GA-BASED MOTION PLANNING FOR UNDERWATER ROBOTIC VEHICLES

Kazuo Sugihara  
Dept. of Information and Computer Sciences  
University of Hawaii at Manoa  
Honolulu, HI 96822  
Tel: (808)956-6049  
Fax: (808)956-3548  
sugihara@hawaii.edu

Junku Yuh  
Dept. of Mechanical Engineering  
University of Hawaii at Manoa  
Honolulu, HI 96822  
Tel: (808)956-6579  
Fax: (808)956-2373  
yuh@wiliki.eng.hawaii.edu

Abstract

This paper discusses GA-based motion planning in 3D space for underwater robotic vehicles. We propose a genetic algorithm (GA) for path planning which generates a collision-free path in an environment with two kinds of obstacles: Solid and hazardous obstacles such that a path cannot intersect solid obstacles while it can intersect hazardous obstacles at the expense of extra costs. The most important advantage of our GA-based approach is the adaptivity in the sense that a GA can respond to environmental changes (e.g., encounter with an unknown obstacle) and adjust a path “globally” to the new environment. Therefore, the GA approach can incorporate on-line motion planning with off-line motion planning. In addition, the coding used in the GA can be extended so that path planning and trajectory planning are solved simultaneously.

Introduction

Motion planning of an autonomous underwater vehicle (AUV) can be decomposed into path planning and trajectory planning, although they are not completely independent of each other. Path planning is to generate a collision-free path in an environment with obstacles and optimize it with respect to some criterion. Trajectory planning is to schedule the movement of an AUV along the planned path. This paper primarily addresses the path planning of an AUV and proposes adaptive path planning which employs a genetic algorithm (GA) [1, 5] to modify the existing path whenever an environmental change occurs (e.g., the AUV detects an unknown obstacle).

There have been a number of methods proposed for motion planning of a mobile robot [8]. An algorithm for motion planning is said to be off-line if the environment is a known static terrain and it generates a plan in advance. It is said to be on-line if it is capable of producing a new plan in response to environmental changes. Most of the previously proposed algorithms are off-line.
In addition, many of them were devised for path planning in 2D space. In 3D space, even finding just a geometrically shortest path amidst obstacles is computationally intractable [8]. Thus, the others employ heuristics, artificial potential fields, A* search, case-based reasoning, etc., in order to deal with path planning in 3D space.

For example, Charles W. Warren [14] used potential fields to find a collision-free path and proposed heuristics to reduce the possibility of being trapped at a local minimum. Kevin P. Carroll et al. [3] used a traditional optimization technique A* to solve a path minimization problem with various constraints. Vasudevan et al. [13] used an AI technique called case-based reasoning which combines previously executed plans and modifies them to generate a new path plan.

These approaches have drawbacks. The possibility of being trapped at a local minimum is essentially inevitable in the potential field approach. Moreover, it is not straightforward to apply it to path planning with complex optimization criteria and constraints rather than just finding a shortest path. Although A* search is general enough to be applied to problems in 3D or higher dimensional space, both time and space requirements for computation of A* become excessive in practice as the dimensions of the problem increase. The case-based planning approach adjusts a path for each obstacle “locally” instead of optimizing globally and sometimes produces unnecessary zigzag moves.

Recently, applications of GA to path planning or trajectory planning have been recognized [2, 4, 7, 9, 10, 15]. GA is a search strategy using a mechanism analogous to evolution of life in nature. A population of solutions is maintained and the solutions are repeatedly transformed by genetic operators such as crossover and mutation. It has widely been recognized that GA works even for complex problems such that traditional algorithms cannot find a satisfactory solution within a reasonable amount of time [6].

However, the previously proposed GAs have drawbacks and could not fully exploit benefits and ability of GAs. First, a coding which maps solutions to symbolic representations often employs a variable-length string such as a sequence of points or line segments in path planning [9, 10]. Since the string length of an optimal solution is not known, the configuration (e.g., genetic operators and their parameter values) of a GA must be designed in an ad hoc manner to guarantee generation of strings of any lengths from an arbitrarily given initial population. Second, the GAs were designed to solve either path planning or trajectory planning, but not both.

In this paper, we propose a GA in which a path in 3D space is coded into a binary string representing a sequence of (direction, distance) pairs, assuming that the path is a sequence of cells in a 3D grid. The coding is designed so that the length of a binary string for each solution is fixed at $O(n \log n)$ bits, where $n$ is the size of the grid. Since strings in a population are of the same fixed length, the standard GA works well.

The GA has two advantages. First, it is adaptive in the sense that it can respond to environmental changes (e.g., encounter with an unknown obstacle) and adjust a path “globally” to the new environment. Thus, the GA is applicable to both off-line and on-line motion planning. Second, the dimension of space has much less effect on performance in the GA approach than the other approaches. It is useful even in higher dimensions and can be generalized to solve both path planning and trajectory planning simultaneously.
Problem Formulation

Suppose that an autonomous underwater vehicle (AUV) needs to move from a start point \( s \) to an end point \( e \) in 3D space. By normalizing the unit of each dimension appropriately, consider the cubical space where \( s \) and \( e \) are on vertical edges located diagonally to each other (see Figure 1). Assume that a path between \( s \) and \( e \) is discretized with reasonable granularity as a sequence of adjacent cells in an \( n \times n \times n \) grid corresponding to the 3D space. Without loss of generality, the coordinate system can be defined so that \( s \) and \( e \) are located at \((0,0,a)\) and \((n-1,n-1,b)\), respectively.

![Figure 1: Motion planning problem of an AUV.](image)

Note that this discretization is applied to only the representation of a path. Map data may be represented in any way as long as we can efficiently access information of a given grid cell such as whether there is an obstacle at the grid cell. Thus, there is no restriction on shapes of obstacles.

We consider two types of obstacles: Solid obstacles and hazardous obstacles. A path cannot intersect a solid obstacle while it may intersect a hazardous obstacle at the expense of extra path length in proportion to the hazardous obstacle’s weight which represents various cost factors.

The distance \( d(i,j) \) from cell \( i \) to its adjacent cell \( j \) in the 3D grid is the Euclidean distance from the center \( \rho_i \) of cell \( i \) to the center \( \rho_j \) of cell \( j \). The length \( \ell(i,j) \) from \( i \) to \( j \) is defined as \( d(i,j) \times (1 + w(\rho_i, \rho_j)) \), where \( w \) denotes the average weight between the locations \( \rho_i \) and \( \rho_j \). The length of a path between \( s \) and \( e \) is the sum of lengths between two consecutively adjacent cells on the path.

We first define a problem of path planning in 3D space and then generalize it to motion planning (i.e., both path planning and trajectory planning) in 3D space.

3D Path Planning Problem

**Input:** \( n \times n \times n \) grid, start cell \((0,0,a)\), end cell \((n-1,n-1,b)\), obstacles, and their weights
Figure 2: An example of the 3D path planning problem.

**Output:** A path between cells \((0, 0, a)\) and \((n - 1, n - 1, b)\) such that the length of the path is minimum, subject to
(a) the path does not intersect any solid obstacle and
(b) the path meets limitations on the maneuverability of an AUV \(^1\)
Figure 2 shows an example of output to the 3D path planning problem.

To deal with path planning and trajectory planning simultaneously, another dimension \(t\) for time is added. In this context, a time-varying environment such as time-dependent obstacles due to changes of the tide and current is incorporated.

**3D Motion Planning Problem**

**Input:** \(n \times n \times n \times \infty\) grid, location \(a\) of the start cell, location \(b\) of the end cell, obstacles, and their weights

**Output:** A path between cells \((0, 0, a, 0)\) and \((n - 1, n - 1, b, T)\) for some integer \(T\) such that the length of the corresponding path in 3D space is minimum, subject to
(a) the path does not intersect any solid obstacle and
(b) the path meets all limitations on the maneuverability of an AUV (including the maximum velocity and the maximum acceleration of the AUV)

Note that there may be other optimization criteria of interest. For example, the completion time \(T\) or the total energy consumption is minimized.

**GA for 3D Path Planning**

A genetic algorithm (GA) \([1, 5]\) for an optimization problem consists of two major components (see Figure 3). First, GA maintains a population of individuals, where each

\(^1\)For example, the maneuverability limitations may include the minimum turning radius.
individual corresponds to a potential solution and the population is a collection of such potential solutions. In GA, an individual is commonly represented by a binary string. The mapping between solutions and binary strings is called a coding. The number of individuals in a population is called the population size. Second, GA repeatedly transforms the population by using a mechanism analogous to biological evolution. The mechanism includes the following steps.

1. **Fitness Evaluation:** The fitness corresponding to an optimization criterion (i.e., objective function) is calculated for each individual.

2. **Selection:** Some individuals are chosen from the current population as parents, based on their fitness values.

3. **Recombination:** New individuals (called offspring) are produced from the parents by applying genetic operators such as crossover and mutation.

4. **Replacement:** Some individuals (not necessarily parents in general) are replaced by some offspring.

The population produced at each transformation is called a generation. By giving highly fit individuals more opportunities to reproduce, the population becomes likely to include “good” individuals throughout generations.

There are 4 major components to be designed in a GA: (1) coding, (2) fitness function, (3) configuration of genetic operators, and (4) parameters of the genetic operators. Among them, the coding is most crucial in success of the GA and requires a designer’s inspiration. The fitness function is usually derived from the problem formulation. The others (3) and (4) are commonly decided through experiments.
Figure 4: Coding of a path in 3D space.

We give some notations and definitions to describe a coding. A path in 2D space is said to be monotone with respect to $x$-coordinate ($x$-monotone for short) if no lines parallel to $y$-axis “cross” the path at two distinct points. Similarly, $y$-monotone is defined. A path in 3D space is said to be $xy$-monotone if no lines parallel to $z$-axis cross the path at two distinct points. Similarly, $xz$-monotone and $yz$-monotone are defined. A projection of a path in 3D space on $x$-$y$ plane is called $xy$-projection of the path. Similarly, $xz$-projection and $yz$-projection are defined.

The coding used in our GA decomposes a path in 3D space into three projections of the path, namely, $xy$-projection, $xz$-projection and $yz$-projection (see Figure 4). Obviously, there exists at least one triple of such 3 projections which represent an arbitrarily given path in 3D space. However, it is not always true that an arbitrarily given triple of projections represents a unique path in 3D space. To guarantee the uniqueness, we assume the following.

**Assumption 1:** A path in 3D space is $xy$-monotone.

**Assumption 2:** The $xy$-projection of the path in 3D space is $x$-monotone and $y$-monotone.

Then, each projection (i.e., a path in 2D space) is represented by a binary string as described below. Finally, the resulting 3 binary strings are interleaved bit by bit. The reason for interleaving is that crossover can transform all the projections of a path at the same time.

Since $xy$-projection is $x$-monotone (and also $y$-monotone), it can be represented by a row-wise (or column-wise) sequence of $n - 1$ pairs of direction and distance such that each pair specifies a segment of the projection between two consecutive rows (or columns). Thus, $xy$-projection is coded into a binary string as follows. The first bit indicates whether $xy$-projection is represented in the row-wise manner or not. The remaining bits of the binary string are grouped into $n - 1$ blocks of $3 + \lceil \log_2 n \rceil$ bits each. The first 2 bits of each block indicate the direction of $xy$-projection: 00 for $\uparrow$, 01 for $\nearrow$, 10 for $\rightarrow$, and 11 for $\downarrow$. In case
of 00 (i.e., the direction ↓), the other $1 + \lceil \lg n \rceil$ bits of the block denote the distance as a signed integer in the range $[-(n - 1), +(n - 1)]$. Otherwise, the other bits are ignored.

Since Assumption 2 implies that $xz$-projection is $x$-monotone and $yz$-projection is $y$-monotone, these projections can also be represented in the same way as for $xy$-projection, except that they do not need the first bit. Therefore, the length of the whole binary string is fixed at $1 + 3(n - 1)(3 + \lceil \lg n \rceil)$.

This coding has two advantages. First, binary strings are of the same fixed length. As previously mentioned, if strings are of variable length, a GA must be designed in an ad hoc manner to guarantee generation of strings of an arbitrary length from any initial population. In addition, derivation of an appropriate length takes some time and makes evolution slower. Second, applications of genetic operators to fixed-length strings always produce syntactically valid strings. This simplifies a process of reproduction and enables the standard GA to work well.

Although we observed in simulation that a similar coding for 2D path planning usually works well in 2D space [12], there is a potential problem in the above coding which we also noticed through the simulation. Since the entire path is represented by a sequence of relative directions and distances, even a small change at the beginning of a binary string may drastically affect the entire path and its fitness while changes of a segment closer to the end of the binary string has only minor effects on them. Thus, if there are obstacles restricting the areas that optimal paths go through, it is possible that the speed of evolution in a GA slows down once the almost entire path except its beginning segment becomes identical to an optimal path.

A possible solution to the above potential problem is to modify the coding as follows. For each row (or column) in a 2D grid, a block of a binary string is composed of two binary integers $j$ and $d$ such that $0 \leq j \leq n - 1$ and $0 \leq d \leq n - 2$ (see Figure 5). The first integer $j$ denotes an absolute position on the $i$-th row. The second integer $d$ denotes how absolute positions on the $i$-th and $(i + 1)$-th rows are connected. If $d < |k - j|$, a path passes $d$ cells on the $i$-th row from cell $(i, j)$ toward cell $(i + 1, k)$, transits diagonally to a cell on the $(i + 1)$-th row, and then goes to cell $(i + 1, k)$. Otherwise, $d$ is regarded as $d \mod |k - j|$. The length of each block of a binary string is $\lceil \lg n \rceil + \lceil \lg(n - 1) \rceil$. Thus, this coding also produces a binary string of fixed length.

Figure 5: An alternative coding.
We are currently investigating whether the potential problem of slow evolution actually degrades the performance of our GA and, if so, whether the modified coding solves the problem.

3D Motion Planning

To solve the 3D motion planning problem (i.e., path planning and trajectory planning at once), we need to extend the coding given above so that a path in a 4D grid is coded into a binary string. As mentioned in Introduction, it is desirable to use fixed-length strings.

A solution to the 3D motion planning problem is regarded as a pair of a path in 3D space and a trajectory on the path. The trajectory on the path can be represented by either a function \( f(t) \) of time \( t \) or a function \( g(\rho) \) of location \( \rho \) on the path, where a value of \( f(t) \) is a location on the path and a value of \( g(\rho) \) is the time at which an AUV passes \( \rho \).

A possible coding is to separate a representation of trajectory from a path in a 3D grid corresponding to the underlying 3D space. The path in the 3D grid is represented as \( \sigma_p \) by the same coding described above under Assumptions 1 and 2. Note that \( \sigma_p \) consists of 3 projections of the 3D path which are monotone. Then, a trajectory on the path is approximated by 3 trajectories on the 3 projections as follows. Suppose that \( \pi \) is a projection which is \( x \)-monotone. A trajectory on \( \pi \) is represented as a sequence of time intervals \( t_i \) (\( 0 \leq i \leq n - 1 \)) each of which denotes how long an AUV stays on the same \( x \)-coordinate value \( i \). Thus, a binary string \( \sigma_t \) of the entire trajectory is obtained by interleaving strings for the trajectories on the 3 projections. Finally, an individual is obtained by interleaving \( \sigma_p \) and \( \sigma_t \).

There are two issues to be investigated further in this coding.

1. Potential inconsistency among strings for the 3 trajectories: If time intervals on the 3 projections are given arbitrarily, it is possible that the time intervals on three coordinates \( x, y \) and \( z \) may conflict each other. Since the performance of a GA may be degraded if it produces many invalid strings, we need a method for interpreting \( \sigma_t \) which resolves the inconsistency. The simplest way is to assign priorities to 3 trajectories and resolve the inconsistency in the order of their priorities.

2. Mapping between the approximated trajectory and a lower-level motion control: Since a trajectory plan may not be fine-grained enough to represent actual motion of an AUV explicitly, we need some mechanism for translating the trajectory plan into lower-level motion control. For example, according to a given sequence of time intervals \( t_i \), we need to decide feasible, smooth control of velocity and acceleration.

Summary

This paper proposed a GA for motion planning of underwater robotic vehicles. We first presented a GA for path planning in 3D space with solid obstacles and/or hazardous obstacles which a path can intersect at the expense of extra costs depending on their weights. Advantages of our GA approach are (1) adaptivity which enables us to incorporate on-line
motion planning with off-line motion planning and (2) integration of path planning and trajectory planning.

Acknowledgements

This publication is funded in part by a grant/cooperative agreement from the National Oceanic and Atmospheric Administration, project # R/ES-4 which is sponsored by the University of Hawaii Sea Grant College Program, SOEST, under Institutional Grant No. NA36RG0507 from NOAA Office of Sea Grant, U. S. Department of Commerce. The views expressed herein are those of the authors and do not necessarily reflect the views of NOAA or any of its sub-agencies. UNIHI-SEAGRANT-CP-98-05.

This is also supported in part by a grant (N00014-97-1-0961) from the Office of Naval Research.

References


