Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles

Bruce H. Krogh* **Charles E. Thorpe****

*Department of Electrical & Computer Engineering and the Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA 15213, USA

**Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA 15213, USA

ABSTRACT

A method is presented for combining two previously proposed algorithms for path-planning and dynamic steering control into a computationally feasible scheme for real-time feedback control of autonomous vehicles in uncertain environments. In the proposed approach to vehicle guidance and control, Path Relaxation is used to compute *critical points* along a globally desirable path using *a priori* information and sensor data[1]. Generalized potential fields are then used for local feedback control to drive the vehicle along a collision-free path using the critical points as subgoals[2]. Simulation results are presented to demonstrate the control scheme.

INTRODUCTION

System architectures and algorithms proposed for guidance and control of autonomous vehicles normally reflect a hierarchical decomposition of the problem[3, 4, 5]. Basic levels in the system hierarchy are illustrated in Fig. 1. At the highest level the problem is represented symbolically and artificial intelligence techniques are used to perform high-level planning. At the lowest level, servo controllers track reference signals, implementing the desired dynamic trajectory. This paper concerns the vehicle guidance problem at the intermediate levels. Using two previously proposed algorithms we integrate the functions of geometric path planning and dynamic steering control. The path-planning algorithm generates a set of critical points along a globally desirable path, while the steering control algorithm performs local navigation and obstacle avoidance.

Based on an internal map of the environment developed from *a priori* information and high-level sensory data, Path Relaxation selects a *minimum cost* collision-free path to the goal[1]. This path could be executed directly by driving the vehicle through each point on the path in turn. Such a simple-minded scheme, however, ignores vehicle dynamics, forces the trajectory to go through each point, and is unable to use new sensory data without complete replanning. A better strategy is to generate a set of *critical points* along the path and to use them as subgoals for dynamic steering control using local feedback information. Dynamic steering control is accomplished in our system using Generalized Potential Fields[2].

This makes for a good division of the problem: Path Relaxation takes care of global issues, like avoiding dead ends, finding an overall optimal or near-optimal path, and deciding between areas with different terrain or better visibility. Local issues, such as cutting corners, slowing the vehicle to maneuver in tight spots, and generating smooth transitions from one step to the next, are all handled by the Generalized Potential Field feedback algorithm. This dynamic steering control also reacts to new sensory data as the vehicle moves, taking into account updated obstacle positions without having to invoke the global path planner. This two-level scheme is computationally feasible for *real-time* implementation. Since the steering control algorithm handles local details, the path planner can work on a *coarse* scale. The set of critical points must be re-evaluated only when there are extreme changes in the environmental information. Moreover, the two levels of control can be executed in *parallel* with the slower sensor processing and path-planning algorithm generating future subgoals while the steering control algorithm guides the system through the current set of critical points.

The following section reviews the path-relaxation algorithm and its implementation for selecting the critical points to be passed to the steering control algorithm. We then describe the *generalized potential field* approach to dynamic steering control using the critical points and real-time sensory information to determine the desired system acceleration at each control instant. Features of the proposed scheme are illustrated by simulation including an example where the environment encountered by the steering control algorithm differs significantly from the model used by the path-planning algorithm to generate the subgoals. The simulation results demonstrate the viability of the proposed approach for real-time guidance and control of autonomous vehicles. Directions for future research and issues currently being investigated are discussed in the concluding section.

PATH RELAXATION[1]

For robot path planning, we want to find an optimal path to a destination through a field of obstacles. Path Relaxation is a two-step process that tries to find the path with the lowest total cost. The cost of a path is a combination of several factors, including distance traveled, nearness to objects, traversability of the terrain, and uncertainty about the area. The first step of path relaxation finds a preliminary path on an eight-connected grid of points. The second step adjusts, or "relaxes", the position of each preliminary path point to improve the path.

One advantage of path relaxation is that it allows many different factors to be considered in choosing a path. Typical path planning algorithms evaluate the cost of alternative paths solely on the basis of path length. The cost function used by Path Relaxation, in contrast, also includes how close the path comes to objects (the further away, the lower the cost) and can include penalties for traveling through particular areas, such as areas outside the field of view of the robot. The effect is to produce paths that neither clip the corners of obstacles nor make wide deviations around isolated objects, and that prefer to stay in mapped areas unless a path through unmapped regions is substantially shorter. Other factors, based on *a priori* information or sensor data, such as roughness of terrain or visibility of landmarks, can be added for particular environments.

GRID SEARCH

The first stage of path relaxation finds an approximate path on a grid. The grid mesh size can be as large as the minimum dimension of the robot and still guarantee that no path will be missed. Once the grid has been set up, the next step is to assign costs to paths on the grid and then to search for the best path along the grid from the start to the goal. "Best", in this case, is a compromise among three potentially conflicting requirements: shorter path length, greater margin away from obstacles, and less distance in high-cost areas. These three requirements are explicitly balanced by the way path costs are calculated. A path's cost is the sum of the costs of the nodes through which it passes, each multiplied by the distance to the adjacent nodes. (In a 4-connected graph all lengths are the same, but in an 8-connected graph we have to distinguish between orthogonal and diagonal links.) The node costs consist of three parts to explicitly represent the three criteria.

1. Cost for distance. Each node starts out with a cost of one unit, for length traveled.

2. Cost for near objects. Each object near a node adds to that node's cost. The nearer the obstacle, the more cost it adds. The exact slope of the cost function will depend on the accuracy of the vehicle (a more accurate vehicle can afford to come closer to objects), and the vehicle's speed (a faster vehicle can afford to go farther out of its way), among other factors.

3. Cost for being within or near an unmapped or other high-cost region. The cost for traveling in one of these regions depends on the particular mission of the vehicle. For instance, if the goal is primarily exploration, then unmapped areas could have lower costs than other areas. There is also a cost added for being near an unmapped region, using the same sort of function of distance as is used for obstacles. This provides a buffer to keep paths from coming too close to potentially unmapped hazards.

For costs 2 and 3 we use a cubic cost function, as illustrated in Fig. 2, which ranges from 0 at some maximum distance, set by the user, to the obstacle's maximum cost at 0 distance. This function has the advantages of giving good saddles between neighboring obstacles, being easy to compute, and being bounded in a local area.

The cost calculated for each node is based on the distances to nearby obstacles and whether that node is within a high-cost area. Next, each node is linked to its 8 neighbors. The start and goal locations do not necessarily lie on grid points, so special nodes need to be created for them and linked into the graph.

The system then searches this graph for the minimum-cost path from the start to the goal, using a standard A* search[6]. The estimated total cost of a path, used by A* to pick which node to expand next, is the sum of the cost so far plus the straight-line distance from the current location to the goal. This has the effect, in regions of equal cost, of finding the path that most closely approximates the straight-line path to the goal.

RELAXATION

Grid search finds an approximate path; the next step is an optimization step that fine-tunes the location of each node on the path to minimize the total cost. One way to do this would be to define precisely the cost of the path by a set of non-linear equations and solve them simultaneously to determine the optimal position for each node. This approach is not, in general, computationally feasible. The approach used here is a relaxation method. Each node's position is adjusted in turn, using only local information to minimize the cost of the path sections on either side of the node. Since moving one node may affect the cost of its neighbors, the entire procedure is repeated until no node moves farther than some small amount.

Node motion has to be restricted. If nodes were allowed to move in any direction, they would all end up at low cost points, with many nodes bunched together and a few long links between them. This would not give a very good picture of the actual cost along the path. So in order to keep the nodes spread out, a node's motion is restricted to be perpendicular to a line between the preceding and following nodes. Furthermore, at any one step a node is allowed to move no more than one unit. As a node moves, all three factors of cost are affected: distance traveled (from the preceding node, via this node, to the next node), proximity to objects, and relationship to unmapped regions. The combination of these factors makes it difficult to directly solve for minimum cost node position. Instead, a binary search is used to find the minimum cost node position to the desired accuracy.

The relaxation step has the effect of turning jagged lines into straight ones where possible, of finding the "saddle" in the cost function between two objects, and of curving around isolated objects. At the boundary between two regions of different cost, the minimum cost path will "refract" in the same way that a ray of light refracts crossing a boundary between two substances. The laws of refraction state that a ray of light crossing a border between two substances with different transmission velocities will follow the path that minimizes transmission time. The path found by path relaxation will "refract" in an analogous way, minimizing cost rather than time.

The output of path relaxation is a sequence of positions $\{\mathbf{x}_o,...\mathbf{x}_n=\mathbf{x}_g\}$ from the initial position of the vehicle $\mathbf{x}_o$ to the goal $\mathbf{x}_g$. As demonstrated by the examples below, these positions can be generated on a relatively coarse scale. The steering control algorithm performs the function of local navigation, using the set $\{\mathbf{x}_o,...,\mathbf{x}_n\}$ as critical points or subgoals along the dynamic trajectory. Determining the best set of critical points is an area for future research.

DYNAMIC STEERING CONTROL

In this section we present a feedback algorithm for the dynamic steering control of autonomous vehicles using local feedback information. Given a set of subgoals from the path-planning algorithm which roughly define a desirable path to the goal, the steering control algorithm must determine the appropriate acceleration of the system at each instant to guide it along a collision-free trajectory to the goal. Since the environment is not known entirely at the path-planning level, our steering control algorithm incorporates real-time sensor information; that is, the steering control is specified as a *feedback law* rather than an *open-loop* algorithm.

The steering control algorithm is based on the generalized potential field approach for obstacle avoidance control proposed by Krogh[2]. In other potential field methods proposed by Hogan and Khatib, an artificial *force* vector is introduced as the sum of the gradients of position-dependent potential fields for the obstacles and goal[7, 8]. The introduction of *generalized potential fields*, that is, potential fields which are both position and velocity dependent, eliminates the possibility of the *stalling* at local minima in the potential field[9]. Other advantages of the generalized potential field approach are briefly described in this section[2].

PROBLEM FORMULATION

As in the previous section, we consider the problem of obstacle avoidance in the plane, modeling the system as a point at position $\mathbf{x}=[x_1,x_2]^T$ with velocity $\mathbf{v}=[v_1,v_2]^T$ (in Cartesian coordinates). The steering control is the acceleration vector $\mathbf{u}=[u_1,u_2]^T$ which is assumed to be constrained in magnitude, that is,

$$\|\mathbf{u}\| \leq \alpha. \tag{1}$$

The velocity can also be limited by a maximum speed, κ, that is

$$\|\mathbf{v}\| \leq \kappa. \tag{2}$$

Thus, the dynamics of the system are given by

$$\ddot{\mathbf{x}}=\dot{\mathbf{v}}=\mathbf{u} \tag{3}$$

subject to the constraints (1), (2).

This simple dynamic model is used to facilitate the *real-time* computation of the desired system acceleration $\mathbf{u}$. The steering control problem is most easily formulated in the global coordinate frame of the system environment since set of critical points, distances to obstacles and the goal position are specified in these coordinates. The mapping of the actual system dynamic model and control constraints (most easily specified in *generalized* coordinates) into global coordinates is discussed by Krogh and Graettinger[10]. To use the model (3) it is necessary to choose the acceleration and velocity constraints (1), (2) so that the desired acceleration computed by the steering control algorithm is feasible. A method for computing these *maneuverability constraints* and applications to autonomous vehicle control are currently under investigation[10]. In the present paper we use the model (1)-(3) to illustrate our basic approach to steering control.

The steering control objective is to drive the system along a collision-free path from its initial position $\mathbf{x}_o$ to a specified goal position $\mathbf{x}_g$. The higher-level path planning algorithm provides a set of critical points $\{\mathbf{x}_1,\ldots,\mathbf{x}_n=\mathbf{x}_g\}$ which represents the best path in terms of global information about the locations of obstacles and the environment. As the vehicle progresses through the environment, real-time sensing provides more detailed and more exact feedback information on the locations of the obstacles within the local vicinity of the vehicle. Using these data, the steering control algorithm must determine the appropriate acceleration $\mathbf{u}$ at each instant.

Computational complexity precludes the use of optimal control or dynamic programming for *real-time* steering control. The remainder of this section describes a computationally feasible algorithm for determining the acceleration based on the concept of *satisficing strategies* rather than *optimality*[11]. The underlying principle for the proposed feedback algorithm is to choose the control at each instant based on the available *local* information so as to guarantee the *acceptability* (but not necessarily the optimality) of the resulting global trajectory.

THE FEEDBACK ALGORITHM[2]

At each control instant the acceleration $\mathbf{u}$ is chosen to maximize the rate of decrease in a generalized potential function $P(\mathbf{x},\mathbf{v})$, subject to the constraints (1), (2) and a further constraint imposed by the finite sampling time (discussed below). Thus, $\mathbf{u}$ is chosen to minimize

$$\dot{P}(\mathbf{x},\mathbf{v})=\nabla_x P\ \dot{\mathbf{x}}+\nabla_v P\ \dot{\mathbf{v}} \tag{4}$$

which gives the control

$$\mathbf{u}=-\alpha\|\nabla_v P\|^{-1}\nabla_v P \tag{5}$$

when only constraint (1) is active.

The potential function is given by

$$P(\mathbf{x},\mathbf{v})=P_g(\mathbf{x},\mathbf{v})+P_o(\mathbf{x},\mathbf{v}) \tag{6}$$

where $P_g(\mathbf{x},\mathbf{v})$, $P_o(\mathbf{x},\mathbf{v})$ are independent potential functions for the goal (or subgoal) and obstacles respectively. As described below these potential functions are defined so that the system is "attracted" by the goal and "repulsed" by the obstacles. The dependence of P_g and P_o on the system velocity as well as position takes into account the dynamic aspects of the trajectory and control constraints.

GOAL POTENTIAL

To compute the goal-attraction potential, $P_g(\mathbf{x},\mathbf{v})$, an appropriate subgoal is first selected from the set of critical points $\{\mathbf{x}_1,\ldots,\mathbf{x}_n=\mathbf{x}_g\}$. The critical points are pursued in sequence; the subgoal is updated to the next critical point when the system begins to decelerate toward the current subgoal. If the next critical point is not visible from the current system position $\mathbf{x}$, an intermediate subgoal is chosen as the edge of the obstructing obstacle which is closest to the desired subgoal. Thus, if the real-time sensing indicates the presence of an obstacle which was not taken into account by the path-planner, this intermediate subgoal directs the system around the obstacle. Note that this leads to a trajectory that follows the general path specified by the sequence of critical points, but the vehicle does not stop or even pass through each $\mathbf{x}_j$.

For the selected subgoal $\mathbf{x}_j$ the goal attraction potential $P_g(\mathbf{x},\mathbf{v})$ is computed so that the gradient results in an acceleration which will eventually bring the system to rest at $\mathbf{x}_g$. Krogh proposed a heuristic decomposition was used to compute orthogonal components of the goal attracting potential gradient[10]. This was based on the time required to reach the goal using the maximum acceleration available in each orthogonal direction, ignoring the obstacles and the acceleration allocation for the orthogonal direction. Recently, a computationally feasible method was derived to find the optimal time to go for the two-dimensional problem[12]. Using this result we define $P_g(\mathbf{x},\mathbf{v})$ as the minimum time to reach the goal subject to the acceleration limit (1). Thus, $-\nabla P_g(\mathbf{x},\mathbf{v})$ directs the acceleration as the optimal control if no obstacles are present.

OBSTACLE POTENTIAL

A potential function can be computed for every visible obstacle in the environment[2]. In the present formulation of the algorithm we compute $P_o(\mathbf{x},\mathbf{v})$ only for visible obstacles in the direction of the current system velocty $\mathbf{v}$. This reduces the computational burden while yielding an acceleration which avoids collisions with the local obstacles. If an obstacle is present the obstacle potential is computed as proposed by Krogh[2]. In particular, $P_o(\mathbf{x},\mathbf{v})$ is defined as the inverse of the *reserve avoidance time*, $\tau_M - \tau_m$, where the *minimum avoidance time* τ_m is the minimum time in which the

velocity toward the obstacle can be brought to zero using maximum deceleration, and the *maximum avoidance time* τ_M is the maximum time in which the velocity toward the obstacle can be brought to zero under constant deceleration without hitting the obstacle. This gives

$$P_o(\mathbf{x},\mathbf{v}) = \quad v_o(v_o^2 - 2\alpha x_o)^{-1}$$

where $-v_o$ is the velocity toward the obstacle and x_o is the distance to the obstacle barrier. We note that $P_o(\mathbf{x},\mathbf{v})$ grows to infinity as the capability to avoid crossing the obstacle barrier goes to zero.

REAL-TIME IMPLEMENTATION

The generalized potential field feedback algorithm described above requires the computation of the feasible acceleration which minimizes (4). Thus, only the gradients of $P(\mathbf{x},\mathbf{v})$ (6) with respect to $\mathbf{v}$ are required, and this is computed analytically for real-time implementation. The simple magnitude constraints (1), (2) permit an analytical solution of the minimization problem (e.g., (5)) which means the control at each instant is computable directly from sensor data giving the distances to visible obstacles.

One practical issue is the finite sampling time for processing sensor data. To account for this delay, an additional constraint is imposed on the acceleration at each control instant. A sampled-data control is assumed with constant acceleration during the sampling interval. It is necessary to assure that when the new sensor data is available, the system will still have the capability to avoid collisions with the obstacles. This is guaranteed by limiting the acceleration in the direction of the obstacle so that the position and velocity at the next control instant will still be safe with respect to the obstacle.

Analytical methods for guaranteeing the efficacy of this feedback algorithm have been proposed and are currently being extended to accommodate more realistic steering control problems[9]. In the next section examples are presented to illustrate the integration of the feedback algorithm with the path-planning algorithm of the previous section.

EXAMPLES

As an example, consider the obstacle-strewn environment illustrated in Fig. 3 which includes a detailed path from $\mathbf{x}_o$ to $\mathbf{x}_g$ generated by the Path Relaxation algorithm. Passing these points to the steering control algorithm, the resulting dynamic trajectory is shown in Fig. 4. The dynamic steering control, which is based on only *local* information about the obstacle locations, generates a dynamic trajectory to the goal which follows the general path received from the path-planner.

One motivation for introducing feedback in the steering control loop is to reduce the required level of detail at the path planning stage. In Fig. 5 a much coarser set *critical points* is specified along the desired path to the goal. Using these points as subgoals, the feedback algorithm again generates a smooth dynamic trajectory to the goal, as shown in Fig. 6. Thus, the local feedback algorithm produces an acceptable trajectory without having a detailed path plan.

Finally, Fig. 7 illustrates the ability of the feedback algorithm to use real-time information not available to the path-planner. The same set of critical points (Fig. 5) were used as subgoals, but as the system moved through the obstacles, three new obstacles were detected which had not been taken into account by the path planner. As shown in Fig. 7, the generalized potential field algorithm successfully guided the system along the desired path while avoiding collisions with the new obstacles.

FUTURE RESEARCH

This paper presents and demonstrates the basic elements for an integrated approach to path planning and dynamic steering control for autonomous vehicles. The concepts have been demonstrated by simulation and work is currently underway to implement the proposed approach to guidance and control for an actual test vehicle.

There are several directions for future research, including:

- methods for determining the appropriate set of *critical points* at the path planning level
- representation of *a priori* information at the steering control level for interpreting real time data
- criteria for deciding when the path should be re-planned, that is, when the steering control algorithm should request new points from the path-planner
- incorporation of learning in the high-level feedback loop

ACKNOWLEDGEMENTS

The authors thank Dai Feng for programming and generating the graphics for the examples, and Barbara Stone for helping prepare the manuscript. This research supported in part by the National Science Foundation under grant ECS-8404607 and in part by DARPA under contract number DACA76-85-C-0003.

References

1. C.E. Thorpe, "Path Relaxation: Path Planning for a Mobile Robot", *American Association for Artificial Intelligence Conference*, Austin, Texas, Aug 1984.
2. B.H. Krogh, "A Generalized Potential Field Approach to Obstacle Avoidance Control", *Robotics International Robotics Research Conference*, Bethlehem, PA, August 1984.
3. G. Giralt, R. Chatila, M. Vaisset, "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots", *1st International Symposium of Robotic Research*, Bretton Woods, NH, September 1983.
4. E. Koch, et al, "Simulation of Path Planning for a System with Vision and Map Updating", *1985 IEEE International Conf on Robotics and Automation*, St. Louis, MO, Mar 1985.
5. J.L. Crowley, "Navigation for an Intelligent Mobile Robot", *IEEE Journal of Robotics and Automation*, Vol. RA-1, No. 1, Mar 1985, pp. 31-41.
6. N. Nilsson, *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, 1971.
7. N. Hogan, "Impedance Control: An Approach to Manipulation: Parts I-III", *ASME J Dynamic Syst., Meas., and Control*, Vol. 107, March 1985, pp. 1-24.
8. O. Khatib, "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots", *1985 IEEE Conf. on Robotics and Automation*, March 1985, pp. 500-505.
9. B.H. Krogh, "Guaranteed Steering Control", *1985 American Control Conference*, Boston, June 1985.
10. B.H. Krogh and T. Graettinger, "Maneuverability Constraints for Supervisory Steering Control", *24th Conference on Decision and Control*, Fort Lauderdale, FL, December 1985.

11. H. Simon, *The Sciences of the Artificial,* MIT Press, Second Edition, 1981.

12. D. Feng and B.H. Krogh, "Acceleration-Constrained Time-Optimal Control in Three Dimensions", *IEEE Trans. on Automatic Control,* submitted for publication

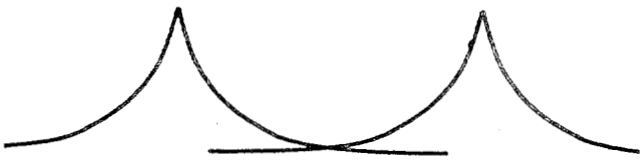

Figure 2: Cross-section of cubic cost functions for two point obstacles.

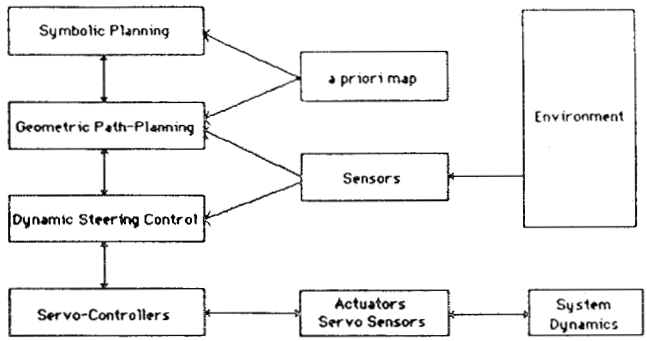

Figure 1: Hierarchical allocation of functions for autonomous vehicle control.

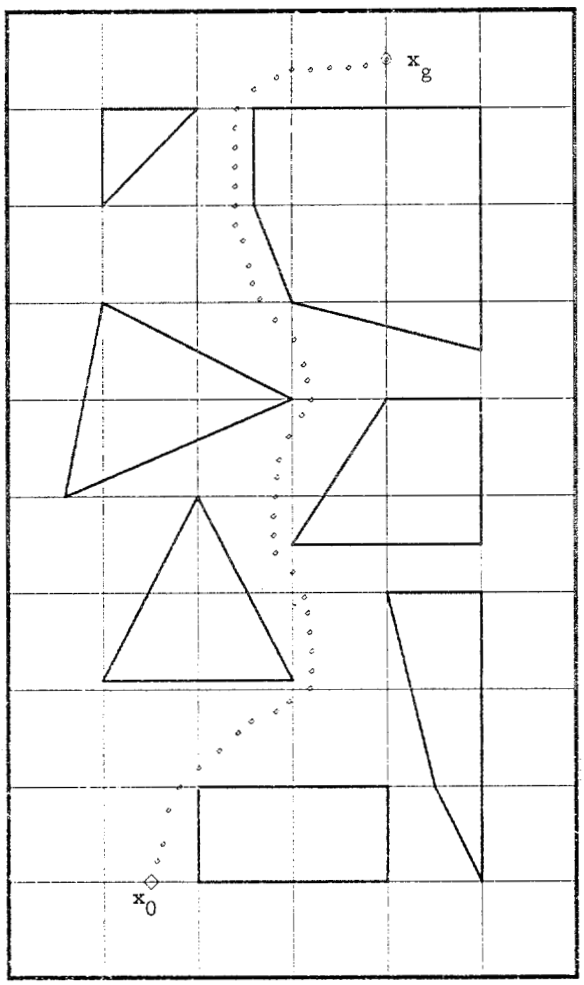

Figure 3: Detailed path plan from Path Relaxation algorithm.

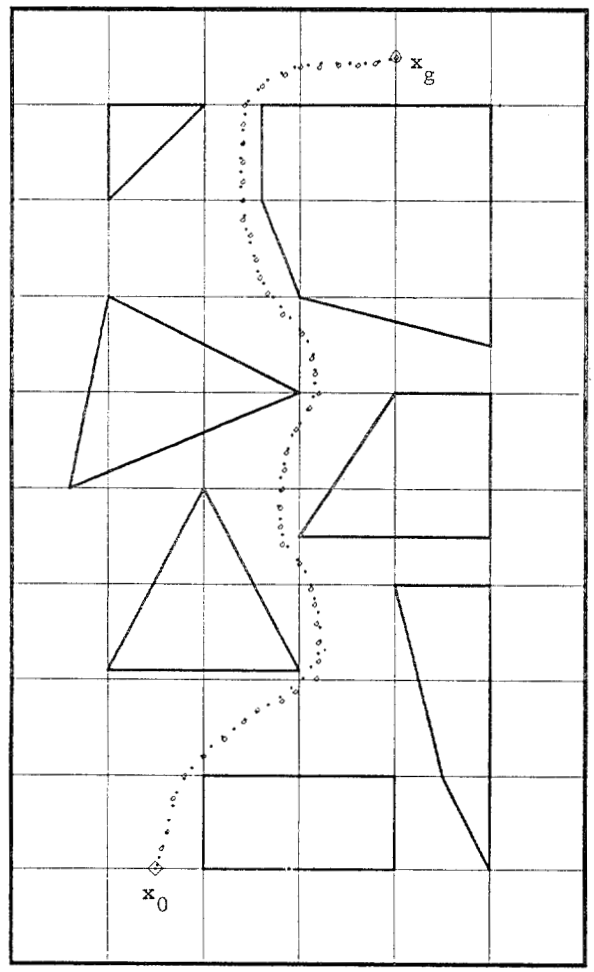

Figure 4: Dynamic trajectory following detailed path from Fig. 3

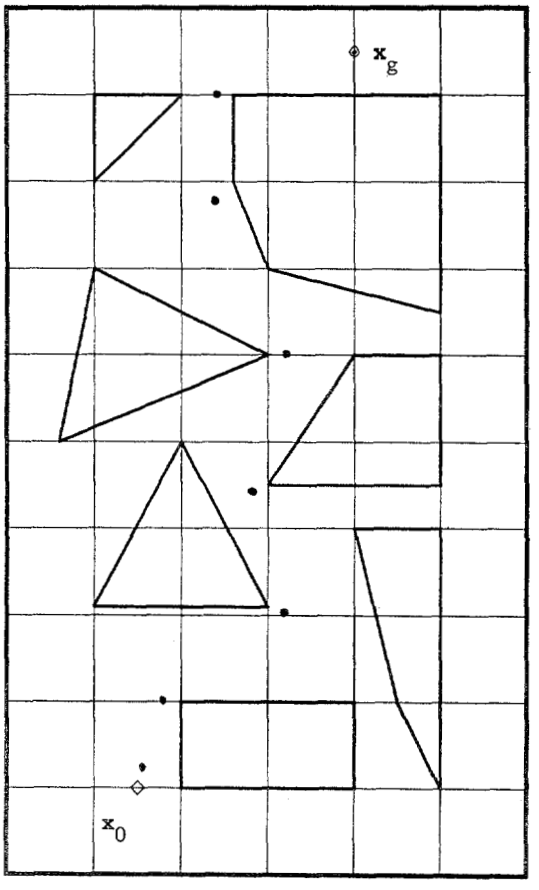

Figure 5: Sparse set of critical points along desired path.

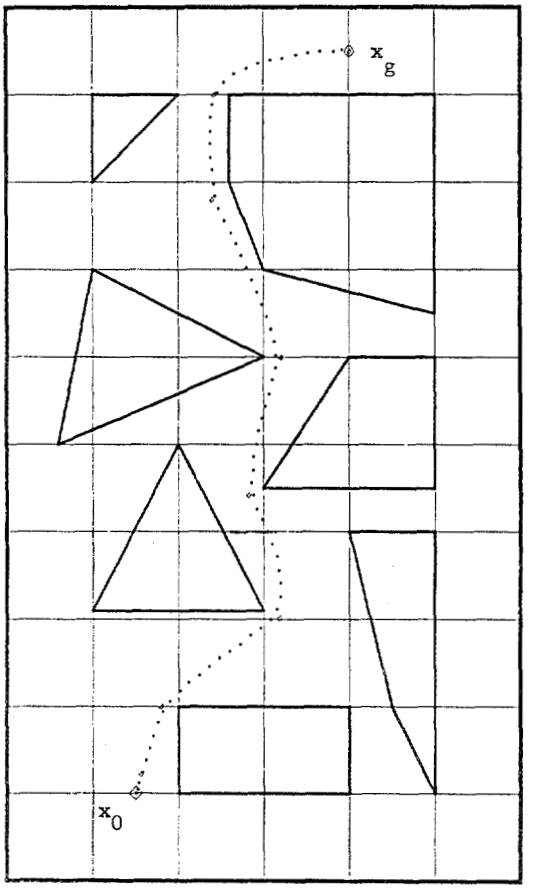

Figure 6: Dynamic trajectory using sparse set of critical points from Fig. 5 as subgoals.

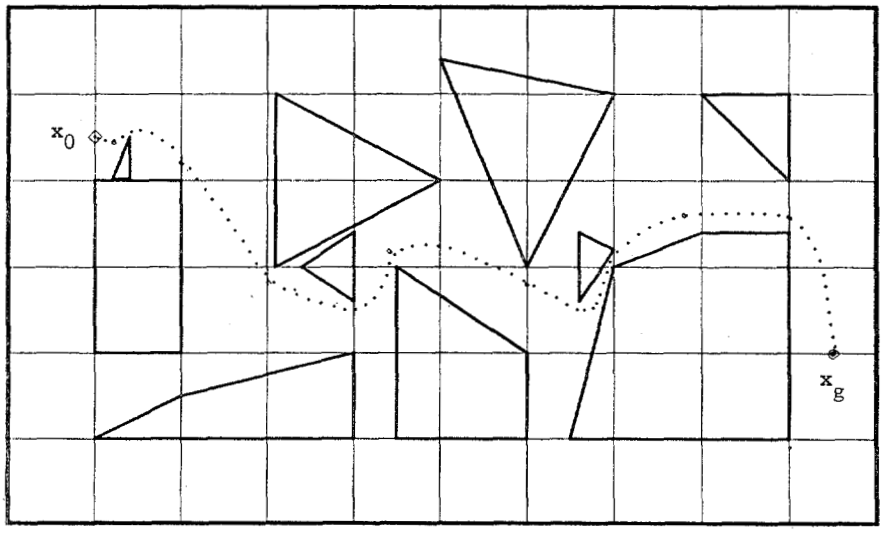

Figure 7: Dynamic steering control for local obstacle avoidance using sparse set of critical points from Fig. 5.