Avoiding the Past: A Simple but Effective Strategy for Reactive Navigation

Tucker Balch and Ronald Arkin
Mobile Robot Laboratory
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332 U.S.A.

***Abstract*—An issue central to the navigation problem is memory. Traditional systems build symbolic maps of the world for navigational reference. Reactive methods, in contrast, eliminate or minimize the use of memory. These reactive techniques have been remarkably successful at solving a wide range of navigational problems. Some problems, however, still present a challenge to reactive strategies, (box canyons for example). The addition of a local spatial memory allows a robot to avoid areas that have already been visited. "Avoiding the past" offers a solution to the box canyon and other navigational problems. An avoid-past strategy has been implemented using a spatial memory within a schema-based motor control model. Experiments have produced promising results in simulation and on mobile robots.**

I. Introduction

Reactive robotic control systems have emerged as an answer to many of the problems which arise in navigation across unmapped terrain. Reactive systems are characterized by tight sensor to motor couplings. Basic "reactions" to stimuli are combined to generate a resultant behavior. The processing requirements for these systems are greatly reduced in comparison to traditional symbolic methods. The reduced computational demand allows a reactive system to operate in real-time and in changing environments.

Reactive methods differ from more traditional navigational strategies in many respects - memory is certainly one of the most significant. Symbolic systems [1, 12] build and maintain internal representations of the world which are referenced for motion planning. In contrast, many reactive systems maintain a state memory at most, and act immediately on external stimuli [6, 9].

Robots utilizing reactive control exhibit local behavior like walking [7] and obstacle avoidance, but their higher-level performance can be limited. Box canyons, for example, are often a challenge for goal-oriented reactive systems. Such systems are adept at traveling into a canyon without collisions, but they cannot easily determine how to get out. This is often referred to as "the fly at the window problem."

The addition of a local spatial memory to an existing reactive architecture enables a robot to solve the box canyon and other navigational problems. This hybrid approach constitutes a step towards symbolic systems, while maintaining the speed and simplicity of reactive systems.

II. Related Work

A. *Reactive Control for Robot Navigation*

There are many instances of reactive robotic systems (e.g., [6, 11, 13]). Our system, the Autonomous Robot Architecture (AuRA) [3] is a hybrid architecture incorporating aspects of both deliberative and reactive control. The scope of this paper is restricted to the reactive execution component.

Motor schemas are the basic unit of behavioral control in AuRA and are active during navigational execution. Many motor schemas have been developed [2] and applied to real world robotic navigational problems. Those in particular that are relevant to the results of this paper include:

- **Avoid-static-obstacle**: A repulsion is generated by a detected barrier to motion:

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

  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 moving away from obstacle

- **Move-to-goal**: Move towards a perceptually discernible goal.

  $V_{magnitude}$ = fixed gain value
  $V_{direction}$ = in direction towards perceived goal

- **Noise**: a random vector used to circumvent certain problems associated with potential fields methods (a sort of reactive grease) [2, 8].

  $V_{magnitude}$ = fixed gain value
  $V_{direction}$ = random direction for a given time persistence

The velocity outputs generated by each of the independently functioning schemas are summed, normalized, and transmitted to the robot for execution (Figure 2). Specialized perceptual strategies (perceptual schemas) provide only the sensory information that is necessary for the particular behavior being supported; this paradigm is referred to as action-oriented perception [3].

B. *Other Approaches*

Several existing strategies are related to our method in their representation of geographic knowledge or use of memory:

A navigational strategy based on gradient fields was developed by Payton [13]. Gradient fields are constructed to describe mission plans and then movement decisions are based on them. The fields may be derived from world map data or other mission constraints. Our method operates at a lower reactive level and differs in that no *a priori* geographic knowledge is assumed.

As part of a model based on analogical representations, Steels developed a wandering behavior which includes a mechanism for avoiding previously visited areas [15]. The idea is extended in our work to more general navigational problems and is incorporated into a schema-based system.

Yamauchi improved the wall and hallway following performance of a mobile robot by utilizing a behavioral memory [16]. Recent motion commands are integrated into future commands. The robot exhibits momentum in its movements which smooth its trajectory. Our strategy is similar in that it utilizes a memory of recent events, but it is spatial and temporal, rather than behavioral.

Our group has utilized a temporal memory [8] to improve the performance of a schema-based reactive system. An evaluation of progress towards the goal is used to help select numerical parameters for the schemas. If the current set of parameters is not working well, the existing values are adapted in a direction more suitable for the current situation.

III. Task and Approach to the Problem

A. *The Navigational Task*

The specific task examined here is navigation to a known goal position across an unmapped world which may be cluttered with obstacles. Some obstacles may form walls, hallways, or box canyons. Furthermore, the robot's sensors are limited so that it can only perceive obstacles that are close by.

The foraging behavior of ants provides insight into our approach. Some species of ant leave chemical trails behind them as they travel. [10]. Since some ants are nearly blind, these trails are an important navigational tool. Typically, the trails lead from the nest to a food source. Repeated traversals of the path by many ants reinforces the trail. A similar technique may be used for robot navigation, but rather than leave a physical trail, the robot's position is recorded in a grid or "spatial memory."

Fig. 1. A Difficult Navigational Problem. The task is to navigate from the upper left corner to the lower right corner. The robot must find its way with no *a priori* knowledge of obstacles or walls. By avoiding areas that have already been visited, the robot can thoroughly explore the world and reach its goal. The path chosen by our algorithm is depicted by the solid line.

Avoid-past is just one part of the complete navigation system depicted in Figure 2. There are many more schemas available in our system, but **Avoid-past**, **Move-to-goal**, **Avoid-static-obstacle** and **Noise** are sufficient to solve the task at hand. Ideally, the schemas would run concurrently, and their outputs continuously integrated. For simulation purposes, the system runs on a uniprocessor, so the schemas run in turn, with their outputs summed once each time step.

At each time step, the perceptual schema **Past-mapper** updates a spatial map. **Avoid-past** references the map and computes a vector which is combined

Fig. 2. Block Diagram: motor schema-based navigational system. The perceptual schema **Past-mapper** maintains a simple map which is used by the motor schema **Avoid-past** to generate a vector away from previously visited areas. The vision system was implemented on the mobile robot, but not in simulation. It was used only to locate goals, not for obstacle avoidance. Vectors from all the motor schemas are summed and normalized to generate a movement vector.

with vectors generated by **Move-to-goal**, **Avoid-static-obstacle** and **Noise**. The sum of these vectors determines the robot's heading and speed for the next time interval.

B. *The* **Past-Mapper** *Perceptual Schema*

The spatial memory is a two dimensional array of integers which corresponds to the environment to be navigated. Each element of the grid records the number of times the corresponding square patch in the world has been visited. Figure 3 shows how the **Past-mapper** schema updates the spatial map.

This memory updating scheme is roughly equivalent to leaving a chemical trail. The more often an area is visited, the larger the "deposit." This "trail" can be used later for navigational decisions. Rather than mark only one grid point as visited, it was found that marking an entire rectangular region yielded better results. The size of the area to be marked corresponds to the semantic notion of "visited." In other words, to "visit" a room, is it necessary to walk over every inch, or to just step into the center? Using a large mark area is equivalent to the latter.

```
/*--- Update the map ---*/
for i = (X - PAST-MARK) to (X + PAST-MARK) do
      {
      for j = (Y - PAST-MARK) to (Y + PAST-MARK) do
           {
           if A[i,j] < MAX then
                A[i,j] = A[i,j] + 1;
           }
      }
```

Fig. 3. Pseudo-code for the **Past-mapper** schema. The variables used are as follows: A[ ] is the two dimensional spatial map. (X,Y) is the index to the map which corresponds to the robot's position in the world (determined on the robot from shaft encoders). PAST-MARK is the size of the area to be marked, MAX is a ceiling on the number of visits that can be recorded at each grid point.

C. *The* **Avoid-Past** *Motor Schema*

Avoid-past is a motor schema which operates with **Move-to-goal**, **Avoid-static-obstacle** and **Noise**, to form a robust schema-based navigation system. **Avoid-past** uses the spatial memory in conjunction with current positional information to compute a vector away from areas that have already been visited. The more often an area has been visited, the stronger the repulsive force generated. Figure 4 shows how the vector is computed.

The direction of the resultant vector is away from the highest concentration of visits. The magnitude is based on the total number of visits registered in the considered area. The vector is normalized, then multiplied by a gain value. The calculation is essentially the "gradient" found in image processing.

IV. Results In Simulation

A. *Parameters Affecting Performance*

Avoid-past's performance can only be examined in the context of other schemas. Each schema has an associated gain value which multiplies its output vector. The relative values of these gains profoundly affect the resultant behavior.

A robot may become trapped when a wall or canyon stands between it and its goal. In order for a robot to be sufficiently "repelled" by previously visited areas (and thus escape), the **Avoid-past** gain must be set higher than the **Move-to-goal** gain. Otherwise **Move-to-goal** would prevail and the robot would remain trapped. For these experiments, the **Move-to-goal** gain was fixed at 1.0 as a reference, and the **Avoid-past** gain was varied.

Occasionally **Move-to-goal** and **Avoid-past** combine constructively to form a large vector. In these cases the robot may be driven dangerously close to an obstacle which it would otherwise avoid. For this reason, the **Avoid-static-obstacle** gain was set at a relatively high value in comparison to earlier work [8, 2].

```
/*--- initialize component vectors: ---*/
xvec.mag = 0;
xvec.dir = 90; /* direction of x component */
yvec.mag = 0;
yvec.dir = 0;  /* dir of orthogonal y component */
count = 0;

/*--- compute MAGNITUDE of component vectors ---*/
for k = (X - PAST-HORIZON) to (X + PAST-HORIZON) do
  {
  for l = (Y - PAST-HORIZON) to (Y + PAST-HORIZON) do
    {
    if (k < X) then xvec.mag = xvec.mag + A[k,l];
    if (k > X) then xvec.mag = xvec.mag - A[k,l];
    if (l < Y) then yvec.mag = yvec.mag + A[k,l];
    if (l > Y) then yvec.mag = yvec.mag - A[k,l];
    count = count + A[k,l];
    }
  }

/*- sum orthogonal vectors to compute direction -*/
tempvec = sum-vector(xvec, yvec);
pastvec.dir = tempvec.dir;

/*-- compute the the magnitude ---*/
pastvec.mag = past-gain *
  (count / ((PAST-HORIZON * 2) ^ 2 * MAX))

return(pastvec);
```

Fig. 4. Computation of the **Avoid-past** Vector. The variables used are as follows: (X,Y) is the current position (in map space), xvec and yvec are the orthogonal components of the **Avoid-past** vector, count is the sum of all visits to the local area, PAST-HORIZON specifies the size of the "local" area, A[ ] is the spatial map, pastvec is the computed **Avoid-past** vector, and MAX is a ceiling on recorded visits to each grid point.

Other important parameters include those internal to **Avoid-past**. The size of the area to be marked as visited (PAST-MARK) and the size of the area to examine for vector computation (PAST-HORIZON) are critical. If PAST-MARK is too small, the robot will appear to cover the same area repeatedly - it is actually exploring space adjacent to, but not the same as, areas visited earlier. Conversely, if PAST-MARK is set to cover too large an area, the robot might skip places it has not explored. The effect of PAST-HORIZON's value on robot performance is similar. For these experiments, PAST-HORIZON and PAST-MARK were both set to 3 feet so that a 6 foot by 6 foot area is covered - approximately 4 times the robot's 3 foot diameter footprint.

As resolution of the map varies, performance becomes correspondingly more or less jerky. High resolution maps yield smoother results, but increase computational demands since the calculation is $O(n^2)$ with respect to PAST-HORIZON. Experiments showed that a resolution of 0.1 feet was sufficient for smooth performance. That resolution requires *Avoid-past* to examine a 60x60 matrix (3600 points) at each timestep. Despite the large number of points to be examined, **Avoid-past** has not become a bottleneck to performance.

We simulated the movements of a robot through a rectangular area measuring 64 feet on each side. For all simulations, the parameter values were set as follows:

- **Move-to-goal gain** = 1.0
- **Avoid-static-obstacle gain** = 4.0
- **Sphere-of-influence** = 3.0 (a parameter of **Avoid-static-obstacle**)
- **Noise gain** = 0.1
- **Noise persistence** = 2 movement steps
- **Avoid-past gain** = 3.0 (when activated)
- **Map resolution** = 0.1 feet
- **PAST-MARK** = 3 feet
- **PAST-HORIZON** = 3 feet
- **MAX** = 10 visits

Even though specific sets of parameter values work better for each type of scenario, the parameters were fixed for all simulations to allow an objective comparison of performance. These parameters were determined empirically. In separate research [8, 14] our group is investigating methods of automatic parameter selection.

B. *A Random Cluttered Environment*

Consider a typical navigation problem as shown in Figure 5. To proceed from the start point on the right to the goal on the left, the robot must negotiate the randomly distributed obstacles. This problem is solved without the **Avoid-past** schema in 105 movement steps.

Figure 6 shows how the system solves the problem with **Avoid-past** activated. In addition to other advantages, **Avoid-past** reduces and smooths path length. The path is visually smoother and the number of movement steps is reduced by 33% to 70. Even though **Avoid-past** was not strictly necessary for this typical scenario, it clearly improved performance. The improvement is due to a momentum effect induced by the **Avoid-past** schema. This momementum effect is covered later.

C. *A Box Canyon*

Now consider the problem in Figure 7. The robot must travel from the upper left corner to the goal in the lower right corner. A double box canyon is positioned between the starting position and the goal. The robot initially heads straight for the goal. As it encounters the wall of obstacles it is forced into the potential well of the box canyon, where it remains trapped.

When **Avoid-past** is activated (Figure 8), the robot is able to complete the task. Initially, it is drawn into the canyon as before, but as it lingers in the potential well, the **Past-mapper** schema repeatedly marks the area as visited. Eventually the **Avoid-past** vector overwhelms the **Move-to-goal** vector and the robot leaves the canyon. The graph at the top of the figure logs the magnitude of

Fig. 5. Typical random cluttered environment navigated without the **Avoid-past** schema. For this run, only **Move-to-goal**, **Avoid-static-obstacle**, and **Noise** were active. The start point is on the right, and the goal is on the left. The run required 105 movement cycles to complete.

Fig. 6. Typical random cluttered environment navigated with the **Avoid-past** schema active. The path from start to goal is shorter (70 movement steps) and smoother than the run without **Avoid-past**.

Avoid-past as the robot navigates. At first, it lingers below 1.0, but as the robot becomes trapped in the canyon the magnitude climbs above 1.0 allowing it to overpower the potential well generated by **Move-to-goal**.

D. *More Challenging Scenarios*

Even more difficult environments may be navigated by robots using **Avoid-past**. Figure 1 shows how the system can solve a simple maze. Along its path from start to goal, the robot encounters two box canyons and two hallways. Notice that, in addition to solving box canyons, **Avoid-past** improves performance in hallways as it reduces side to side jitter. This is a result of "momentum" that **Avoid-past** adds to the robot's trajectory. Since the robot is repelled from its most recent location its present direction is reinforced. Figure 9 illustrates a solution to another difficult reactive navigational problem.

V. Results on a Mobile Robot

A general motor schema-based reactive control system was developed to test new schemas on mobile robots. The program can be reconfigured to control several different models of Denning robots: a Denning DRV-1 named George, two MRV-2s named Ren and Stimpy, and an MRV-3 named Buzz. **Avoid-past** has been tested successfully on each platform.

In July of 1992 Buzz participated in the AAAI Autonomous Robot Competition [4]. Overall, Buzz finished fourth in the contest. Part of this success stems from the incorporation of **Avoid-past** into Buzz's control strategy. In one part of the competition, Buzz encountered a box canyon which it could not have navigated without **Avoid-past**.

Figure 11 shows Ren navigating a box canyon. The robot is positioned at the far end of the room and must navigate to the goal in the lower left foreground. But two boxes form a canyon (a "box" canyon !) which intervenes. The behavior is similar to the box canyon solution in simulation. Initially, the robot heads towards the goal and lingers in the canyon. Later it is pushed out of the canyon and navigates around it. Eventually, it reaches the goal. For this run the various control parameters were set as follows:

- **Move-to-goal gain** = 1.0
- **Avoid-static-obstacle gain** = 3.0
- **Sphere-of-influence** = 2.5 (a parameter of **Avoid-static-obstacle**
- **Noise gain** = 0.0 (noise was not activated)
- **Avoid-past gain** = 1.4
- **Map resolution** = 0.1 feet
- **PAST-MARK** = 3 feet
- **PAST-HORIZON** = 3 feet
- **MAX** = 10 visits

In simulation and on a mobile robot, noise is often required for successful navigation. Noise may be eliminated or significantly reduced when **Avoid-past** is activated.

VI. Problems and Improvements

A. *Temporal Appropriateness*

The most significant problem encountered centers on the lack of a time-decay mechanism in the current implementation of **Avoid-past**. Recall ants, the trail-makers from

Fig. 7. Attempt to navigate a box canyon without the **Avoid-past** schema. The robot is unable to navigate out of the canyon and becomes trapped.

nature. Their chemical trails evaporate or wash away with time. As a result, the most recent trail is the strongest. This is appropriate since old trails may no longer apply in a dynamic environment.

This flaw became apparent when a "wander" behavior was developed. **Avoid-past** was set to a low gain value and **Move-to-goal** was deactivated. The hope was that by avoiding visited areas, the robot would explore the entire "world." Unexpectedly, the robot lingered for extended periods in corners, sometimes never leaving. The lingering occurred as the robot "boxed itself in" from behind with its spatial memory. If the memory of visits were allowed to decay, the repulsive force behind the robot would clear and it would leave the corner. Interestingly, this behavior does not occur when **Move-to-goal** is activated.

A related problem occurs when **Avoid-past** is implemented on a mobile robot. The spatial map becomes inaccurate as the robot moves about the world. Initially, shaft encoders accurately reflect the robot's position, but as the robot moves, translational, and especially rotational errors degrade positional accuracy. This causes errors in correspondence between the world and the spatial map. Such errors may be minimized by taking advantage of the fact that *recent* relative movement is represented accurately. If data in the spatial map decayed with time, the information would always be locally correct. A trail which decayed with time can be created in computer memory by recording the time of the most recent visit rather than the total number of visits to each point. The vector calculation would weigh grid points by recency rather than accumulation of visits.

Fig. 8. A Box Canyon Navigated With the **Avoid-past** Schema. The robot is forced out of the canyon by the addition of the **Avoid-past** vector. The graph at the top depicts the magnitude of the **Avoid-past** vector, which varies between 0.0 and 2.0 units through execution.

B. Momentum

As was mentioned earlier, **Avoid-past** adds "momentum" to the robot's trajectory. Sometimes this momentum is counter-productive. When the **Avoid-past** gain is set too high the robot rams into obstacles and zooms past the goal (Figure 10). On mobile robots, special care must be taken to select gain values that enable the robot to navigate without collisions. Part of this problem stems from communication delays in our system (computer to robot) which reduce the timeliness of obstacle position information.

VII. Conclusion

This research has shown that local spatial memory, integrated into a reactive navigation system, allows a robot to solve complex navigational problems efficiently. By maintaining and utilizing spatial memory at a low level, key advantages of reactive systems are preserved: simplicity and speed.

The parameters for the simulations and mobile robot experiments described here were determined empirically by the experimenters. In the future we hope to more completely analyze how performance varies as these parameters change. Other future work will include the development of a "decay" mechanism for the spatial map, and an application of the **Avoid-past**) schema to multi-agent environments.

VIII. Acknowledgments

The Mobile Robot Laboratory is supported in part by NSF grants IRI-9100149 and IRI-9113747 and the Georgia Tech Material Handling Research Center. The authors would like to thank Doug MacKenzie, William Wester, and Maria Hybinette for their help in the preparation of this paper. Andrew Henshaw improved the normalization of the **Avoid-past** vector and proposed the use of timestamps in the spatial memory.

Fig. 9. Another Challenging Navigational Problem.

Fig. 10. "Momentum" - If the **Avoid-past** gain is set too high, the robot will ram into obstacles and zoom past the goal.

References

[1] Albus, J., McCain, H., and Lumia, R., NBS Standard Reference Model for Telerobot Control System Architecture (NASREM). NBS Technical Note D.C., 1987.

[2] Arkin, R.C., "Motor Schema-Based Mobile Robot Navigation", *International Journal of Robotics Research*, Vol. 8, No. 4, August 1989, pp. 92-112.

[3] Arkin, R.C., "The Impact of Cybernetics on the Design of a Mobile Robot System: A Case Study", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 6, Nov/Dec 1990, pp. 1245-1257.

[4] Bonasso, P., Dean, T., "The AAAI Mobile Robot Competition," to appear, *AI Magazine.*

[5] Brooks, R., "Intelligence Without Representation". *Artificial Intelligence*, 1991.

[6] Brooks, R., "A Robust Layered Control System for a Mobile Robot", *IEEE Jour. of Robotics and Auto.*, **RA-2**, 1, p. 14, (1986).

[7] Brooks, R.,. "Elephants Don't Play Chess", *Robotics and Autonomous Systems, Vol. 6, Nos. 1, 2* pp. 3-16, 1990.

[8] Clark, R.J., Arkin, R.C., 1 and Ram, A., "Learning Momentum: On-line Performance Enhancement for Reactive Systems", Proc. *IEEE Conf. on Robotics and Automation*, Nice, France, May 1992, pp. 111-116.

[9] Connell, J.H.,*Minimalist Mobile Robotics: A Colony-style Architecture for an Artificial Creature*, Academic Press, 1990.

[10] Goss, S., Beckers, R., Deneubourg, J.L., Aron S.,, Pasteels, J.M. "How Trail Laying and Trail Following Can Solve Foraging Problems for Ant Colonies", *NATO ASI Series*, Vol. G 20, Springer-Verlag, Berlin (1990).

[11] Kaelbling, L., and Rosenschein, S., "Action and planning in embedded agents", in *Designing Autonomous Agents*, p. 35, (P. Maes, ed.), MIT Press, Cambridge, MA, 1990.

[12] N. Nilsson, N., (Ed.) "Shakey the Robot", SRI International Tech. Note 323 (1984).

[13] Payton, D., "Internalized Plans: A Representation for Action Resources", *Designing Autonomous Agents*, Maes, P. Ed., MIT Press (1991).

[14] Pearce, M., Arkin, R.C., Ram, A., "The Learning of Reactive Control Parameters Through Genetic Algorithms", *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Raleigh, NC, 1992, pp. 130-137.

[15] Steels, L., "Exploiting Analogical Representations", *Designing Autonomous Agents*, Maes, P. Ed., MIT Press (1991).

[16] Yamauchi, B., "Behavioral Memory Techniques for Robot Navigation", Research Paper, Hughes Research Laboratories, Malibu CA.

(a) Starting configuration, the goal is in the foreground.

(b) The robot maneuvers into the canyon.

(c) The robot is repelled from the canyon.

(d) The robot maneuvers around the canyon.

(e) Approaching the goal.

(f) At the goal.