles.

We have observed composition of behaviors via subsumption, priority based arbitration, parallel activation(no composition) , and linear superposition of primitive behaviors. In the next paper that is described here interaction models are treated as vector fields instead of potential fields. It is true that a potential field has a vectorial equivalent, but constructing a potential field from a generalized vector field is may result in complex representations. Consider for example a simple circulating pattern of vector fields. It's equivalent potential function would be complex as Bizzi in [Bizzi,1991], argues in his paper presented below. Further, in this paper the repertoire of behaviors is considered to be limited. This repertoire is used to approximate a planned field, which may be constructed from the local perception to generate the actuator commands. Bizzi's method suggests composition based on summation of vector fields . This paper is described below.

Bizzi et al in [Bizzi,1991] consider the issue of transforming motor plans into actions as a field-approximation problem. Their approach is based on the use of force fields as a representation of the manipulator's interaction with the environment. The manipulator is considered to be operated by a set of impedance controllers that operate in parallel. Each controller modulates an output-force field according to a predefined control law. Given a plan that is specified as a vector field, the problem is to approximate this plan by fields generated by the controllers. The fields generated by the controllers are elementary behaviors termed basis fields. Note that the problem is one of approximation because, the manipulator's repertoire of basis fields is limited by the structure of the controllers. Therefore at best an approximation can be achieved. This paper therefore demonstrates a technique to compose primitive behaviors (which are controllers that operate the manipulator) to approximate a plan. Note that the plan is specified in the same domain as that of the controller responsible for motor actions, i.e. as force fields. This was observed in [Grupen,1991b, Khatib,1985, Connolly,1992]. We now proceed with a brief description of the method suggested in this paper: A manipulator is a mechanical system. Bizzi restricts his discussion to time invariant systems and the output of a time invariant system is expressed as

$$y = g(x, u)$$

where y is the output, x the state and u the input. The output may be an effort variable. In the case of a manipulator the output effort is a vector variable. If the input is held at a constant value the output equation describes the behavior of a mechanical impedance:

$$y = g_0(x) = g(x, u_0)$$

This impedance can be designed to be passive so that the manipulator's behavior is equivalent to that of a passive physical system. Bizzi in this paper deals with the static component of the impedance which describes the relation between effort and position at steady state. As stated above the output effort of a manipulator is a vector variable. Therefore, the static impedance established by the value of the input (a vector in the manipulator workspace) is a field whose domain is the work space. If U is the set of all possible input values, the *repertoire* is a set of output fields generated by all possible input values.

$$X = \{y = g(x, u) \,|\, u \in U\}$$

Now, the planner specifies a field that represents the desired impedance of the manipulator at a set of work space locations. Call this P(x). The objective is to find a field $F(x) \in X$ which minimizes the norm

$$\|P(x) - F(x)\|^2$$

defined over the manipulator's work space. This derivation is studied in greater detail below:

Let $x = (x_1, \ldots, x_N)$ represent a generic point in the manipulator's work space. The planned filed is expressed as a finite set of D force vectors (as opposed to a continuous field to make the problem easier) which are represented as $\{P^i\}$ $(i = 1, \ldots, D)$ associated with D work space locations. The character of the vector field features are shown in Let the manipulator be operated by K impedance controllers each operating a joint stiffness function g(q), where q(x) the inverse kinematic function The force modulated can be determined by $Q_i = u_i g_i(q)$ Now let

$$\phi_i(x) = J^T g_i(q(x))$$

Then the end-effector torque field is given by $F_i = u_i \phi_i(x)$ where u_i is a control input. The end-effector repertoire is the linear span

$$X = \left\{ \sum_{i=1}^{K} u_i(x) \phi_i(x) \right\}$$

The task now is to minimize

$$\sum_{i=1}^{D} \|F\left(x^i\right) - P^i\|^2$$

If P is rewritten as $\hat{P}$ such that

$$\hat{P} = \left(P_1^1, \ldots, P_1^D, P_2^1, \ldots, P_N^D\right)$$

then $\hat{P}$ is expressed in the form

$$\Phi u = \hat{P}$$

Now if ND= K and Φ is not ill conditioned then a solution for u exists, However since the repertoire is limited in general a least square solution for u is obtained by alternate methods. Φ determines the values of the basis fields at D locations.

Bizzi has shown that the additive property of vector fields can be used to determine a net field that approximates a plan by the appropriate choice of control inputs. Contrast this with the potential field method based on scalar fields as proposed by [Khatib,1985] There it was assumed that the gradients along the planned field could be implemented directly by the manipulator. Given that the basic fields are irrotational then each basis filed is a gradient of a potential function. The repertoire is the linear sum of these potential functions. The advantage in using a vector field approximation approach is that they are more general. This is because, a simple set of rotating fields would require a complex potential filed surface with multiple minima and maxima. Then approximation by the irrotational basis fields may result in a large error.

Let us briefly sum up the study to this point.

- A reactive architecture may be viewed as a composition of reactive behaviors. Each behavior perceives the local world and responds to this perception.

- Representation of the local world and task in a domain consistent with the space of actuator commands, facilitates the construction of a model for interaction with the world.

- As cited in Khatib, reactive approaches may fail in the presence of local minima in the state space. In such a situation a global search may have to be employed to find a solution. This compromises the requirement for real time performance.

- In the papers studied in this section, behaviors were composed by executing them in parallel (as in [Yamauchi,1991], [Grupen,1991a]), i.e these behaviors were mutually exclusive and are collectively inclusive, based on a linear superposition of primitive behaviors. (as suggested by Arkin, Khatib), and superposition of vector fields, that approximate a plan [Bizzi,1991].

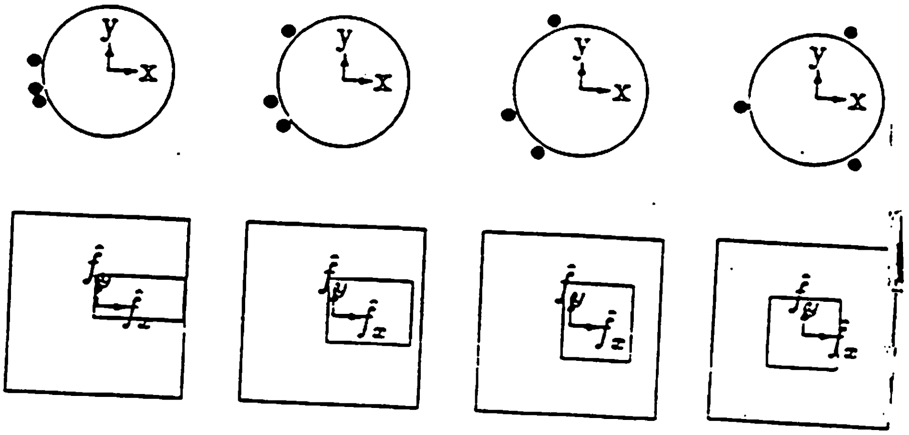


Figure 12: First harmonic models are sufficient for ellipsoidal objects

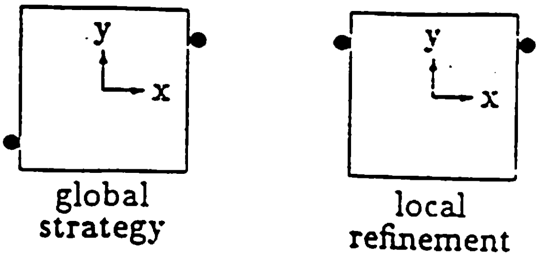


Figure 13: First harmonic models may not be sufficient

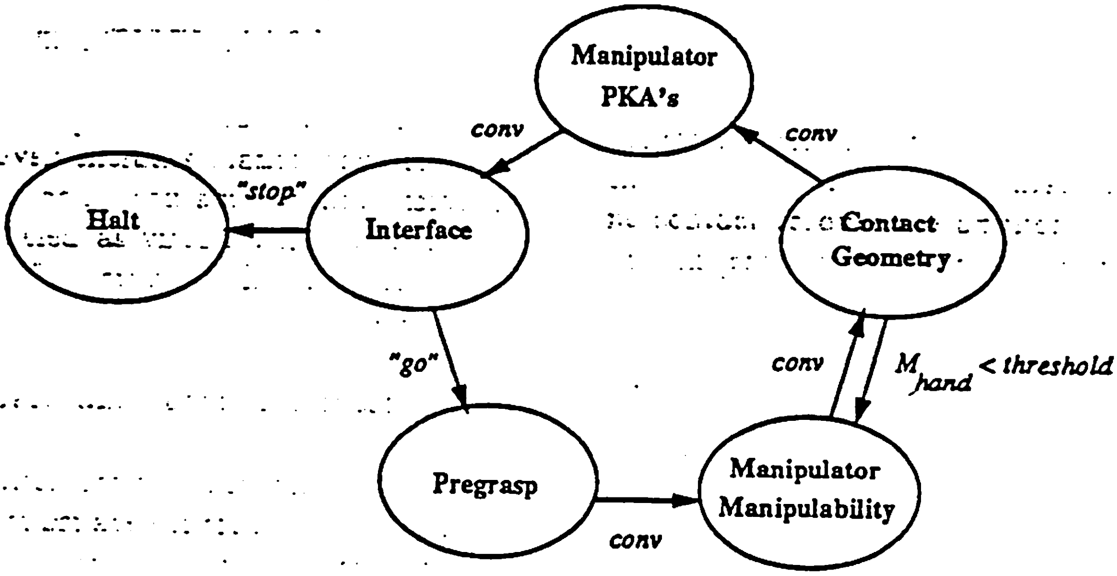


Figure 14: Reactive Architecture as a Finite State Machine

2.3 Reactivity and Planning

We have seen that planning is sometimes impractical for it is not timely. Further planning depends on a complete world model, that cannot be constructed due to the incompleteness of perception. Reactive approaches aim to offer a solution by generating timely actions. They also make plans robust by accounting for unmodeled processes. However reactive approaches may fail to reach the goal state based purely on current perceptions due to the presence of global minima. This is because, reactive approaches are ***short sighted*** since they take opportunistic advantage of the current situation to chose the next step. This is akin to the disadvantages of hill climbing. Therefore, these approaches require more than local information to generate successful solutions. This suggests an effective merger between reactive approaches and deliberative approaches may be required. These issues are studied in this section, in [E. Gat,1991, Payton,1990, Kaebling,89].

Gat's paper [E. Gat,1991] extends the arguments presented in [Chrisman,1991] by showing that the internal state is necessary for coping with incomplete perception. He discusses the problem associated with maintaining internal state and suggests ways to overcome them. He also demonstrates through a control architecture, that neither a purely reactive approach nor planning in the on-line sense, where planning takes place during task execution, are solutions to design problems associated with intelligent systems. Instead, the solution may be to maintain internal state, construct plans at a high level of abstraction that guide the actions of an agent, not control them. The control of actuators is reactive, which is a moment by moment control of the robot's actuators.

Online planning according to Gat is problematic for three reasons:

1. Planning is *time consuming.*

2. Planning requires a world model at a level of *completeness* and fidelity that sensor technology cannot provide at the moment.

3. The *world state may change* while the robot is planning.

These problems manifest themselves only when the information maintained in the internal state does not match the current world state. If a world model is feasible and correct always, then on-line planning is perfectly feasible. Therefore, any time consuming computation would encounter problems if the world state changes while the computation is going on.

Gat points out that reactive approaches try to eliminate time consuming computations and minimize the amount of stored information about the world. However, viewed this way, avoiding internal state is tantamount to assuming that very few things about the world can be usefully predicted. This is not the case either. Gat's solution to this problem is to use internal state and represent plans at a high level of abstraction. Abstraction is an inverse measure of precision of description. Abstraction is useful in making predictions with a greater degree of success because, they are unaffected by small variations in state. However plans may go wrong even when stored at a high level of abstraction.

The ATLANTIS architecture developed by Gat et al., has three components: the controller, the sequencer and the deliberator. The controller is a purely reactive control system that is responsible for a moment by moment control of the robot. The sequencer is responsible for selecting which of the several transfer functions the controller is to compute, as well as parameterizing the appropriately chosen transfer function. The sequencer is responsible for taking corrective actions in the event of failures. The deliberator is responsible for maintaining the world models and constructing plans to be used by the sequencer. The design methodology adopted is bottom-up. First a set of transfer functions, which produce a set of useful primitive behaviors is constructed. Deliberative computations are performed when information is required by the sequencer, that requires time consuming computations.

Gat's paper in essence argues the following;

1. Internal state is required to perform on-line planning

2. Planning must be done at an abstract level to ensure a higher level of predictability

3. Since the robot must deal with a real world, Plans must act as a guide to the robot's actions.

4. Actions are controlled by a sequence of behaviors.

5. A mechanism must be provided to detect and recover failures.

Gat in [E. Gat,1991] has suggested that the shortcomings of classical planning do not mandate the design of a purely reactive system. Instead an alternative is to design plans at a high level of abstraction, that guide the reactive unit to perform actions. Gat makes a lot of intuitively convincing arguments but does not provide justifications to his statements. For example he does not explain how failures are handled. The explanation of ATLANTIS is very brief. He does not explain how behaviors can be composed. He also does not explain, how planning at a high level of abstraction may guide the reactive component. I make this statement because, the greater the degree of abstraction the harder would be the communication problems between the planning and reactive sub-systems. However Gat's paper serves an important guide at a systems level. In this paper and in every paper we have observed that pure reactivity is insufficient. We have also argued that planning is impractical at times. Gat's paper actually sums up these observations. Most of the architectures studied at the beginning in section 2.1 also use a similar hierarchy to the ATLANTIS architecture. Since these observations also surface in the papers discussed elsewhere I believe these arguments to be true and that indeed the solution to building an intelligent system may be in developing a hierarchy of reflexive, reactive and deliberative behaviors , with increasing immediacy, reduced reasoning and abstraction as the character of this architecture when progressing down the hierarchy.

In the previous paper studied Gat argues about maintaining plans at a high level of abstraction and to use plans to guide behavior. In this paper Payton et al., convincingly argue that while plans must guide and not constrain behavior, abstraction of plans results in loss of information along the pathways in the systems hierarchy. They instead suggest that plans must not abstract information so as to hide them but make them accessible, so that they can be used to allow for opportunistic action, which contributes to the intelligence of the system.

The authors site two cases where such abstraction leads to disadvantages. The first is at the behavior level. In a previous paper discussed in this letter in section 2.1 [Payton,1986] Payton argued that the arbitration of behavior is based on a priority. This fusion of behavior commands termed command fusion supplants sensor fusion and avoids the delays caused by sensor fusion. However, by arbitrating, a great deal of information that is contained within each individual decision process, is lost. The process of selecting an appropriate command in many cases requires that several alternatives be weighed. When multiple behaviors are evaluating their alternatives on the basis of different criteria it may be impossible to arrive at a compromise. This inability as stated above is because a great deal of information contained within each behavior is unavailable. Consider the following example from this paper. A robot is in motion and is currently following a road. There is an obstacle on this road that partially obstructs the path. The robot is endowed with a follow road behavior and avoid obstacle behavior. Both these behaviors are capable of giving turn commands, as part of their decision making process. In this case it is required that the vehicle move left in order to avoid the obstacle and stay on the path. However, the arbitration mechanism completely subsumes the road following behavior and the robot turns right veering of the road. The problem of information loss occurs from modularity as much as it occurs from arbitration. Modularity typically hides the internal state within behaviors. Hencë the authors argue that information is lost when it is hidden or contained in commands which are subsumed.

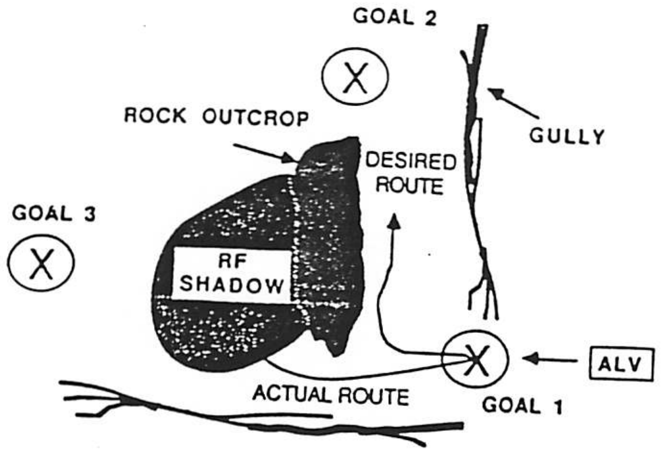


Figure 15: The terrain and goals.

The second reason for abstraction to be disadvantageous is because it blocks information flow between the different levels in the control hierarchy. More specifically it prevents the system from taking opportunistic advantages from unexpected situations, when such situations are not accounted for in the abstract plan. Consider an example from the paper. The robot must move from the current position to goal 3 . This is specified in Figure 15 . The abstract plan is depicted by a set of intermediate goals in this case goal 2. The robot must avoid the RF shadow in order to maintain communication with the host. From goal 1 the robot was unable to turn towards goal 2 because of local obstacles. This was an error in the geometric model but the abstract plan did not highlight a possible problem. Fortunately, the RF contact was not lost, when the robot moved along the indicated path. However, the agent persisted in moving to goal 2 from this point on and soon RF contact was lost. It is clear that the agent should have progressed to goal 3 because that is the objective. The abstract plan failed to give an indication that goal 3 should have been the appropriate course of action. The system could not therefore take opportunistic advantage of this unexpected situation.

Both these problems arose due to the information lost due to abstraction. The authors propose solutions to each of these problems. First they make the behaviors as fine-grained as possible to minimize their internal state. Secondly the flow of meaningful information can be maintained across levels if plans serve as a resource for advice within a single system, rather than as constraints imposed by one level on another. Each of these is discussed below.

In the case of behaviors the authors argue that instead of modules with internal states and instance variables, behaviors are comprised of atomic functional elements that have no inaccessible internal state. These simple decision making units and their interconnections together define a behavior. Each unit is a specific concept which the designer establishes through carefully chosen connections and the functions for each unit are designated based on the desired output characteristics for that unit. Arbitration then is distributed and unlike the arbitration scheme is expressed as a fine-grained connectionist architecture. Consider the road following example again. See figure 16. The turn for obstacle, is constructed as an interconnection of five decision processes .

Each unit outputs a value that corresponds to the desirability of that unit. The desirability of each turn command is now determined by fusing the outputs of the two behaviors. The turn rate is now determined by applying a function such as addition of the inputs. In this case a hard left is chosen in a winner-take-all network. This organization is also flexible because new units can be added without affecting the existing network. Thus, the connectionist model makes it possible to simultaneously satisfy the constraints of multiple independent behaviors. Payton et al also argue that in this manner a hierarchy of behaviors can be developed in the style of Brooks' architecture, but each new behavior is allowed to bias the decision instead of subsuming existing behaviors.

Payton et al suggest that plans must not obstruct or constrain the local decision making process but guide them. In order to do so, all relevant

knowledge must be organized and then without any further abstraction must be presented in full for real time decision making. The plan must therefore be viewed as a resource and not a program for action. As a resource plans must serve as sources of advice to agents which are already competent in dealing with the local problems. In this sense they can be used optionally to enhance the system
performance. The original state space in which the plans are formed is retained enabling the plan to provide advice whenever the current state of the system can be identified within that state-space. Such plans are called ***internalized plans*** which embody the search and look ahead of traditional planning without providing an abstracted account for an explicit course of action. Payton suggests that in the robot navigation domain one way these plans can be constructed is by representing them as is a gradient description. This is shown in figure 17 .

In this case there is no explicit route plan yet the vectors determine the the best course of action that may be taken. In the RF example, the field biases the turn decision against reaching the bottom of the out crop. However if local circumstances force the robot to move towards the bottom of the outcrop it will do so. At first the gradient field data may be unavailable to guide the robot's motion once it reaches the RF shadow area. Local sensing would cause the vehicle to possibly move on and pick up the gradient field again. The gradient field is an ideal example for internalized plans because the state space in which the original problem was formulated is the same state-space in which the plan is represented. Another method is based on the use of potential fields. These methods may seem similar but the difference lies in the construction methodology of these two representations. As we know a potential field model is a superposition of charges with the obstacles being repulsive sites and goals being attractive sites. The vector at any point is computed by summing up the charges. The resultant field however may contain local minima. (See discussion on Khatib's paper). In contrast the gradient field is constructed through a time consuming graph search process. This field does not contain local minima and will always yield optimal paths to the goal. Secondly potential fields are used to control the robot directly. Gradient fields are not. However, internalized plans provide an efficiency of interpretation as at the cost of efficiency of space.

Payton's approach is significantly different from
the classical view adopted. He has argued and suggested the following:

1. Abstraction results in loss of information

2. Behavior composition through arbitration does not take into account the decision making process of other behaviors

3. composition of behaviors is expressed as a network of functional units in which each unit biases the final decision. Contrast this with subsumption in which a higher layer of control arbitrates by subsuming a lower layer. This as Payton points out results in loss of critical information that is necessary to determine the right action.

4. Plans are resources that should guide action not constrain them

5. Reactive architectures may be represented as a set of behaviors which are composed using a connectionist approach

Rosenschein and Kaebling [Kaebling,89], in their paper argue that both planned and reactive approaches have advantages and suggest a way of integrating both of them. They argue that since planning is essentially *atemporal* it would be advantageous to incorporate a planner into a reactive architecture, whose response is time bound rather than embed a reactive controller in a classical planning system. The authors argue that in building such a system it is important to characterize the semantics of the inputs and outputs of the planning system. These aspects are studied in this paper.

Rosenschein et al argue that classical AI views generation of behavior as a two step procedure. That is planning and execution. Planning produces a data structure; execution is a step-by-step interpretation of this data structure to produce overt behavior. Planning provides a declarative formalism, that shifts the burden of reasoning to the machine. Hence in principle the control system can handle behaviors that are too complex for the programmer to anticipate. The problem however is that, since traditional planning may be considered as a guided search through a space of plans, then depending on the combinatorics of the search which in turn depends on the complexity of the domain, this process may not succeed in generating an output in time.

In contrast reactive control offers the advantage of guaranteed response and hence the ability to quickly react to a changing environment. The control system is modeled as reacting directly into the current situation. The behavior of the agent is specified in such a system as situation action rules that are evaluated at frequent intervals. Infact we may think of this as a program executing a tight loop, which exhibits high conditionalty. Since these conditions can be evaluated in parallel, reactive systems can be described as circuits or networks, implementing a function, that maps a stream of information from the input to a stream of output commands to the effectors. The problem is to design this function so that computation may be performed efficiently and repeatedly.

Rosenschein et al therefore argue that since both these approaches are advantageous and neither approach dominates the other they must be integrated. One method as they argue is to embed a reactive architecture into the classical planner-base architecture. Note that this can also be referred to as execution monitoring. This approach takes the view of downloading a set of specifications and constraints that must be implemented through a composition of primitive behaviors. However it is not clear how the time required for planning can be accounted for. Therefore they argue that it may be advantageous to embed a planner in a reactive control system. The authors then discuss as to how the semantics of the inputs and outputs of the planner may be implemented. They propose that the the machine be decomposed into three subsidiary variables *init, goal, plan* and four machines E_{init} E_{goal} *Planner* and *Execution* . E_{init} maps the input to a set of initial conditions. E_{goal} maps the input to a goal , *Planner* maps the goal and initial conditions to a plan and *Execution* maps the plan to an output. What makes this different is that the *Planner* and *Execution* units respond to the input information concurrently. It may be possible that the *Planner* may not have an output, but the *Execution* unit still responds to the input information. The *Planner* continuously generates a plan depending on the observation of the world. The value of a variable provides information about the external reality. For example consider the use of a data structure to describe facts about the world. This data structure would be of little use were it not that the value of the data structure gives conditions that currently hold true in the environment. Thus they argue that if a particular variable has a value then this value contains information about the real world and any functional unit must exploit this information. The *init* variable gives information about the conditions that hold true in the world to the planner, the value of the *goal* gives information as to what must hold true. Then the Planner outputs a *plan* variable which is a plan that gives information as to what conditions must be achieved. The Execution unit interprets this information in addition to its current perception to determine a course of action so that the plan may be satisfied.

The concept that comes out in this paper is interesting. However, I do not think functionally embedding a reactive unit in a planner is different from this scheme. Rosenschein et al. argue that by embedding a planner in a reactive unit, real time performance can be generated. This is true but not necessary. For example [Arkin,1988], [Brooks 1986], [Payton, 1986] have also argued for real time behavior by either constructing behavior hierarchies or by embedding a reactive unit in a planner. The issue here is the time required for plan generation. Behavior is observed in [Arkin,1988], [Brooks 1986], [Payton 1986] even in the absence of a plan albeit the behavior may not be 'competent'. Extending this argument to the authors proposal, we observe that the output of the execution unit is incompetent till the time that an appropriately instantiated plan variable is available. Therefore, this architecture does not offer any specific advantage.

3 Observations

In this letter, I have presented several features of papers related to reactivity. The objective has been to understand the definition, scope and limitations of reactive systems. Intelligent behavior as Brooks points out may be the manifestation of many simple processes interacting and coordinated by the environment. This study has shown reactivity to be one such approach. While reactivity does contribute to intelligent behavior, this approach has certain short comings. This section tries to encapsulate all the papers studied in this letter under various heads presented below:

- *On the character of the real world.* The real world is an interaction of multiple independent processes or agents. This interaction contributes to the dynamics and complexity of the environment. State changes in the world occur not only due to the effect of the action of the agent(s) that we wish to control but due to the actions of other agents over which we have no control. Therefore, we cannot predict the dynamics of the real world. This is true of most papers studied. This character and working definition of the real world is preserved in [Brooks, 1986], [Payton,1986, Payton,1990],[Payton,1986, Payton,1990], [Arkin,1988, Arkin,1987], [E. Gat,1991, Chrisman,1991, Kaebling,89].

- *On Perception.* The real world cannot be observed completely. This incompleteness of perception arises not only due to the unpredicdability of the real world but due to sensory limitations. These limitations in part arise because of sensor noise, monetary costs of sensing, the presence of physical obstructions, computational costs of sensing and mutually exclusive sensing. In all the domains that have been studied and presented in this letter perception is incomplete. This is true of

[Brooks, 1986],[Arkin,1988, Arkin,1987], [Payton,1986],[Payton,1990],[Grupen,1991b], [E. Gat,1991],[Kaebling,89] .

- *On modeling the world.* The real world cannot be completely modeled because it cannot be completely observed. Since a complete world model cannot be constructed the real world is said to be uncertain. This is true of [Brooks, 1986], [Payton,1986], [Payton,1990] ,[Arkin,1988], [Arkin,1987] ,[Grupen,1991a].

- *On the contraindications of planning* The value of horizontally decomposed planning based architectures is subject to the following issues:

  1. *Completeness of the world model.* The success of a plan hinges on the availability of a complete world model. As stated above this is not possible for a real world. Therefore, uncertainties in the world arise in the presence of unmodeled processes, which reduce the predictability of plans. As Chrisman points out expectation utilization is adversely sensitive to the predictability of the environment [Chrisman,1991] . In this sense these architectures are not robust [Brooks, 1986].
  2. *Combinatorics of planning.* Planning may be considered as a guided search through a space of plans. Then depending on the combinatorics of the search which depends on the complexity of the domain, this process may not succeed in generating an output in time [Kaebling,89]. Classical planning also assumes that the state changes in the world occur only due to the actions of the agent under control. This is not true of a real world. Therefore planning is essentially an *atemporal* process [Kaebling,89]. Architectures that are based on a horizontal decomposition scheme as studied in Brooks' paper [Brooks, 1986], are time consuming also because of the associated sensor fusion problem. In these architectures the sensors are used to generate a relatively complete view of the world and Brooks argues this is a time consuming process.
  3. *Flexibility and Robustness of horizontally decomposed architectures.*On a systems level, horizontally decomposed architectures are not flexible either. [Brooks, 1986]

The advantage of planning is that it shifts the burden of reasoning to the machine. Hence in principle the control system can handle behaviors that are far more complex than the programmer can anticipate, [Kaebling,89].

- *On Behavior.* Behavior may be defined as actions generated by a functional unit called *agent* in response to perception of the environment. Note that no behavior is observed till action is generated. Behaviors are known to be of three types:

  1. Reflexive behaviors: These behaviors are extremely responsive to time. Action may involve no internal state and is a result of a computation on the sensory inputs. These behaviors by themselves cannot execute a task [Payton,1986, Yamauchi,1991]
  .
  2. Reactive behavior: These behaviors generate action in response to a task from their current perceptions. They are responsive and can independently pursue a goal.
  3. Deliberative behavior: Action generated as a result of a global representation of the world. In this sense the horizontal decomposition of planning based architecture generates deliberative behavior.

- *On the definition of Reactivity.* Reactivity is defined as actions that are generated in response to the current perception of the world, [Chrisman,1991]. Reactivity has also been defined as circuits or operator networks, that map a stream of input information to a stream of output commands to the effectors [Kaebling,89]. There are other possible definitions. In a space where a point represents the world state, reactivity can be a gradient descent process, where action causes a transition to the next best successor state. This was more formally represented in [Khatib,1985], where the manipulator command was determined by the gradient of a function in a space of possible states. Perception defines the current state. As is also pointed out in [Grupen,1991a], the migration of a contact is an iterative process where the next contact configuration is one that results in a smaller state error than the current existing state.

- *On the advantages and disadvantages reactivity.* Reactive behavior is advantageous for the following reasons.

  1. It is responsive to temporal constraints. It can be used to attain real time performance [Brooks, 1986, Payton.1986]

, [Arkin,1988, Arkin,1987, Grupen,1991a, Yamauchi,1991, Khatib,1985],

[Connolly,1992, E. Gat,1991, Kaebling,89, Payton,1990]

2. Since actions are determined from the current perception of the environment, reactive approaches account for uncertainty in the world in the presence of unmodeled processes. This has been the motivation in [Brooks, 1986, Payton,1986, Arkin,1988, Arkin,1987, Payton,1990]. Thus reactivity suppresses bounded noise in the environment [Grupen,1991a]. For the same reason, reactivity increases the robustness of plans, by accounting for unmodeled processes [Payton,1986, Arkin,1988, Payton,1990, E. Gat,1991, Kaebling,89].

The disadvantage of reactivity is that it is an opportunistic approach. Therefore, action generated is one that causes a transition to the next best state. This results in a problem if the state space has local minima in it. In this space, the goal state is the global minima. This is pointed out in [Grupen,1991b], [Khatib,1985]. Connolly, [Connolly,1992] provides a solution to this problem, by using harmonic functions. These functions rid the state space of local minima, therefore gradient descent strategies would lead to the goal. Further, Connolly's method suggests that robust control results from incrementally refining the C-space models and thus, the quality of reactive control can improve with the addition of more than local information, fused over time. In general information may be fused over time to refine a perceptual state, but reactive forms of control realized as gradient descent still require convexity in the control surface, which limits the optimality (of the grasp configuration for example) in the resulting strategies. Refinements to the perceptual state may result both from information fused over time or from *a priori* global information (as in [Arkin,1988]).The failure of reactive approaches as a result of local minima in the state space may also require a more detailed search or a global search strategy. In this sense reactive approaches would require more than local information, to execute a task.

- *On Reactive architectures.* Reactive architectures have been expressed as

  1. Finite state machines ([Grupen,1991a]).

2. A behavior hierarchy ([Brooks, 1986])
3. A lateral decomposition of behaviors ([Payton,1986], [Payton,1990], [Arkin,1988]).

Hierarchical and lateral decomposition techniques, result in robust and flexible architectures. All the architectures respond to the real time requirements. Behavior decomposition is based on the designer's understanding of the domain. None of the papers studied describe a method to construct these primitive behaviors.

Each behavior perceives some aspect of the local world and determines the appropriate action. This solves the sensor fusion problem, because each behavior is mapped by design to a small set of sensors. Further, sensors could be shared [Brooks, 1986, Arkin,1988, Payton,1986].

Reactive architectures are context activated. This activation is in the context of a task to be performed using the current local perception of the environment. Second the working models also employ more than local information to reduce the complexity of the world. This has been discussed in grasping.

Each behavior determines action through a model for interaction. These models have been expressed as potential fields, [Khatib,1985, Connolly,1992, Arkin,1988] , as force domain models [Grupen,1991a]. Grupen states that by representing the task, the world in the domain of effector commands, interaction is facilitated between the perception and action. This is the representation scheme adopted in [Grupen,1991b, Khatib,1985, Connolly,1992, Arkin,1988]. The issue then is to compose these behaviors to generate the output command to the effector. We have seen that purely reactive approaches may get stuck at local minima in the state space. Also, during the study we noticed that all the papers studied need global information to resolve this problem. Behavior composition has been expressed as a sequence of behaviors constrained by a plan. [Payton,1986, Arkin,1988] In the reactive architectures studied in [Grupen,1991b, Yamauchi,1991], composition was observed as aggregate behavior. In Grupen's example successful execution of the task required both contact migration and hand compliance. In Yamauchi's paper all three behaviors are essential. So we may view the architecture as a network of operators, each operating on some aspect of the local world to determine actions that generate successively better states . The choice of behaviors are made using behavior composition techniques presented below.

- *On behavior composition.* The following types of behavior composition techniques have been studied.

  1. Subsumption: In a hierarchical decomposition architecture arranged as a hierarchy of behaviors, a behavior with a greater degree of competence can subsume the role of a behavior with a lower degree of competence [Brooks, 1986].
  2. Priority based arbitration: A more general form of the above approach, where each behavior places an integer priority and the behavior with the highest priority is chosen [Payton,1986].
  3. Distributed arbitration: In this case a connectionist architecture is employed. Each decision process is allowed to bias the final decision [Payton,1990].
  4. Aggregate behavior: All the behaviors are essential to the task completion and are in a sense mutually exclusive and collectively inclusive [Grupen,1991a, Yamauchi,1991].
  5. Linear superposition: Khatib has pointed out that if the world is represented as a potential field, then the gradient vector at any point determines the actuator command. This has been possible because of the property of superposition that holds true in potential fields. Bizzi has pointed out that the manipulator repertoire may be limited in such a case at best the planned field may be approximated. He uses a least squares approximation of the planned field to determine the control inputs for the basis fields.

- *On combining planning and reactivity.* The following techniques have been suggested.

  1. Represent plans at a high level of abstraction and use reactive behavior to control the effectors,[E. Gat,1991] . Plans must guide the reactive unit.
  2. Abstraction results in loss of information. Therefore, after the information is completely assimilated, it must be present without further abstraction, to guide behaviors, not constrain them,[Payton,1990
  3. It is better to integrate a planner in a reactive architecture than the other way around, because planning is an atemporal process. [Kaebling,89].

4 Conclusion

In this letter a small set of papers pertaining to reactivity has been studied. These papers have encompassed issues such as, incompleteness of perception , the contraindications of planning, behavior-based robotics, reactive architectures, advantages and properties of reactive systems, models of interaction with the world, behavior composition, disadvantages of reactive systems and a mechanism for integrating reactive and deliberative behaviors. All these issues have been studied and presented as part of this one semester study. Reactive architectures demonstrate the requirement for real-time action in the real-world. However, reactivity alone may not be able to find a solution when the world is complex. In such cases a time consuming combinatorial search for a plan in a space of plans, i.e. planning might be required to reduce this complexity of the world, by showing direction to this short sighted process. As Chrisman [Chrisman,1991], points out reaction is adversely affected by the complexity of the world and expectations fail when the environment is unpredictable.

References

[Arkin,1988] Arkin R. C. , " Reactive reflexive navigation for an autonomous vehicle, AIAA' 88.

[Arkin,1987] Arkin R. C., "Motor schema based navigation for a mobile robot", Proc. of the IEEE Intl. Conf. on Robotics and Automation, pp264-267, 1987 .

[Bizzi,1991] Bizzi E. et al. " Transforming plans into actions by tuning passive behavior: A field-approximation approach", Proc. of the IEEE Intl. Symposium on Intelligent Control, pp.101-109, 1991.

[Brooks, 1986] Brooks, R. "A robust layered control system for a mobile robot" , IEEE Jrnl. of Robotics and Automation. VOL. RA-2, NO.1, 1986.

[Connolly,1992] Connolly, C. I. , "Harmonic Control" , Submitted to IEEE conf. On Robotics and Automation, 1992.

[Chrisman,1991] Chrisman L et al. "Intelligent agent design issues : Internal agent state and incomplete perception.",AAAI,1991

[Chapman,1991] Chapman D., "Combinatorics and Action", AAAI,91

[E. Gat,1991] Gat E., "On the role of internal state in the control of autonomous mobile robots", AAAI,1991.

[Y. Gat,1991] Gat Y. et al., "Simple world modeling for Reactive Navigation", AAAI,1991

[Grupen,1991a] Grupen R. A., "Planning grasp strategies for multifingered robot hands", In Proc. of IEEE Intl. Conf. on Robotics and Automation, 1991

[Grupen,1991b] Grupen R. A. et al., "Force domain models for multifingered grasp control", In Proc. of IEEE Intl. Conf. on Robotics and Automation. ,1991

[Grupen,1991c] Grupen R. A. et al. "Sensor-based control for multifingered manipulators", AAAI,1991.

[Grupen,1991d] Missing reference

[Khatib,1985] Khatib, O. "Real time obstacle avoidance for manipulators and Robots", In Proc. of IEEE Intl. Conf. on Robotics and Automation, pp. 500-505, 1985.

[Kaebling,89] S. Rosenschein et al., " Integrating planning and Reactive Control", Proc. NASA Conf. on space telerobotics, vol.2, JPL publ. 89-7,JPL,CA pp. 359-366.

[Payton,1986] Payton D. W. "An architecture for reflexive autonomous vehicle control", Proc. of IEEE conf. on Robotics and Automation, 1986.

[Payton,1990] Payton D. W. et al., "Plan guided reaction", IEEE Transactions on Systems, Man and Cybernetics", vol.20, no.6,pp1370-1382,1990.

[Yamauchi,1991] Yamauchi B. et al., "A behavior-based architecture for robots using real-time vision", Proc. Of the IEEE Intl. Conf. on Robotics and Automation, pp.1822-1827, 1991.

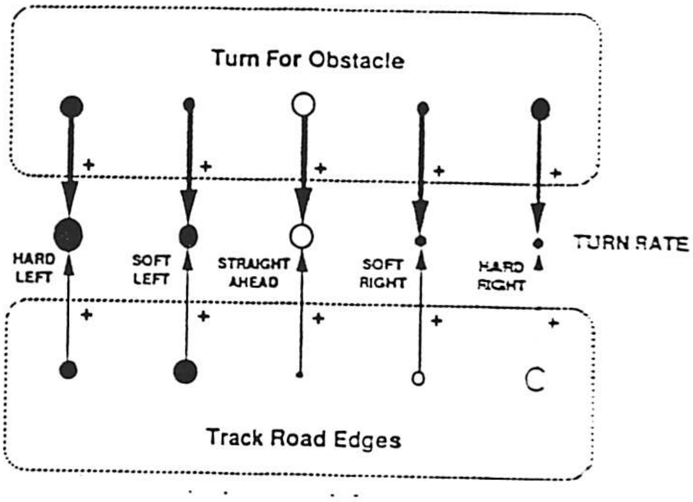


Figure 16: Fine grained primitives

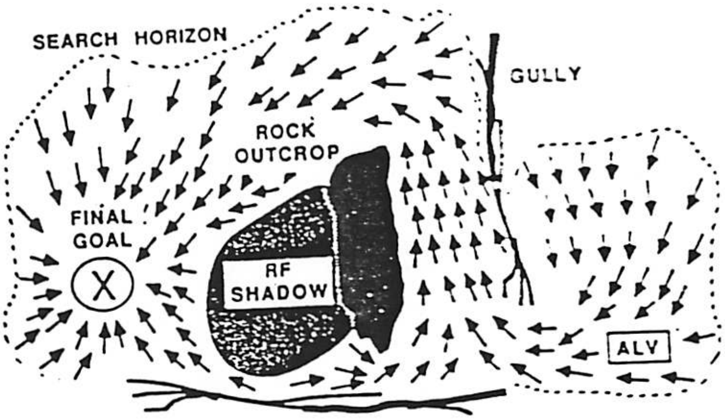


Figure 17: Plans guide behaviors