CHAPTER V

MOTOR SCHEMA BASED MOBILE ROBOT NAVIGATION

Path planning and navigation, at the execution level, can most easily be described as a collection of behaviors. *Don't run into things! Go to the end of the sidewalk then turn right! Stay to the right side of the sidewalk except when passing! Watch out for the library - the turn is just beyond it! Follow that man!* This collection of commands constitutes some of the possible behaviors for an entity trying to move from one location to another. Traditional control structures – those that use an inflexible and rigid approach to navigation – do not provide the essential adaptability necessary for coping with unexpected events. These events might include unanticipated obstacles, moving objects, or the recognition of a landmark in a seemingly inappropriate location. These unexpected occurrences should influence, in an appropriate manner, the course which a vehicle (or person) takes in moving from start to goal.

A potential solution can be drawn from models that have been developed in the domains of brain theory and robotics. Schemas, a methodology used to describe the interaction between perception and action, can be adapted to yield a mobile robot system that is highly sensitive to the currently perceived world. Motor schemas operating in a concurrent and independent, yet communicating, manner can produce paths that reflect the uncertainties in the detection of objects. Additionally they can cope with conflicting data arising from diverse sensor modalities and strategies.

The purpose of this chapter is to provide insights into the design of a control system based on motor schemas for mobile robots. Section 1 describes the motivations for the use of schema theory in this domain – drawing from work in both brain theory and robotics. Section 2 discusses the tack taken for a motor-schema-based control system in the Autonomous Robot Architecture (AuRA), utilizing a mobile robot equipped with ultrasonic and video sensors; specifically the role of the pilot and the motor schema

manager. Section 3 presents the results of simulations using schemas that specify different behaviors and draw on simulated sensor input. Section 4 describes the implementation of the motor subsystem in AuRA. A summary and evaluation concludes this chapter.

§1. Motivation

The concept of schemas originated in psychology [19,99,109] and neurology [57,48]. Webster [138] defines a schema as "a mental codification of experience that includes a particular organized way of perceiving cognitively and responding to a complex situation or set of stimuli". The model used for this paper draws on more recent sources: the applications of schema theory to brain modeling and robotics. As brain theory can unequivocally be called a sound basis for the study of intelligent behavior, the first part of this section will present the contributions of brain science that influenced the design of the schema control system described below. Roboticists for some time have drawn on schema theory, not always in the form envisioned by brain theoreticians. The previous work in robotics that relates to the schema-based approach to navigation is described in the final part of this section.

§1.1 *Brain Theory and Psychology*

The action-perception cycle (Fig. 43) provides a principal motivation for the application of schema theory [95]. Sensor-driven expectations provide the plans (schemas) for appropriate motor action, which when undertaken provide new sensory data that is fed back into the system to provide new expectations. This cycle of cognition (the altering of the internal world model), direction (selection of appropriate motor behaviors), and action (the production of environmental changes and resultant availability of new sensory data) is central to the way in which schemas must interact with the world.

Most significantly, perception should be viewed as action-oriented. There is no need to process all available sensor data, only that data which is pertinent to the task at hand. The question for the roboticist would be: how do we select from the wealth of sensor data available that which is relevant? By specifying schemas, each individual component of the overall task can make its demands known to the sensory subsystem, and thus guide the focus of attention mechanisms and limited sensory processing that is available.

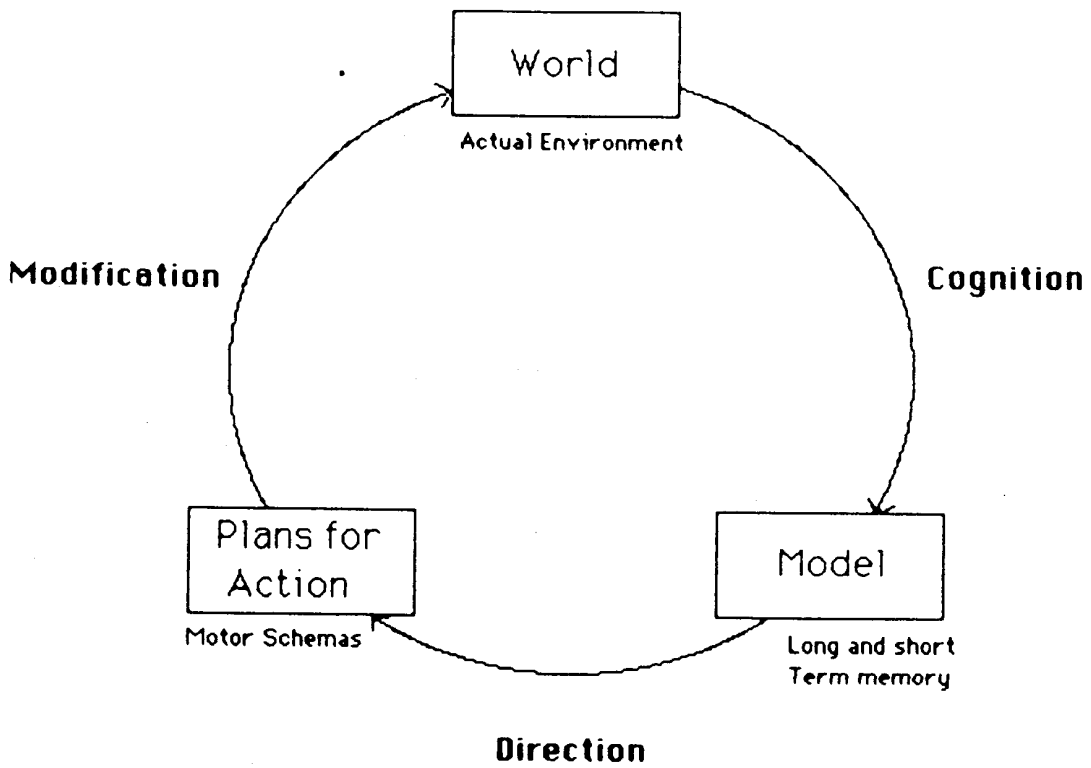


Figure 43: Action-perception cycle

Guided by Arbib's work [6,5] in the study of the frog and its machine analog *Rana Computatrix*, the frog prey selection mechanism serves as a basis for analysis. In particular, Arbib and House [8] have developed a model for worm acquisition by the frog in an obstacle-cluttered environment (a spaced fence - Fig. 44). Although Arbib and House describe two models to account for the behavior of the frog, the second is the most readily applicable to the mobile robot's domain (the first model is based on visual orientation). In their work, they describe primitive vector fields (Fig. 45): a prey-attractant field, a barrier-repellent field, and a field for the animal itself. These fields, when combined, yield a model of behavior (Fig. 46) that is consistent with experimental observations of the frog.

In the mobile robot system described below, analogs of these fields are used (prey-attractant ⇒ **move-to-goal**, barrier-repellent ⇒ **avoid-static-obstacle**). Additionally, new fields are added to describe additional motor tasks (**stay-on-path**, **avoid-moving-obstacle**, etc.)

This model, in conjunction with expectation-driven sensing, provides a basic correlate with the functioning of the brain (albeit the frog brain). Although the brain has been handling visually guided detours since time immemorial, the benefits of using a neuroscience model would wane if it proved impractical for a mobile robot. In the sections following, the practicality of this approach is demonstrated, especially regarding the decomposition of the task to a form which is readily adaptable to distributed processing. This is essential if the real-time demands of mobile robot environmental interaction are to be met.

§*1.2 Robotics*

Schema theory as applied to robotics has almost as many different definitions as there are developers. In the realm of robotic manipulators, Lyons' schemas [78] and Geschke's servo processes [50], (a schema analog), are used as approaches to task level control. Overton [101] has described the use of motor schemas in the assembly domain. The UMASS VISIONS group, guided by Hanson and Riseman, has applied perceptual schemas to the interpretation of natural scenes; Weymouth's thesis [140] and Draper's paper [41] are prime examples of this work. Although AuRA will, in the future, include perceptual schemas running in the context of the VISIONS system, perceptual schemas

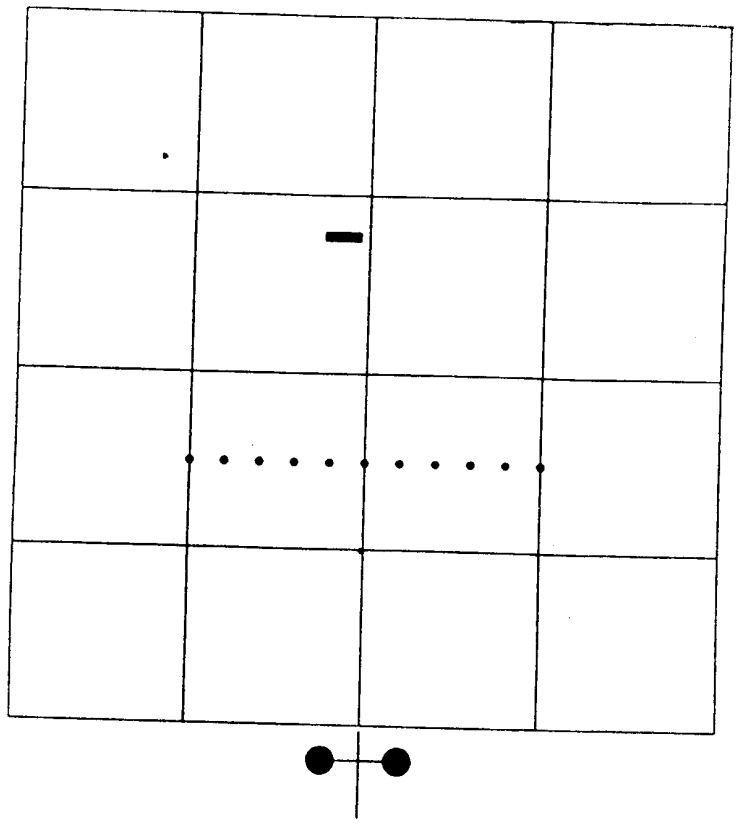


Figure 44: A depiction of a frog prey-selection scenario.
The two large blackened circles at the bottom of the figure denote the frog's eyes, the smaller circles are fence-posts, and the darkened rectangle a supply of worms.
(reprinted from [8] with permission)

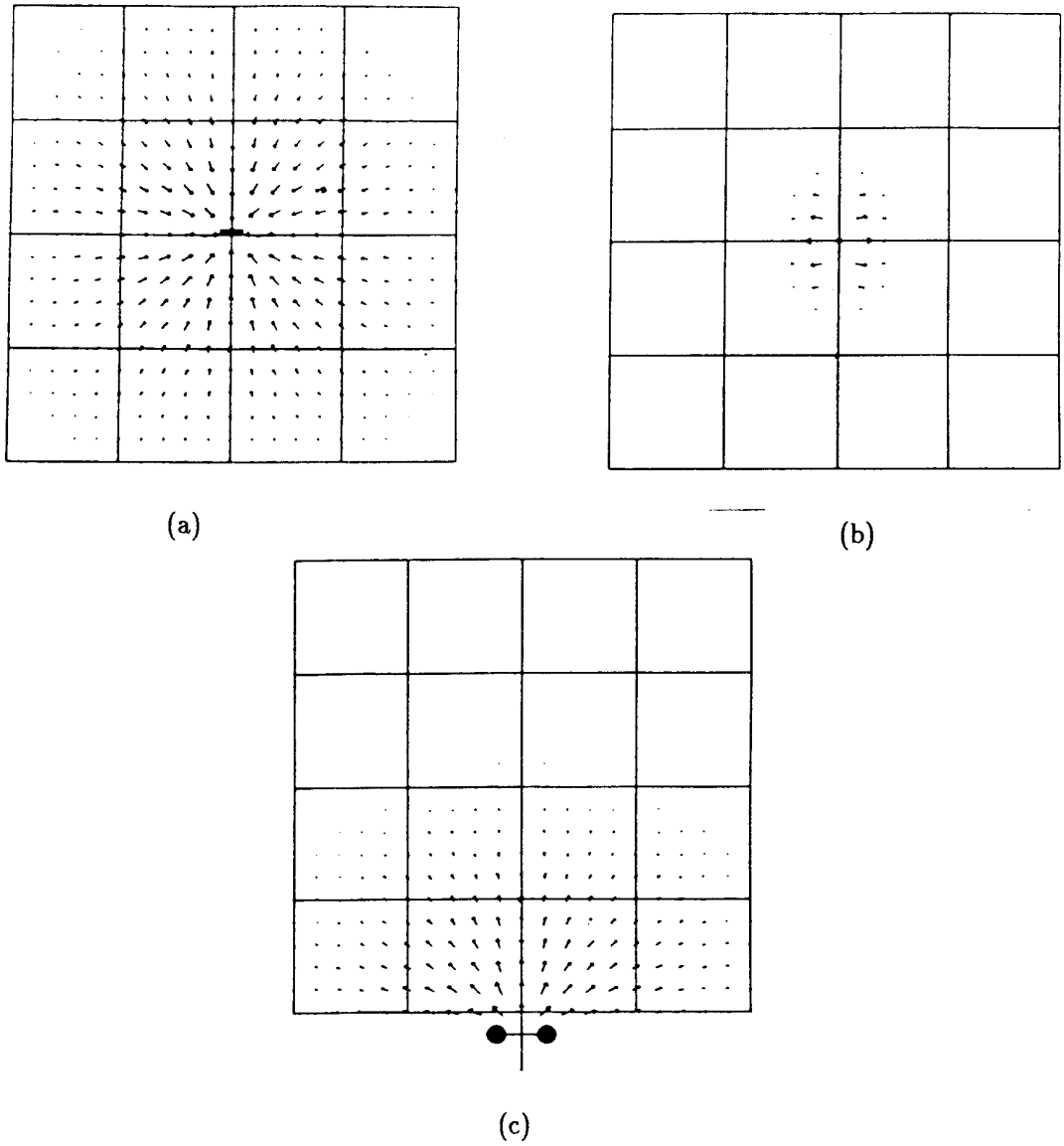


Figure 45: Primitive vector fields associated with Figure 44.

a) Prey-attractant field.

b) Barrier repellent field.

c) Frog representation field.

(reprinted from [8] with permission)

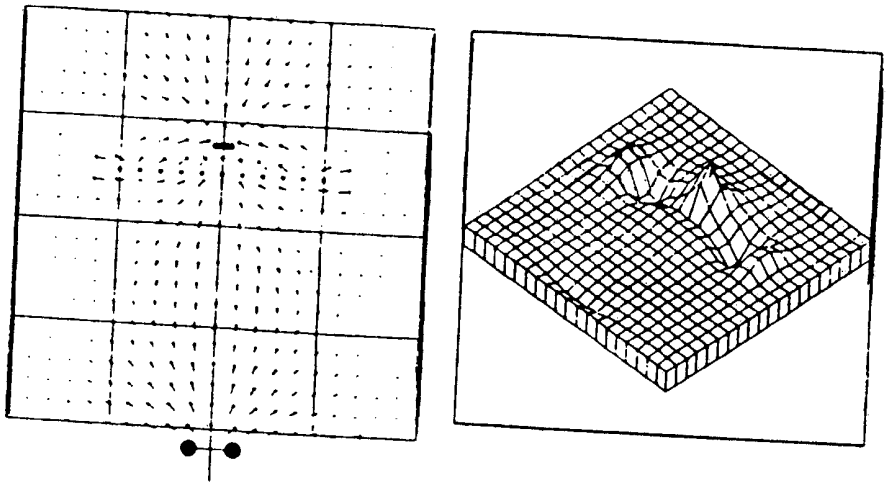


Figure 46: Resultant frog-prey selection vector field.
(reprinted from [8] with permission)

as they appear in the VISIONS system are not a principal concern of this chapter.

One of the simplest and most straightforward definitions for a schema is "a generic specification of a computing agent" [78]. This definition fits well with the concept of a behavior (an individual's response to its environment) – each schema represents a generic behavior. Schema-based control systems are significantly more than a collection of frames or templates for behavior, however. The way in which they are set into action and interact immediately distinguish them from simpler representational forms. The instantiations of these generic schemas provide the potential actions for the control of the robot. A schema instantiation (SI) is created when a copy of a generic schema is parameterized and activated as a computing agent.

Lyons further defines a motor schema as a control system or motor program which describes a task. Overton [101] describes a motor schema as "a control system which continually monitors feedback from the system it controls to determine the appropriate pattern of action for achieving the motor schema's goals (these will, in general, be subgoals within some higher-level coordinated control program)". This more constrained definition is also in accord with the system described below. Sensory perception provides the feedback to affect individual instantiations of motor schemas, each SI thus providing an appropriate behavior which collectively determine the overall system's behavior. Some other definitions for motor schema include an "interaction plan" [7] or "unit of motor behavior" [77].

Other work in the path planning domain, although not schema based, bears a resemblance to the schema control system. Brooks [24] uses a planning system with a "horizontal decomposition" which effectively emulates multiple behaviors. Although related, there is still a rigid layering present which distinguishes it from a schema-based approach. Payton [107] describes a multi-behavior approach for reflexive control of an autonomous vehicle. The association of virtual sensors with a selected set of reflexive behaviors bears a similarity to the schema-based approach. An arbitrary choice of behavior, however, based on a priority system, is made during execution, without provision for a mechanism to combine the results of concurrent behaviors. Kadonoff et al [60] also incorporate multiple behaviors for the control of a mobile robot and similarly arbitrate between these behaviors, proposing a production system for arbitrating competitive strategies and the use of an optimal filter for the treatment of complementary strategies.

The schema system described below is strongly influenced by Krogh's [68] generalized potential fields approach and to a lesser degree by Lyons' [79] tagged potential fields. It bears a superficial resemblance to the integrated path planning and dynamic steering control system described by Krogh and Thorpe [69]. Potential fields are used, in each case, to produce the steering commands for a mobile robot. A major distinction between their system and our schema model lies in the tracking of the individual obstacles (individual SIs for each obstacle, important for the treatment of uncertainty) and the incorporation of additional behaviors such as road following and treatment of moving obstacles. The state of each obstacle's SI is dynamically altered by newly acquired sensory information. The potential functions for each SI reflect the measured uncertainty associated with the perception of each object. The schema approach is not limited to obstacle avoidance, but is versatile enough for road following, object tracking and other behavioral patterns.

§2. Approach

Motor schemas, when instantiated, must drive the robot to interact with its environment. On the highest level, this will be to satisfy a goal developed within the planning system; on the lowest level, to produce specific translations and rotations of the robot vehicle. The schema system enables the software designer to deal with conceptual structures that are easy to comprehend and handle. The task of robot programming is fundamentally simplified through the use of a divide and conquer strategy.

§2.1 *Schema-based Navigation*

AuRA's pilot is charged with implementing leg-by-leg the piecewise linear path developed by the navigator. To do so, the pilot chooses from a repertoire of available sensing strategies and motor behaviors (schemas) and passes them to the motor schema manager for instantiation. Distributed control and low-level planning occur within the confines of the motor schema manager during its attempt to satisfy the navigational requirements. As the robot proceeds, the cartographer, using sensor data, builds up a model of the perceived world in short-term memory. If the actual path deviates too greatly from the path initially specified by the navigator due to the presence of unmodeled obstacles or positional errors, the navigator will be reinvoked and a new global path computed.

If the deviations are within acceptable limits, (as determined by higher levels in the planning hierarchy), the pilot and motor schema manager will, in a coordinated effort, attempt to bypass the obstacle, follow the path, or cope with other problems as they arise. Additionally, the problem of robot localization is constantly addressed through the monitoring of short-term memory and appropriate **find-landmark** schemas. Multiple concurrent behaviors (schemas) may be present during any leg, for example:

- **Stay-on-path** (a sidewalk or a hall)
- **Avoid-static-obstacles** (parked cars, trees, etc.)
- **Avoid-moving-obstacles** (people, moving vehicles, etc.)
- **Find-intersection** (to determine end of path)
- **Find-landmark**(building) (for localization)

The first three are examples of motor schemas, the last two perceptual schemas. To provide the correct behavior, a subset of perceptual schemas must be associated with each motor schema. For example, in order to stay on the sidewalk, a **find-terrain**(sidewalk) perceptual schema must be instantiated to provide the necessary data for the **stay-on-path** motor schema to operate. If the uncertainty in the actual location of the sidewalk can be determined, the SI's associated velocity field, applying pressure to remain on the sidewalk, will reflect this uncertainty measure. The same holds for obstacle avoidance: if a perceptual schema for obstacle detection returns the position of a suspected obstacle and the relative certainty of its existence, the actual avoidance maneuvering will depend not only on whether an obstacle is detected but also on how certain we are that it exists. Differing strategies within each SI will determine how to manage the perceptual uncertainty. If an event is potentially fatal, even large amounts of perceptual uncertainty would produce motor behavior, but erring in the direction of safety.

An example illustrating the relationship between motor schemas and perceptual certainty follows. The robot is moving across a field in a particular direction (**move-ahead** schema). The **find-obstacle** schema is constantly on the look-out for possible obstacles within a subwindow of the video image (windowed by the direction and velocity of the robot). When an event occurs, (e.g. a region segmentation algorithm detects an area that is distinct from the surrounding backdrop or an interest operator locates a high-interest

point in the direction of the robot's motion), the **find-obstacle** schema spawns off an associated perceptual schema (**static-obstacle** SI) for that portion of the image. It is now the **static-obstacle** SI's responsibility to continuously monitor that region. Any other events that occur elsewhere in the image spawn off separate **static-obstacle** SIs. Additionally an **avoid-static-obstacle** SI motor schema is created for each detected potential obstacle.

The motor schema SI hibernates waiting for notification that the perceptual schema is sufficiently confident in the obstacle's existence to warrant motor action. If the perceptual schema proves to be a phantom (e.g. shadow) and not an obstacle at all, both the perceptual and related motor SIs are deinstantiated before producing any motor action. On the other hand, if the perceptual SI's confidence (activation level) exceeds the motor SI's threshold for action, the motor schema starts producing a repulsive field surrounding the obstacle.[1] The sphere of influence (spatial extent of repulsive forces) and the intensity of repulsion of the obstacle are affected by the distance from the robot and the obstacle's perceptual certainty. Eventually, when the robot moves beyond the perceptual range of the obstacle, both the motor and perceptual SIs are deinstantiated. In summary, when obstacles are detected with sufficient certainty, the motor schema associated with a particular obstacle (its SI) starts to produce a force tending to move the robot away from the object. Fig. 47a shows a typical repulsive field for an **avoid-static-obstacle** SI. The control of the priorities of the behaviors, (e.g. when is it more important to follow the sidewalk than to avoid uncertain but possible obstacles) is partially dependent on the uncertainty associated with the obstacle's representation. Other isolated motor schema velocity fields are shown in Fig. 47b-d. Various combinations of motor schemas are illustrated in Fig. 48. Recognize the fact that although the entire field is expensive to compute, each active motor SI need only determine the velocity vector at the robot's current location relative to the environmental object, making the computation very rapid. Further, as the SIs are activated in parallel, even better performance is attainable.

Multiple instantiations of a single schema are frequently the case. Each generic "skeleton" is parameterized when instantiated. Consequently, it is entirely possible that two different instantiations of the same generic schema produce significantly different fields

[1]The obstacle is first grown in a configuration space manner [76] to enable the robot to be treated henceforth as a point for path planning purposes.

Figure 47: Isolated motor schema SI vector fields

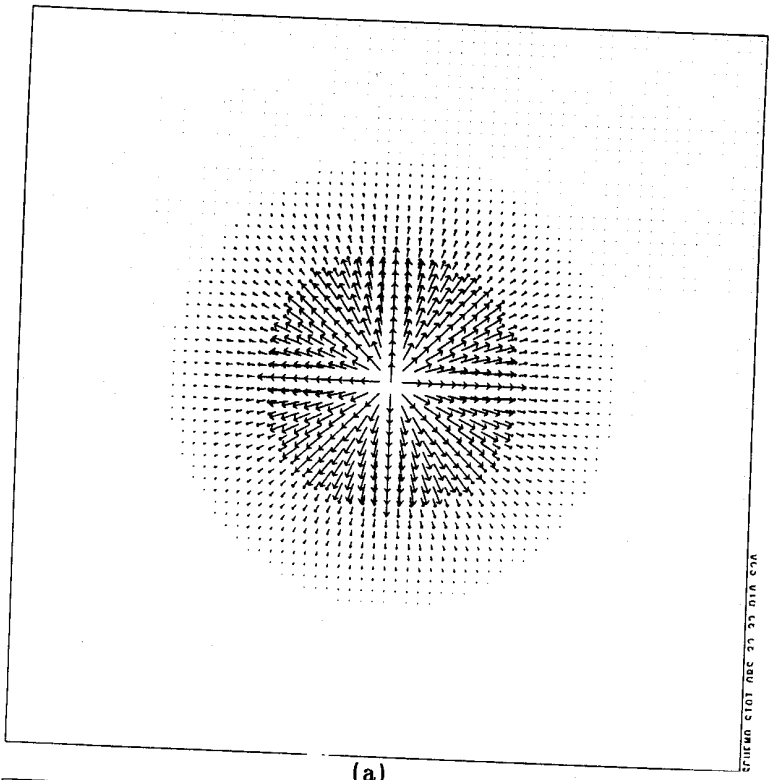
a) Avoid-static-obstacle.

b) Move-to-goal.

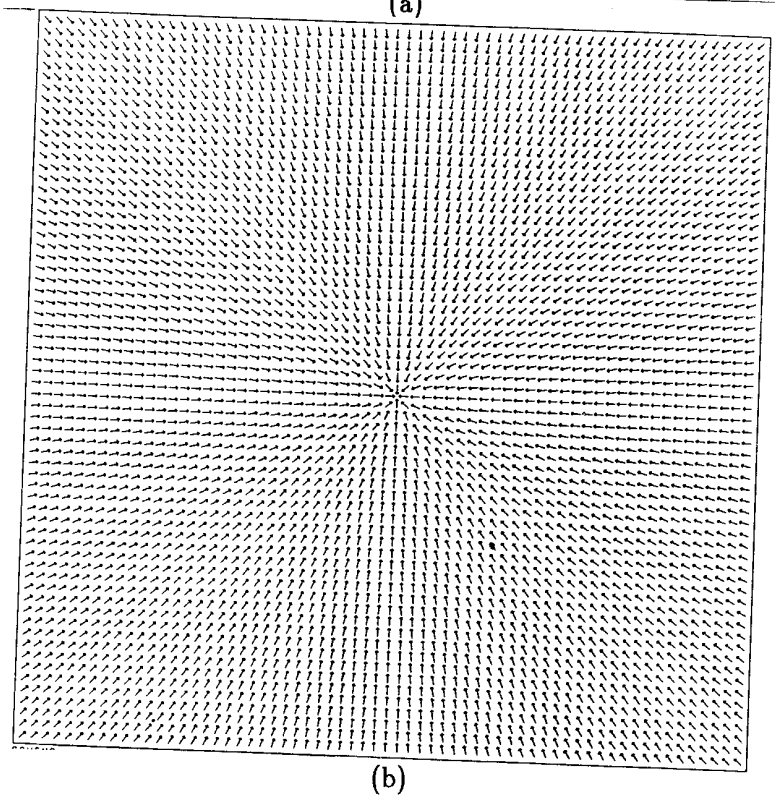
c) Move-ahead.

d) Stay-on-path.

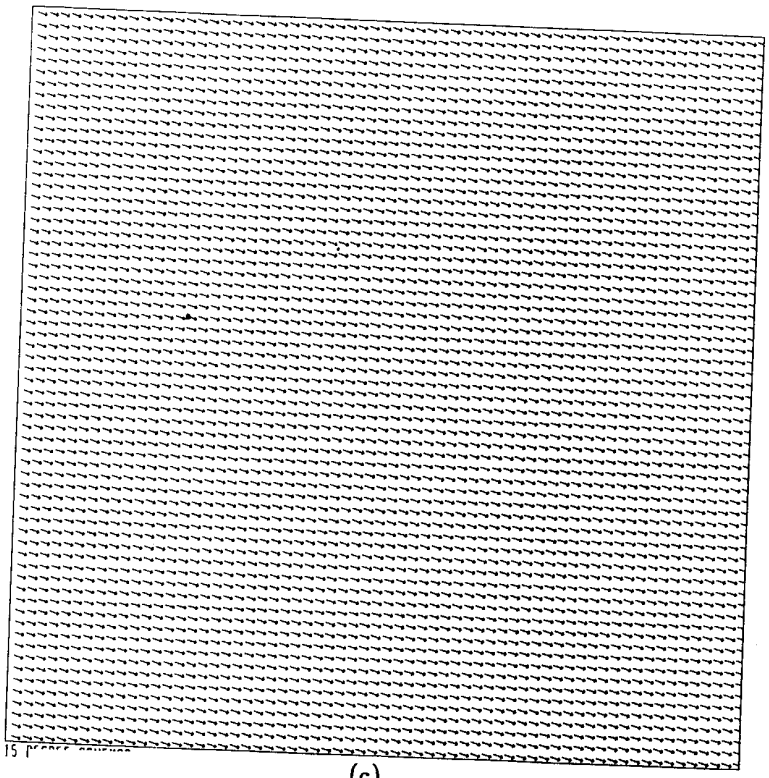
(Figure continued on following page).



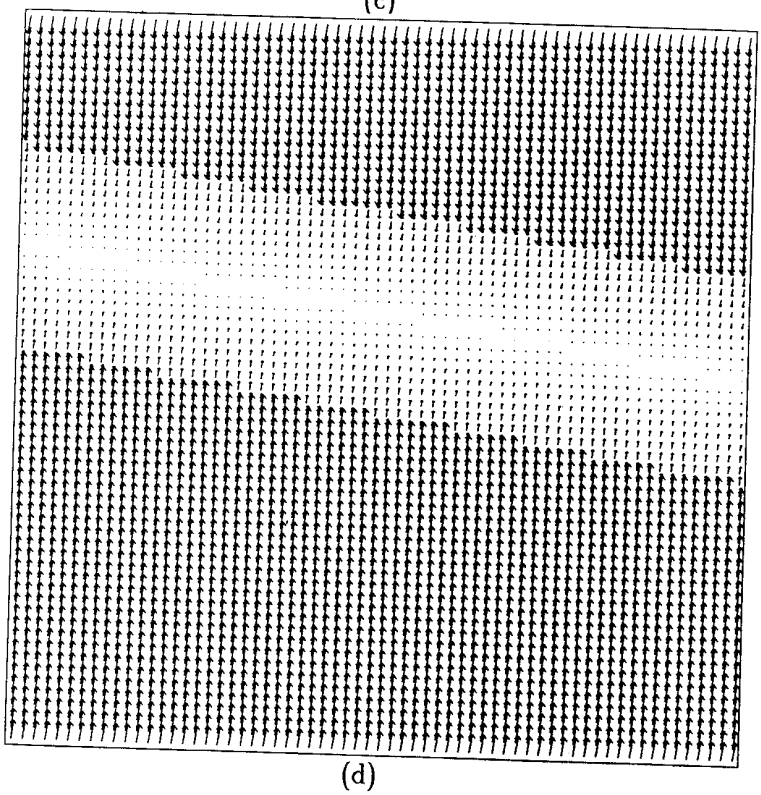
(a)



(b)



(c)



(d)

Figure 47 continued.

Figure 48: Several combined motor schemas

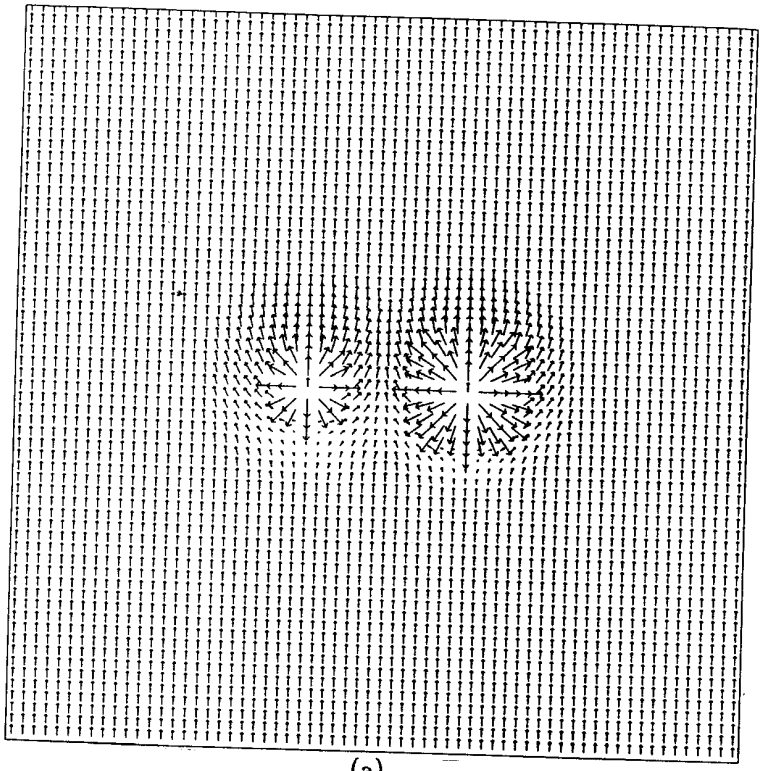
a) Move-ahead SI + two avoid-static-obstacle SIs.

b) Move-ahead SI + stay-on-path SI.

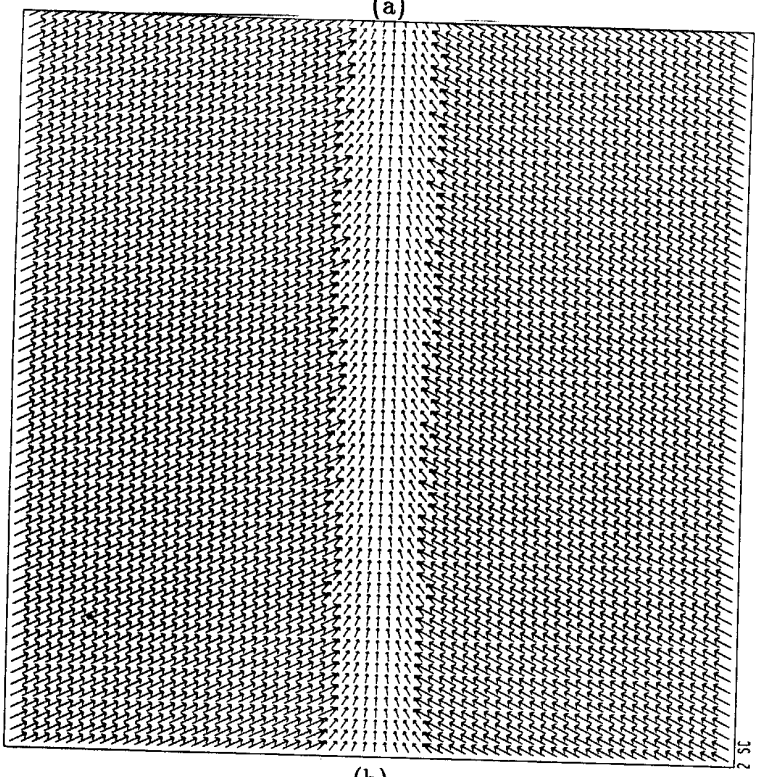
c) Move-ahead SI + stay-on-path SI + one avoid-static-obstacle SI.

d) Move-to-goal SI + stay-on-path SI + two avoid-static-obstacle SIs.

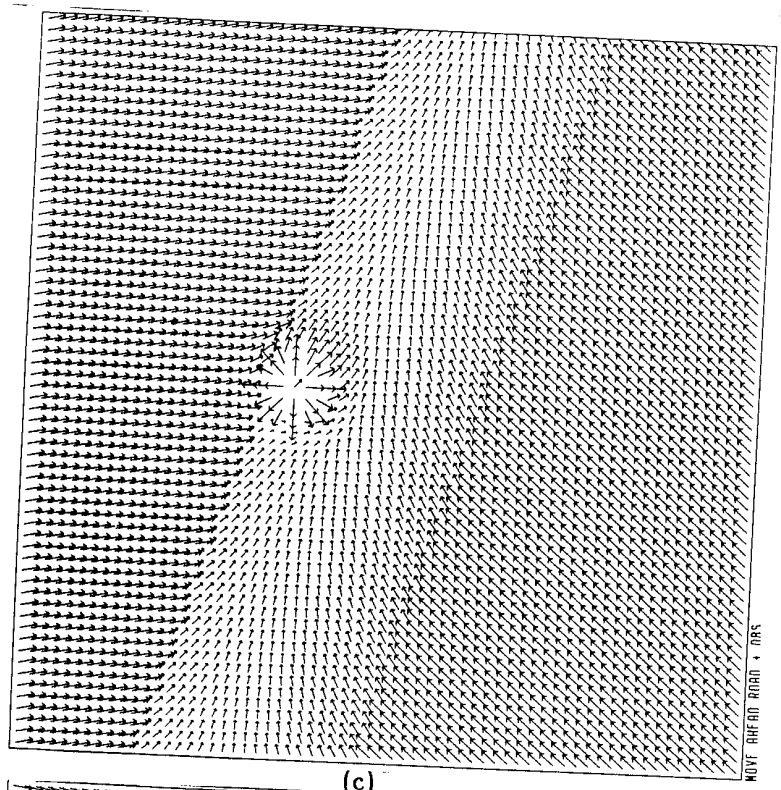
(Figure continued on following page).



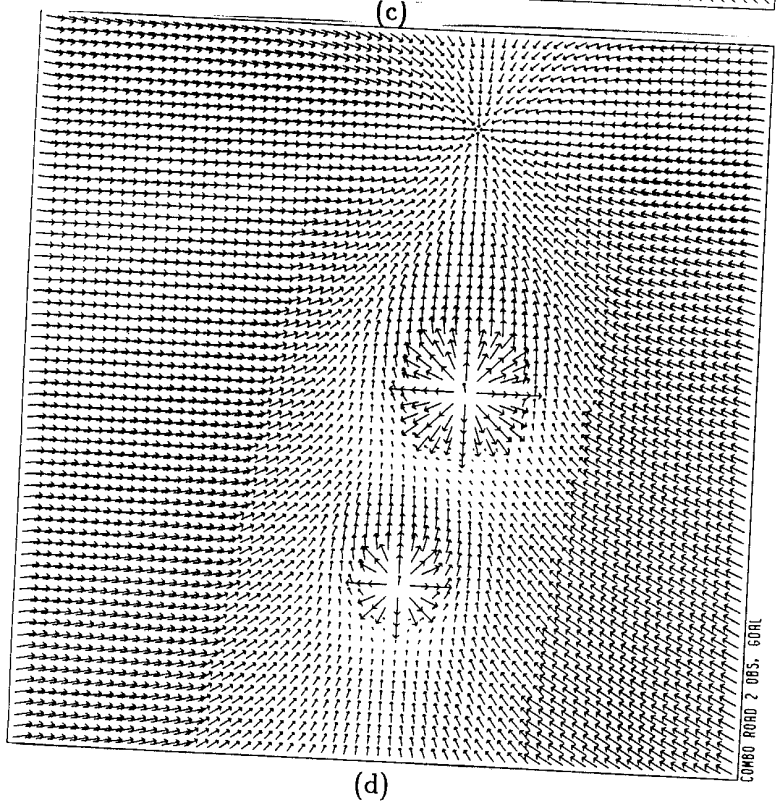
(a)



(b)



(c)



(d)

Figure 48 continued.

under similar sensory conditions (as in the case of path following for a sidewalk or hall discussed above). The parameters set at instantiation may depend on the sensory events that triggered the instantiation or from information retrieved by the pilot from LTM.

If each schema functions independently of each other, how can any semblance of realistic and consistent behavior be achieved? Two components are required to satisfactorily answer this question. First, a combination mechanism must be applied to all the SI-produced vectors. The result is then used to provide the necessary velocity changes to the robot. The simplest approach is vector addition. By having each motor SI create a normalized velocity vector, a single **move-robot** schema monitors the posted data for each SI, adds them together, makes certain it is within acceptable bounds and then transmits it to the low-level robot control system. In essence, the specific velocity and direction for the robot can be determined at any point in time by summing the output vectors of all the active individual SIs. As each motor SI is a distributed computing agent, preferably operating on separate processors on a parallel machine, and needs only to compute the velocity at the point the robot is currently located and a few points in its projected track (and not the entire velocity field), real-time operation is within reach.

The second component of the response to the question posed in the previous paragraph is communication. Potential fields can have problems with dead spots or plateaus where the robot can become stranded. By allowing communication mechanisms between the SIs, the forces of conflicting actions can be reconciled. Lyons [78] proposes message passing between ports on one SI and connected ports on another SI as a schema communication mechanism. Alternatively, a blackboard mechanism is used in the VISIONS Schema Shell (discussed below). In either case, communication mechanisms can solve problems that might otherwise prove intractable. An example to illustrate this point follows.

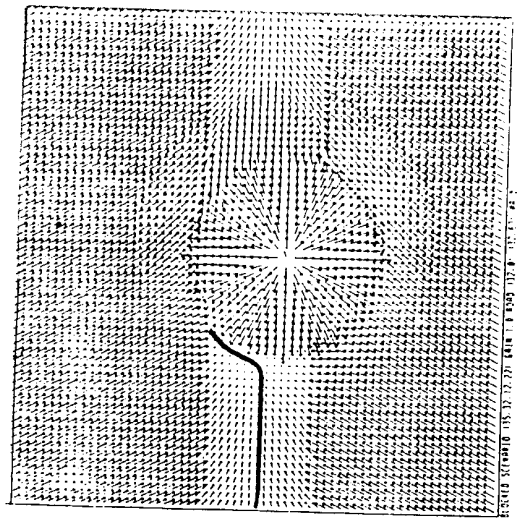
The robot is instructed to move in a particular direction, stay on the sidewalk and avoid static obstacles. Suppose that the sidewalk is completely blocked by an obstacle; eventually the velocity would drop to 0 and the robot would stop (Fig. 49a). The fact that the vehicle has stopped is detected by the **stay-on-path** SI through interschema communication with the **move-robot** SI (the **move-robot** SI combines the individual motor SIs and communicates the results to the low-level motor control system). The **stay-on-path** SI, when created for this particular instance, was instructed to yield if an obstacle blocks the path. The **stay-on-path** motor schema reduces its field (Fig. 49b)

and allows the robot to wander off the sidewalk thus circumnavigating the obstacle. As soon as the direction of the force produced by the offending obstacle indicates it has been successfully passed, the **stay-on-path** field returns to its original state forcing the robot back on the path (Fig. 49c).

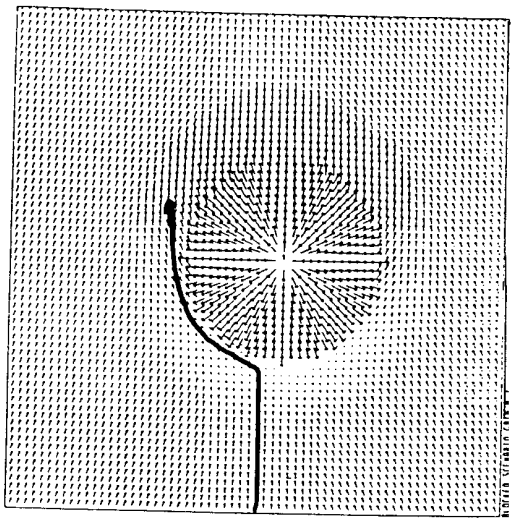
Suppose, however, the **stay-on-path** SI was instantiated for a hall. Then, under no circumstances, would the force field associated with the **stay-on-path** SI be reduced or else the robot would crash into the wall. The robot would instead stop, and signal for the navigator (higher level component of the planner) to be reinvoked and produce an alternate global path that avoids the newly discovered blocked passageway. These communication pathways are specified within the schema structures themselves.

It is entirely possible that the trajectory of the robot can be computed for a small distance rather than just its instantaneous velocity at the immediate location. Each motor schema would have to interact with the **move-robot** SI, using the vector summation output to determine the position of the robot relative to its perceptions for the next time step. This is of particular significance if the sensor sampling rates are low. Trajectories can be determined that reflect the robot's perceptions at a given point in time, rather than just reacting to current sensing. This is of value in determining when to activate other schemas in anticipation of special problems or needs. Care must be taken in highly dynamic environments (e.g. moving objects) so that the plans developed do not ignore changes in the world that are evidenced only through perception.

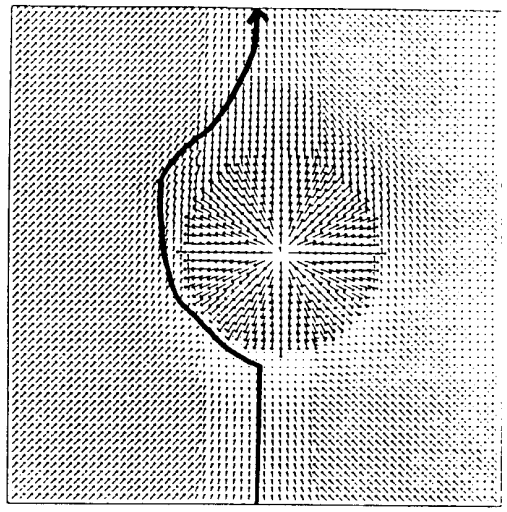
Another approach explored is the addition of a background stochastic **noise** schema. This SI produces a low-magnitude random direction velocity vector that changes at random time intervals, but persists sufficiently long to produce a change in the robot's position if the robot's velocity was otherwise zero. Its role is to perturb the velocity of the robot slightly, removing the robot from undesirable equilibrium points, which arise when the active motor SIs counterbalance each other. The behavior produced by this schema corresponds to the "wander" layer in Brooks' horizontally layered architecture [24]. This schema would serve to remove the robot from any potential field plateaus or ridges upon which the robot becomes perched (e.g. from a direct approach to an obstacle - Fig. 50). Other traps common to potential field approaches (e.g. box canyons) can be handled by establishing hard time deadlines for goal attainment. If these deadlines are violated, the pilot would be reinvoked to establish an alternate route using STM data



(a)



(b)



(c)

Figure 49: Blocked sidewalk scenario

a) Robot stops in dead spot due to pressure to both remain on sidewalk and avoid the obstacle.

b) Gain lowered on **stay-on-path** SI allows robot to bypass obstacle.

c) Once obstacle is passed **stay-on-path** SI returns to normal, forcing robot back onto the sidewalk.

gathered by the cartographer during the route traversal.

It is worth noting that a single sensory event may have two or more SIs associated with it. For example: if the robot is looking for a mailbox to get its bearings for localization purposes, a perceptual schema for localization (**find-landmark**) would process portions of the image that are likely to be mailboxes. If the mailbox happens to be in the path of the vehicle, a concurrent **avoid-static-obstacle** SI would view that object not as a mailbox but rather as an obstacle, and would be concerned only with avoiding a collision with it. This "divide and conquer" approach based on action-oriented perception simplifies programming and overall system design. A more complex scenario appropriate for a mobile robot appears in Fig. 51.

§3. Implementation Strategy

The implementation tool chosen for the motor schema system is the Schema Shell [40,41,42], a system developed by the VISIONS group at UMASS for use in the perceptual schema analysis of natural scenes. It currently runs on a Texas Instruments Explorer workstation and is tied to the Computer Science Department's VAXen over Chaosnet. The schema communication mechanism is blackboard-based. The Schema Shell system is expected to be moved to the department's newly acquired Sequent parallel processor. Although the Explorer only simulates distributed processing, everything points towards the availability of a truly distributed environment in the not too distant future.

The schemas themselves (in the Schema Shell) consist of a schema template and multiple strategies. Associated with each instantiated schema is an object hypothesis maintenance (OHM) strategy. This part of the SI monitors the blackboard for new events (e.g. sensory data) that would produce changes in the SI's posted output. Other strategy components for each SI handle conflict resolution, cooperative enhancement, initialization and other relevant factors. Not all strategies are necessary or desirable for all schemas. Figure 52 shows a typical generic motor schema cast in the Schema Shell format.

Pocock at the UMASS Laboratory for Perceptual Robotics is developing an alternative schema-based robot control system [107] based on Lyons' port automata formalization [78] of schema theory. As it seriously addresses real-time distributed scheduling, it may prove

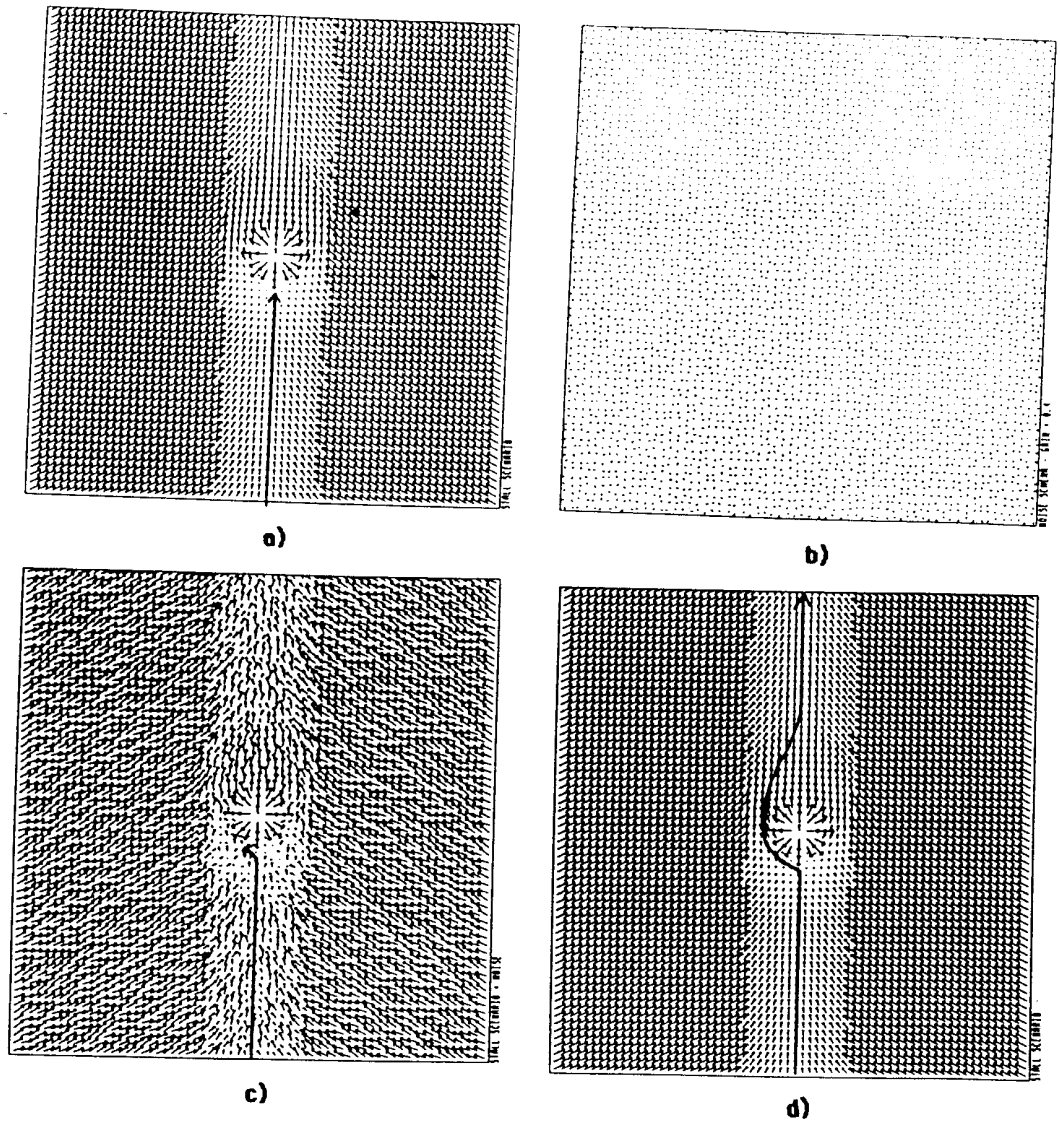


Figure 50: Stall scenario

a) If the robot approaches an obstacle exactly head-on, it is possible for it to become stalled.

b) **Noise** SI provides small magnitude random direction vector to push robot off of the tiny plateau.

c) **Noise** schema added to a).

d) The robot can now successfully bypass the obstacle. The **noise** SI is then deinstantiated.

Pilot issues instructions to follow sidewalk while avoiding obstacles.
Continue approximately 200 ft on sidewalk then turn right at lamppost onto intersection (first encountered). Watch for landmarks (mailbox on left, building edge on right) for localization.

Motor Schemas instantiated by pilot:

- Stay_on(identify_terrain(sidewalk,60%)) (assumes sidewalk is ahead to start).
- Move_ahead(210,start_heading) (nominal distance plus some slop).
- Avoid_static_obstacles(15,identify_obstacle(robot_heading,nil,70%))
  Start maneuvering around when within 15 feet. Nil denotes static obstacle. 70% is threshold for motor action.
- Avoid_dynamic_obstacles(20,identify_obstacle(robot_heading,-robot_heading,40%))
  Start evasive action when head on approach within 20 feet.
- Follow_dynamic_obstacle(8,start_heading,identify_obstacle(True,start_heading,95%))
  When an obstacle is moving in the correct direction, within 8 feet of the robot , follow it (regardless of robot's current heading).
- Avoid_dynamic_obstacles(5,identify_obstacle(robot_heading,True,40%))
  Start evasive action when within 5 feet for any dynamic obstacle (includes crossing dynamic obstacles).
- Turn_when(find_landmark(lamppost 1+5ft,90,90%)) – right 90 degrees.
- Turn_when(find_landmark(intersection_3a,90,90%)) – right 90 degrees.
- Localize(find_landmark(mailbox_7,90%)).
- Localize(find_landmark(building_2a.edge3, 85%)) – prune spatial uncertainty map on landmark recognition.
- Stop_when(not (sidewalk_1 = identify_terrain(ahead,90%))) – missed turn.

  (Percentages denote thresholds for motor action).

Perceptual Schemas instantiated by pilot:

- Identify_obstacle(robot_heading,obstacle_heading,certainty) Only detects obstacles in the way of the robot (distinct from landmarks). Robot_heading and obstacle_heading are directional filters. Certainty is threshold for identification. Returns obstacle position and type. 1 identify_obstacle spawned for each strategy type above.
  obstacle - generic - many spawned for each identify_obstacle.
  Returns certainty. Tracks motion over time.
  types: static_obstacle and dynamic_obstacle (predicates)
  obstacle_heading (nil if static), speed (0 if static).
- Find_landmark(LTM_model,certainty) - 1 spawned per landmark.
  Assumes robot's current position for observation is available in global coordinates (spatial uncertainty map). Certainty is threshold for recognition. Returns landmark location. Landmark is not necessarily in direct track of robot, could be anywhere.
  landmark(LTM_model) - many/landmark spawned off
  Returns certainty.
- Identify_terrain(position,certainty) – Returns terrain type.

At end of maneuver, deinstantiate all obstacle schemas.

Figure 51: Example mobile robot schema scenario

```
;-----------------------------------------------------------------------
;                        MOVE-AHEAD Motor Schema
;       ->
;       ->        x-axis is 0; frame of reference is robot's initial heading afterturn;
;       ->
;       ->                          .

 (make-motor-schema
        :name "MOVE-AHEAD"
        :default-argument-list (list '("heading" 0))
        :body
        (progn
                (de-schema  move-ahead (original-heading current-heading move-ahead-force-table))

                (de-strategy move-ahead ohm ()
                  (progn
                    (call-strategy 'move-ahead 'init)
                    (write-to-window "move-ahead ohm")
         (loop
                    (write-to-window (setf #!current-heading
                  (read-or-wait '(current-orientation-messagep robot-position))))

                      (write-to-window (write-to-blackboard
                        (list "move-ahead" (look-up-move-ahead-force #!current-heading) (time-stamp))
                              'vector-section)) ; write resultant force to vector section
                      )
         ))

;               strategies for move-ahead

                (de-strategy move-ahead init()
                  (build-move-ahead-force-table)                          ; initialize lookup table
                  (setf #!original-heading (read-or-wait '(orientation-messagep robot-position)))
                  (write-to-window "move-ahead init done")
                )


       ; conflict in case of reverse direction- send message-to-mover-to-terminate
       ; contains list of contradictory motions or evidence

       ; support in case of confirmed direction
       ; support contains list of related schemas use to confirm hypothesis
       ; e.g. to stop-when's
            )  ); End make-motor-schema
```

Figure 52: Example move-ahead schema as implemented in the Schema Shell

a useful tool for mobile robot research when completed. At that time, its relationship to the VISIONS Schema Shell will be considered.

§4. Simulation

Simulations were run on a VAX 750 using the following motor schemas: **stay-on-path**, **move-ahead**, **move-to-goal**, **avoid-static-obstacle**. Each simulation run (Figs. 53–54) shows the sequence of resultant overall force fields based on perceived entities. These entities include path borders, goals, and obstacles. The grid size is 64 units by 64 units and the sensory sampling update time (once per second) is based on a nominal velocity of 1 unit/second. The maximum vector length for display purposes has been set to 2.0 normal velocity units. The actual vector magnitude within the obstacles is set to infinity (a discrete approximation). All obstacles are currently modeled as circles (as in Moravec's tangent space [93]). The field equations for several of the motor schemas appear below.

The field equations for both the **avoid-static-obstacle** and **stay-on-path** schemas are linear. An example showing the velocity produced by an obstacle (O) is given below:

Avoid-obstacle

$O_{magnitude} =$

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

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

$O_{direction}$ = along a line from robot to center of obstacle moving away from obstacle

More complex equations could be used (e.g. cubic as in [69]) but were deemed unnecessary in these early stages of the research.

Stay-on-path

$V_{magnitude} =$

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

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

$V_{direction}$ = in specified compass direction

Move-to-goal

$V_{magnitude}$ = fixed gain value

$V_{direction}$ = in direction towards perceived goal

In some of these simulations the uncertainty in perception was allowed to decrease the sphere of influence of an obstacle. When a threshold was exceeded (50% certain), the sphere of influence of the obstacle started increasing linearly as the certainty increased up to its maximum allowable value. Another alternative is to increase the gain on the

obstacle proportionately with the increase in certainty (up to its maximum).

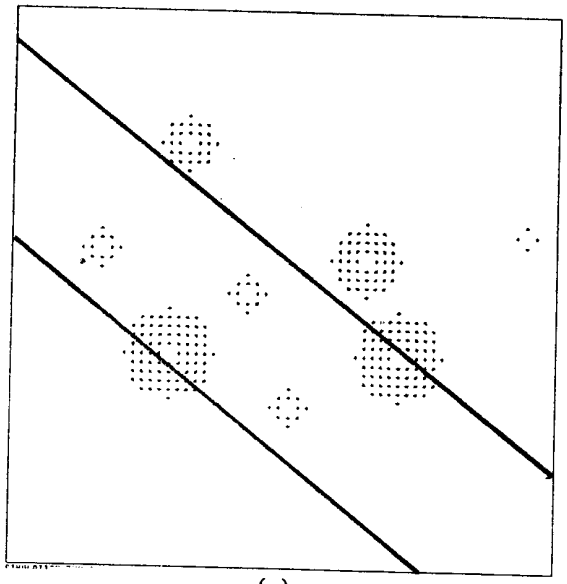
Figure 53a illustrates the robot's course on a sidewalk moving towards a goal. The course is studded with 8 obstacles, only 7 of which are perceptible to the robot during its journey (Fig. 53b). Note how the vector fields change as the robot encounters more obstacles along the way (Figs. 53c-e). When it has successfully navigated obstacles and they have moved out of range, their representation is dropped from short-term memory and the associated motor schema is deinstantiated (Fig. 53e). The robot stays on the path for the complete course successfully achieving its goal while avoiding each obstacle. An expanded version could update long-term memory as a result of experience, thus incorporating learning.

Figure 54 shows the robot's path to a specified goal through a field of 9 obstacles. This simulation prevents perceived objects that have too great an uncertainty from producing a repulsive field. In this case, the uncertainty increases with the distance from the obstacle. A decrease in uncertainty results in an increase in the sphere of influence of the obstacle. Consequently, the uncertainties and the resultant obstacle fields change as the robot moves through the course. Figures 54b-f use a **move-to-goal** SI while Figs. 54g-h use a **move-ahead** SI. Actually the robot would operate under the control of a **move-ahead** SI until the goal is perceived (assuming dead–reckoning or inertial guidance is not used). At the moment of goal perception, the **move-ahead** SI would be deinstantiated and a **move-to-goal** SI created in its stead.

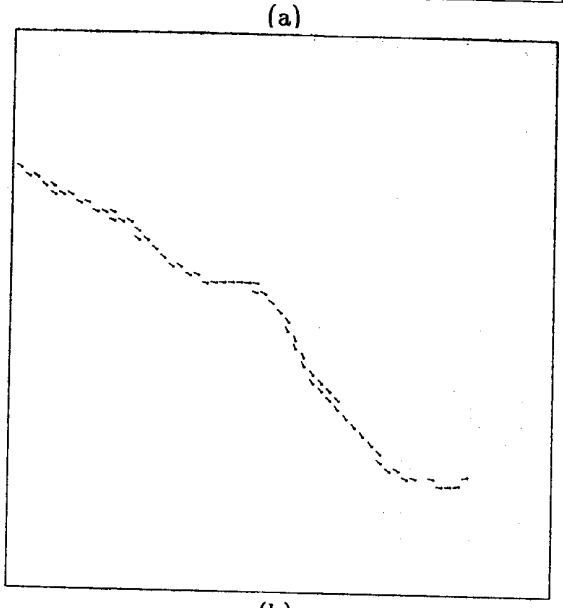
§5. Motor subsystem

AuRA's motor subsystem accepts the output from the motor schema manager's **move-robot** SI and produces the required velocity for the vehicle. Little has been said thus far about the vehicle interface and other components of the motor subsystem other than stating that this component of AuRA is largely vehicle dependent. In the case of the UMASS Denning Research Vehicle (DRV), the motor controllers and motors themselves have been provided by the manufacturer (Denning Mobile Robotics Inc. of Woburn, Mass.). The interested reader is referred to the Denning documentation set [38] for the details of the control circuitry.

Communication with the vehicle is another story. The robot runs a terminal emulation



(a)



(b)

Figure 53: Schema simulation run

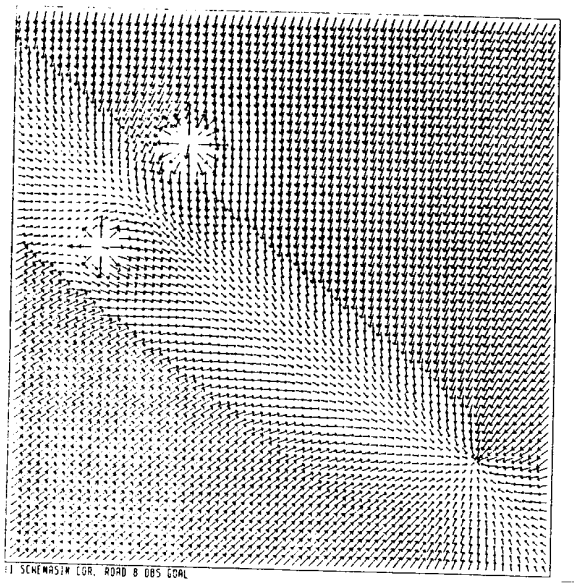
This simulation shows 7 **avoid-static-obstacle** SIs, a **move-to-goal** SI, and a **stay-on-path** SI.

a) Shows the layout of the obstacle ridden course.

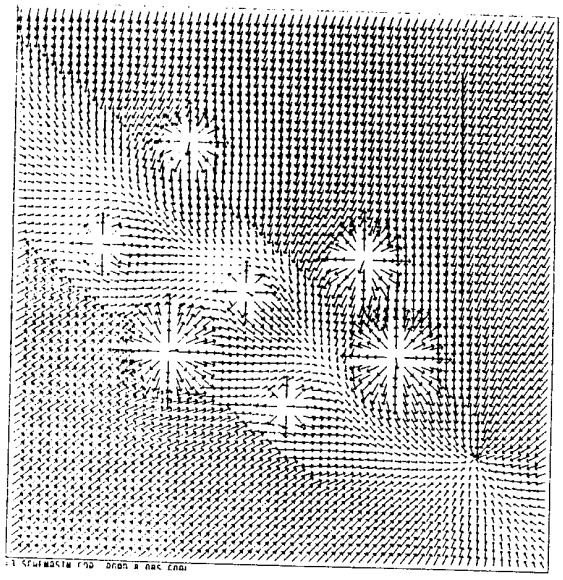
b) Simulated robot path through course.

c-e) With the robot starting at the upper left, the robot's progress through the course can be observed. Note that the obstacles are added as they are perceived by the sensory system. No *a priori* knowledge of their location is assumed.

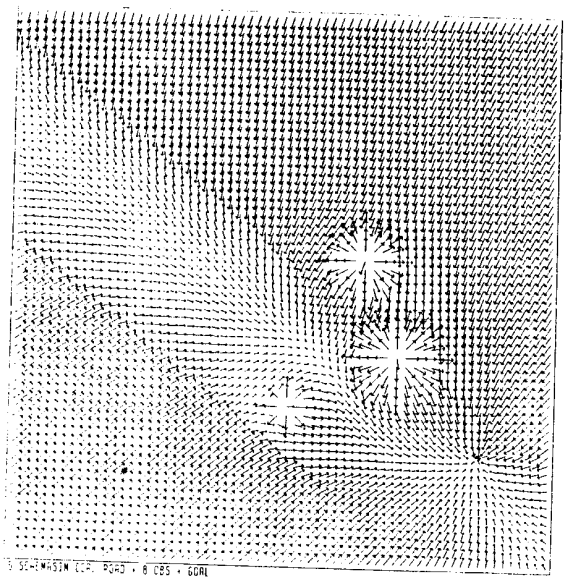
(Figure continued on following page).



(c)



(d)



(e)

Figure 53 continued.

Figure 54: Another simulation run

This simulation includes 9 avoid-static-obstacle SIs and 1 move-to-goal SI.

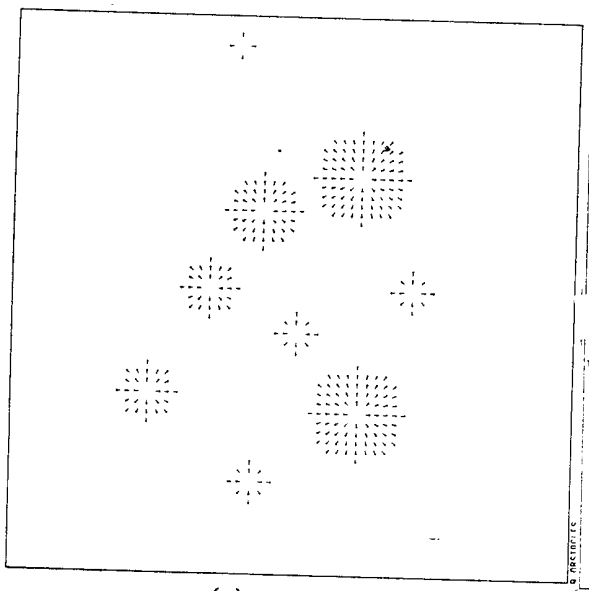
a) Location of 9 obstacles.

b) Path of robot as it crosses from left to right around obstacles to the goal.

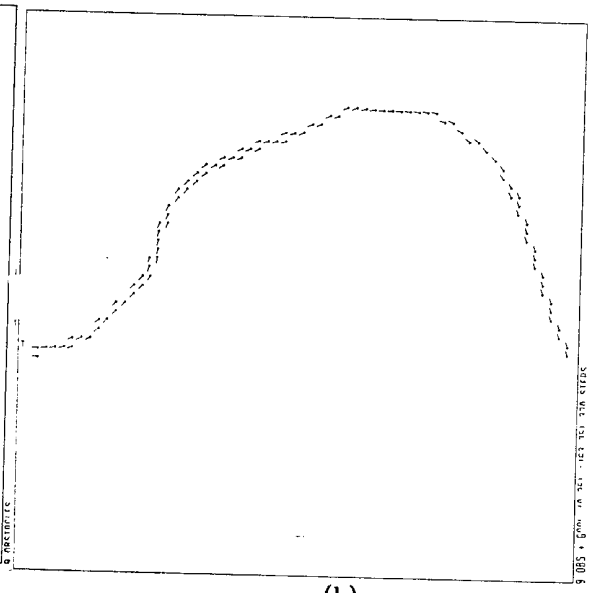
c & d) Velocity fields based on robot's perceptions as it moves from left to right as shown in b).

This simulation includes an uncertainty measure for obstacles which increases with the distance of the obstacle from the robot. If the obstacle is relatively uncertain, its position is shown but it produces no field (e.g. the two rightmost obstacles in Fig. c). As the robot approaches, it becomes more certain of the obstacles and starts to produce a repulsive field surrounding the obstacle.

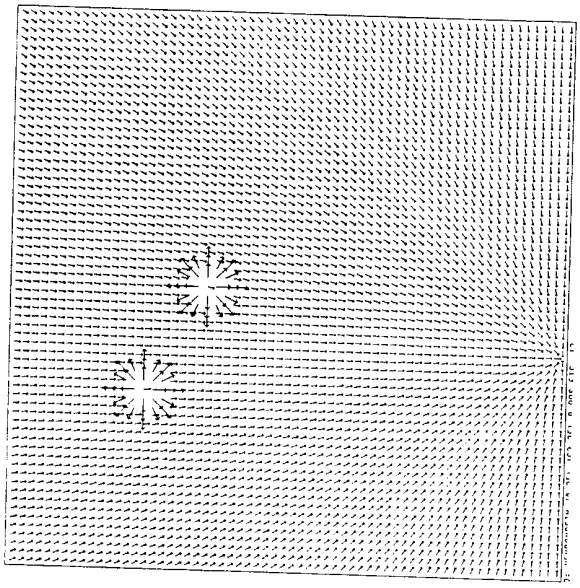
(Figure continued on following page).



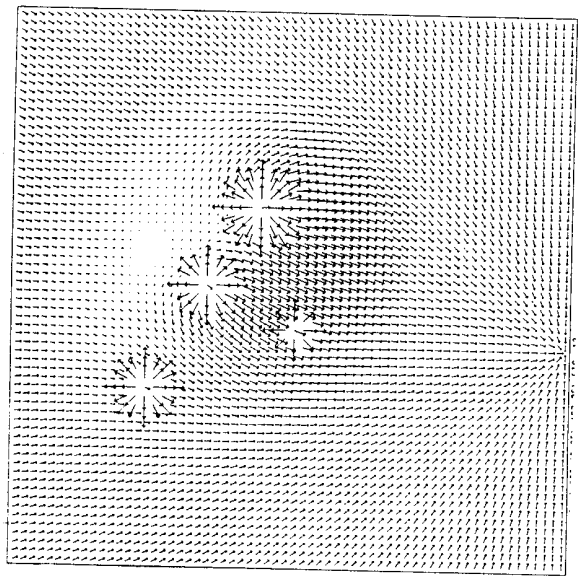
(a)



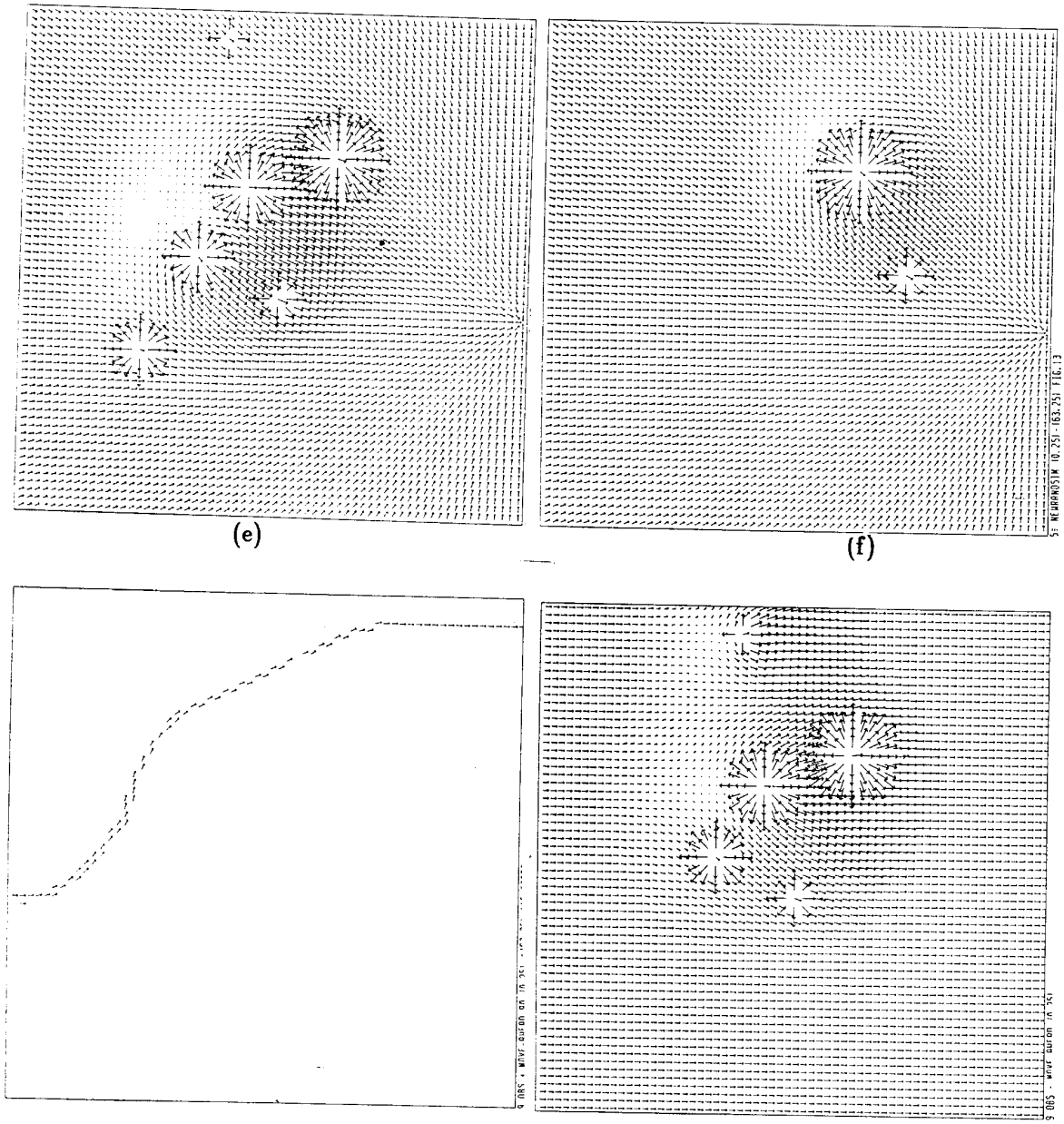
(b)



(c)



(d)



(e) (f) (g) (h)

Figure 54 continued.

e & f) Continuation of sequence shown in Fig. 54 c-d.

g) Robot path using the same starting point as in Fig. 54b but a **move-ahead** SI replaces the **move-to-goal** SI.

h) A typical vector field for the path shown in g). Contrast this against Fig. e to see the distinction between moving towards a specific goal (as in e) or just moving in a general direction (as shown here).

program on its MC68000 processor. This provides a small library of functions that are accessible when a terminal is directly attached to the vehicle. To have the robot communicate with a host computer required the development of a primitive library coded in C and documented in [12]. These routines invoked device drivers, coded by Laboratory for Perceptual Robotics co-worker R. Ellis, which were essential in effective and reliable communication with the host VAX. Most of the primitive functions (many based on the DRV counterparts) are listed in Appendix B. These routines are the ones that generally would need to be recoded for a different set of robotic hardware.

In most instances the asynchronous communication protocol is adequate for the task at hand. The major deficit lies in the transmission of the ultrasonic data over a serial line. Typical time for a single package of ultrasonic data (24 readings) to be sent to the host is on the order of 2-3 seconds. Using a time-shared VAX caused an even greater variability in real-time response. One solution was to boost the process priority to very high levels, effectively shutting or slowing down the other user processes. Although this makes response times more predictable, when dealing with multiple AuRA processes running on the same VAX other components of the overall system suffer. When the system is moved to the Sequent in the future, many of the host processing problems should evaporate. Nonetheless, recoding of HARV's on-board terminal emulator to package the ultrasonic data in a more compact format would still be advisable.

§6. Summary

Motor schemas serve as a means for reactive/reflexive navigation of a mobile robot. This schema-based methodology affords many advantages. These include the use of distributed processing, which facilitates real-time performance, and the modular construction of schemas for ease in the development, testing and debugging of new behavioral and navigational patterns. Complex behavioral patterns can be emulated by the concurrent execution of individual primitive SIs.

The use of velocity fields to reflect the uncertainty associated with a perceptual process is another important advance. By allowing the force produced by a perceived environmental object to vary in relationship to the certainty of the object's identity (whether it be an obstacle, goal path, or whatever), dynamic replanning is trivialized. Since the

sensed environment produces the forces influencing the trajectory of the robot, when the perception of the environment changes, so do the forces acting on the robot, and consequently so does the robot's path. This is all accomplished at a level beneath the *a priori* knowledge representations.

It is interesting to note that what might appear to be a naive approach, the summing of the individual vector outputs of the SIs, works quite well, both in simulations and the experimental results described in Chapter 8. Certainly as the velocity increases, so does the need to account for the velocity of the robot itself in the generation of its trajectory. More complex formulations have been forwarded by both Khatib [64] and Krogh [68] for obstacle avoidance using potential fields. These and other approaches for both potential field formulation and combination mechanisms surely merit additional investigation.

There are times when this methodology of low-level reactive planning will fail, as it suffers from the pitfalls common to potential fields. Failure is detected when the robot's velocity drops to unacceptably low levels (in the case of potential field minima) or by exceeding a hard real-time deadline (in the case of cyclic behavior). At those times, the pilot is reinvoked to conduct a "local-global" form of planning (see Chapter 4). The pilot draws on information present in short-term memory including instantiated meadows that are relevant to this particular leg and a sensor-based world model built by the cartographer. This form of replanning should only be needed rarely as navigational planning helps to ensure avoidance of modeled obstacles. Generally only unmodeled obstacles can lead to the breakdown of schema-based navigation. Higher level knowledge then must be invoked to maneuver the robot out of its dilemma. Most of the time however, schema-based navigation is more than adequate for the task.

A working motor-schema-based navigation system has been implemented as part of the AuRA architecture and is used to conduct actual robot experiments validating the concepts shown only as simulations in this chapter. Many different behaviors have been produced using our mobile robot HARV. These include avoidance, exploration, hall following, navigation amidst obstacles, door entry, impatient waiting, "drunken sailor" single wall following, and follow-the-leader behaviors. Experiments demonstrating these simple to more complex activities are described in Chapter 8. The current experimental testbed is not implemented on a multiprocessor, but it is anticipated that when the schema shell is transferred to the Sequent multiprocessor, the motor schema manager

will soon follow. Work is currently underway in extending the two-dimensional schema system to three dimensions [15], ultimately providing navigational capabilities in both the aerospace and undersea domains.