VORONOI STRAINS - A SPLINE PATH PLANNING ALGORITHM FOR COMPLEX ENVIRONMENTS

Martin Saska
Informatics VII: Robotics and Telematics
Julius-Maximilians University Wuerzburg
Wuerzburg, Germany
email: saska@informatik.uni-wuerzburg.de

Martin Hess and Klaus Schilling
Informatics VII: Robotics and Telematics
Julius-Maximilians University Wuerzburg
Wuerzburg, Germany
email: {m.hess,schi}@informatik.uni-wuerzburg.de

ABSTRACT
Path planning and obstacle avoidance methods are often required for robots working in more and more complicated environments. This paper introduces a novel approach called Voronoi Strains for solving this task. The algorithm applies particle swarm optimization of cubic splines which are connected to strings. The initialization of the evolutionary algorithm is based on the Voronoi graph method as well as on strains of bacteria. Strains whose evolution is stuck in a dead end because of a local optimum die off. Only the strain located close to the global optimum or the biggest local optimum will survive the evolutionary process.

Different settings of PSO parameters have been tested in various simulation experiments. The Voronoi Strains approach was also compared with other PSO methods using a different kind of initialization.

KEY WORDS
Voronoi graph, path planning, Particle swarm optimization, spline functions, mobile robotics

1 Introduction

Path planning and obstacle avoidance are very important in most robot controlling systems. The goal of these tasks is to find a trajectory from the actual position S of the robot to a desired goal position G with respect to position and shape of obstacles. The path can be optimized regarding length, shape, executing time, power consumption, distance to the obstacles or any other requirement.

Many different algorithms have been mentioned in literature [1], [2], [3], [4], but most of them only provide a set of points between the positions S and C or the trajectory is assembled by simple components (usually lines and parts of circles).

The algorithm presented in this paper results in a trajectory composed from cubic splines. These splines are advantageous for continual motion of the robot. They have continuous first and second derivatives. It is also easy to guarantee a smooth connection of the splines composed in a chain and to set their initial conditions (the points S, C and the headings of the robot in these points).

Optimization of the spline path using conventional methods is too slow and therefore unusable in realtime applications. As a result many authors use standard path planning graph methods (e.g. visibility graph [5] or occupancy grid [6]) where the edges of the cheapest path are smoothed using splines [7]. In most cases this spline trajectory is suboptimal, because the optimization process only considers a predefined part of the robot's workspace.

Another possibility to optimize the trajectory is to use a progressive approach, which improves the solution step by step. The optimization process can be stopped every time, which is required by realtime applications, but the quality of the solution cannot be guaranteed. Evolutionary approaches (mainly genetic algorithm (GA) [8]) are frequently used in mobile robot applications. In most of these approaches the path is simply represented by a set of waypoints [9], [10].

In the Vornoi Strains algorithm (VS) a different evolutionary method called Particle Swarm Optimization (PSO) is used. PSO is a relatively new approach and it usually provides better results than GA in situations with a high amount of local extremes. The same solution in such case can be obtained with smaller number of evolutions by using PSO. Voronoi Graphs (VG) are used to initialize the PSO algorithm, in order to reduce further computational time.

PSO [11] is an optimization algorithm inspired by the behavior of flocking birds. Each solution of the searched nonlinear function (called particle) is represented by a set of parameters. A group of these particles (called swarm) has the same meaning as a population in genetic algorithms.

It is supposed that the best position $\overrightarrow{p_i}$ of each particle i, the best solution $\overrightarrow{p_g}$ achieved by the population, actual positions $\overrightarrow{x_i}(t)$ and velocities $\overrightarrow{v_i}(t)$ of the particles are known during the whole optimization process. Positions and Velocities are updated in each iteration by the equations

$$\begin{aligned} v_i(t) = w v_i(t-1) \\ + \Phi_1\big(p_i - x_i(t-1)\big) \\ + \Phi_2\big(p_g - x_i(t-1)\big), \end{aligned} \tag{1}$$

$$x_i(t) = x_i(t-1) + v_i(t), \tag{2}$$

where w is an inertia weight, which decreases linearly from w_{start} to w_{end} during the entire iteration process. Φ_1 and Φ_2 are diagonal matrices weighted with a factor φ_i, which

consists of random numbers drawn from a uniform distribution between 0 and 1. The velocities $v_i(t)$ are limited to values of the interval $<-V_{max}; +V_{max}>$.

The Voronoi Diagram [12], which is used for the initialization of the PSO in the Voronoi Strains algorithm, divides the plane in spheres of influence $V(p_i)$ consisting the points P.

$$V(p_i) = \{y : |y - p_i| < |y - p_j|\}, \tag{3}$$

where $p_j \in \{P - p_i\}$ and points P are the centers of the obstacles. Therefore the voronoi graph is situated as far as possible from the obstacles. The shortest path in such a graph is still needlessly long and also not smooth. But it is a good estimation of the region, where the optimal trajectory could be situated.

The rest of the paper is organized as follows. The novel method called Voronoi Strains is described in section 2. The results presented in section 3 consist of two parts. The best settings of the PSO parameters and their influence on the optimization process are described in subsection 3.1. After this the VS algorithm is compared with simple PSO planning methods in subsection 3.2. Section 4 consists of the conclusion and suggestions for future work.

2 Algorithm description

2.1 Particle description

The path planning problem for a carlike mobile robot can be described by a search in the space of functions. We reduce this space to a subspace which only contains strings of cubic splines [13]. Splines are natural for robot movements. They are easy to implement and could be smoothly connected together. The mathematical notation of a spline in 2D space could be:

$$\begin{aligned} f(t) &= a_x t^3 - b_x t^2 + c_x t + d_x \\ g(t) &= a_y t^3 - b_y t^2 + c_y t + d_y, \end{aligned} \tag{4}$$

where a, b, c, d are constants defined by

$$\begin{aligned} a &= 2P_0 - 2P_1 + P_0' + P_1' \\ b &= -3P_0 + 3P_1 - 2P_0' - P_1' \\ c &= P_0' \\ d &= P_0, \end{aligned} \tag{5}$$

where $t \in < 0, 1 >$. P_i, P_i' are vectors that define the spline in the space. According to these equations each spline is only defined by the points P_0 (starting point), P_1 (final point) and the tangent vectors P_0', P_1'. To guarantee continuity of the whole path, every two neighboring splines in the string share one of the terminal points and it's corresponding tangent vector. Starting point, final point and the corresponding tangent vectors of the string of splines are defined by initial conditions. The total number of variables that define the whole trajectory in 2D is therefore only $4(n-1)$, where n denotes the number of splines in the string. The structure of the particles used for optimization is shown in Figure 1.

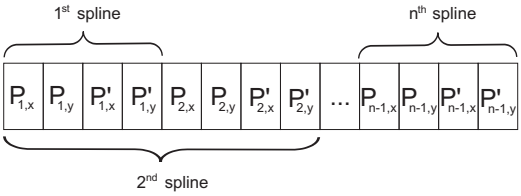

Figure 1. Each particle consists of a vector of spline parameters.

2.2 Initialization

Normally evolutionary algorithms are based on a randomly generated initial population that should uniformly cover the whole search space. In our case, the position of an optimal solution is approximately known. That is why we generate the population close to this area. As a result the algorithm converges faster but the ability to escape local minima is reduced, because the population diversity is lost. This causes problems when the estimation of the solution is not correct.

The basic idea of the Voronoi Strains approach is to suppose that the optimal spline trajectory probably is situated close to the shortest path in the Voronoi Graph. The optimal trajectory is not only the shortest spline path from the actual to the desired position. It is also the minimum of the fitness function (8). Likewise edges of the VG are evaluated by

$$f(e) = l(e) + \left(\frac{\alpha}{\min_{o \in O} d(e, o)} \right)^2, e \in E \tag{6}$$

where α has the same value as in the fitness function, $l(e)$ is the length of edge e and $d(e, o)$ is the shortest distance between edge e and the obstacle o.

The "cheapest" path from node S to node G in the evaluated graph is the area where the optimal spline trajectory is situated with the highest probability. The optimization process can fail in situations that show many local extremes in this region. Then the PSO approach cannot escape and therefore cannot reach the optimal solution because of a low population diversity.

This problem is solved by creating new strains in the population. The price of the edges of the cheapest path always is risen two times and the next strain is located along the new cheapest path in the graph. The new and the old paths can contain the same edges, but these "bridges" are usually short.

The number of particles in each strain is defined by

$$s(i) = s_{total} \frac{2^{m-i}}{2^m - 1} \tag{7}$$

where s_{total} is the total number of particles in the population and m is the number of strains.

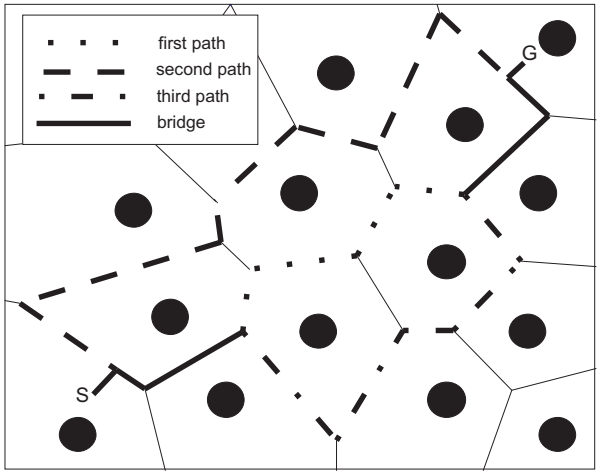

Figure 2. Voronoi diagram and the cheapest paths used for the strain initialization.

During the initialization of the PSO algorithm the Voronoi paths are divided into $n-1$ parts of identical length (n is the number of splines in the string). The control points P_i of the splines are randomly generated on each part in consideration of their geometric representation. The vectors P_i' are initialized in the direction of $\overrightarrow{P_{i-1}P_i} + \overrightarrow{P_iP_{i+1}}$. A basic example of the cheapest paths in a Voronoi graph is shown in Figure 2.

2.3 Particle evaluation

The rating (fitness) of a particle is calculated by the fitness function f. The global minimum of this function corresponds to a smooth and short trajectory. Therefore the optimization function f must strictly penalize trajectories that possibly cause a collision with an obstacle. Other constraints (e.g. smooth derivation) are guaranteed by the particle construction (see section 2.1) and need not to be included in the fitness function f.

In this paper f is defined as

$$f = f_{length} + \alpha f_{distance}, \tag{8}$$

where α determines the influence of obstacles. The distance between the calculated trajectory and the obstacles growths with a higher value of α, but such a path can be unnecessarily long. Conversely the trajectory can be too close to obstacles if the value of α is set too low.

The variable f_{length} grows with the trajectory length and is computed by

$$f_{length} = \frac{\int_0^1 \sqrt{\left(f'(t)\right)^2 + \left(g'(t)\right)^2}dt}{l_{MIN}} \tag{9}$$

where l_{MIN} is the Euclidean distance between actual and desired position of the robot.

Table 1. Mean values of the fitness function of the best particle after 50 iterations for different values of V_{max} and w_{start}.

	V_{max}				
w_{start}	10	70	150	220	300
0.1	14.51	11.10	11.21	11.65	11.86
0.2	14.34	11.03	11.18	11.52	11.57
0.4	13.43	10.98	11.24	11.43	11.65
0.6	12.28	11.00	11.43	11.80	12.00
0.8	11.82	11.23	11.82	12.06	12.39

The second part of the fitness function $f_{distance}$ is inversely proportional to the minimal distance to the obstacles. This function "pushes" the path into free space and can be described by

$$f_{distance} = \left(\min_{o\in O}\left(\min_{t\in<0;1>}(d(t))\right)\right)^{-2}, \tag{10}$$

where O is the set of all obstacles in the robots configuration space and

$$d(t) = \sqrt{\left(f(t) - o_x\right)^2 + \left(g(t) - o_y\right)^2}. \tag{11}$$

3 Results

The Voronoi Strains algorithm was intensively verified in two experiments. Both are based on a statistically processed set of runs, because each run of the evolutionary approaches is unique and thus the final trajectories are different. Each value presented in the following subsections was obtained from 400 runs of the algorithm.

The first experiment was made to study the influence of different PSO algorithm parameters on the quality of the resulting trajectories. In the second one we compared the Voronoi Strains algorithm with PSO approaches with a different initialization part.

3.1 PSO parameter settings

All results have been obtained by the VS algorithm using the following constants: $w_{end} = 0.05$, $\varphi_1 = 2$ and $\varphi_2 = 2$. The resulting trajectories were composed from 9 splines ($dim(x_i) = 36$) and the population consisted of 28 particles (4 particles in the 1th strain, 8 in the 2nd strain and 16 in the 3rd strain). Each optimization process was interrupted after 50 iterations. 150 circular obstacles were distributed randomly in the workspace.

The mean fitness values of the experiments with different settings of w_{start} and V_{max} (equations (1) and (2)) are presented in table 1. The best results were obtained with $w_{start} = 0.4$ and $V_{max} = 70$. It is interesting to compare these with the simple PSO planning method using random initialization [14], where the optimal values are $w_{start} = 0.6$ and $V_{max} = 250$. The most significant is

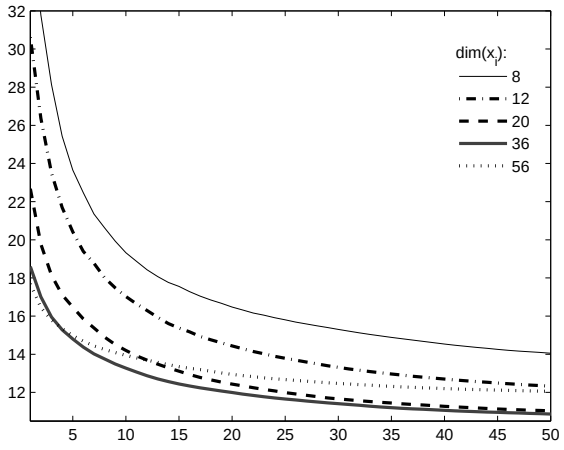

Figure 3. Progression of the best particle with different settings for $dim(x_i)$.

the difference of V_{max}. Particles in the VS algorithm are initialized close to the optimal solution and they are kept in this position by the low value for the maximal velocity. If V_{max} is too big, the particles start to explore the whole space and the advantage of the initialization is lost. Contrariwise the PSO algorithm with a too low value of V_{max} can converge only slowly and the optimal trajectory cannot be found during 50 iterations.

A low value of w_{start} is advantageous to the VS approach, because the exploratory process is skipped (due to better initialization) and the optimization starts directly in exploitative mode. During the exploratory mode of the PSO the whole workspace is explored and w_{start} has a big value. Exploitative mode commonly follows where the value of w_{start} is reduced and the space is searched only close to the most promising solution.

The progression of the mean fitness value of the best particle is presented in figure 3 for different numbers of splines used to evolute the trajectory. The lowest mean value of the best particle after 50 iterations was achieved with $dim(x_i) = 36$ (10 splines in the string). A higher dimension results in a better initial population (higher mean value of the best particle), but also slows down the optimization process. Contrariwise the PSO is sped up if the value of $dim(x_i)$ is low, but the space is reduced and the optimal solution can be of poor quality (e.q. not free of collisions).

A solution of the situation with 150 randomly generated obstacles is presented in figure 4. The optimal trajectory (thick line) was found by the PSO algorithm with optimal values for the parameters ($dim(x_i) = 36$, $w_{start} = 0.4$ and $V_{max} = 70$) after 50 iterations. Figure 5 shows one of the places, where the randomly generated initial trajectory (thin line) is situated too close to an obstacle. After the optimization the path is safe.

Table 2. Mean fitness values and covariances of the fitness function of the best particle after 50 iterations.

method	150 obstacles	10 obstacles
PSO-simple	56.56 (444.53)	1.68 (0.0047)
PSO-line	21.31 (44.57)	1.67 (0.0039)
Voronoi Strains	10.98 (0.89)	1.67 (0.0035)

3.2 Comparison of PSO methods with different initialization approaches

In the experiments described in this section three different algorithms were compared in two dissimilar scenarios. The first algorithm (PSO-simple) only uses basic initialization. Particles in the initial population are generated randomly and so the control points of the splines cover the complete workspace and the trajectories possibly contain loops. In the second approach (PSO-line) it is supposed that the optimal trajectory is short and without loops or back motion. Therefore the line connecting the points S and G is divided into n same parts (similar as in section 2.2) and the control points P_i are randomly generated close to the dividing points. The Vectors P_i' are initialized in direction $\overrightarrow{P_{i-1}P_i} + \overrightarrow{P_iP_{i+1}}$.

The first scenario is identical to the situation with 150 obstacles used in section 3.1 and demonstrates the ability of the VS algorithm to work in complicated environments. The mean fitness value and the covariance of the fitness function of the best particle after 50 iterations is presented in table 2. The mean solution achieved by the VS approach is much more optimized than the trajectories found by other methods, but the most significant fact is found by comparing the covariances. The low covariance in the VS approach shows it's high robustness in comparison to the PSO-simple and the PSO-line method.

The second scenario consists of only 10 obstacles in a workspace of identical size to the one in the first experiment. The results presented in the second column of table 2 illustrate a low effect of the initialization on the optimization process in such a simple situation. The biggest advantage of the VS approach (missing exploratory mode) is lost, because in this case the exploratory mode in the simple PSO approach has a very short duration.

4 Conclusion and future Work

In this paper we presented a novel path planning approach that could be useful in complicated environments. The Voronoi Strains algorithm was tested in two kinds of situations with a different number of obstacles. It was also compared to similar PSO methods.

The algorithms achieved interesting results in workspaces with a high number of obstacles. In these complicated environments it found much better solutions than the PSO-line and the PSO-simple algorithm with the same

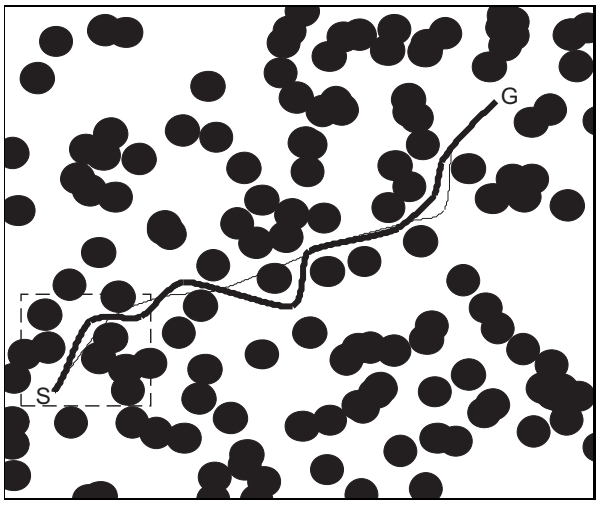

Figure 4. The final path (thick line) found by VS after 50 iterations and the best path in the initial population (thin line).

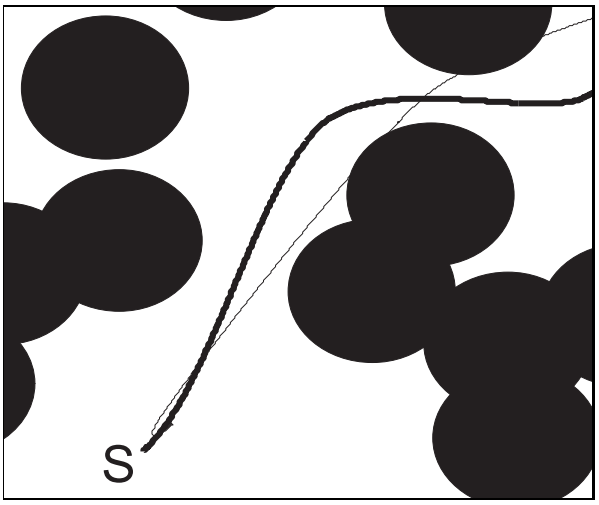

Figure 5. Zoomed part of the starting area of Figure 4.

amount of iterations. Additionally the VS approach proved to be much more robust and it's dependence on the initial random population was lower.

The VS algorithm is based on the initial population that is placed around the cheapest paths in the Voronoi graph. The evolutionary process is therefore much faster and in addition the population is protected against the loss of diversity. The approach also has a bigger probability to avoid local minima. The Experiments nevertheless demonstrated that the optimization process is not sped up by the VS approach, if the environment is not so complicated.

In the future we would like to adapt the algorithm to be utilizable in dynamical environments. Our idea is to use the old population and continue the optimization with a partly changed fitness function. Such an approach must ensure to keep the diversity of the population which is necessary for the PSO algorithm to work properly.

Another possible way to go might be to use the algorithms with other types of initial populations (maximum entropy based, systematic coverage, Visibility Graph based, etc.) for further comparisons. The PSO algorithm could also be compared to other evolutionary methods (Genetic algorithms, simulated annealing, ant colony, etc.).

REFERENCES

[1] J.C. Latombe. *Robot motion planning.* fourth ed. Kluver Academic Publishers. Fourth edition, 1996.

[2] J. Borenstein and Y. Koren. The vector field histogram: Fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation* 7(3), 278-288,1991.

[3] H. Meng and P.D. Picton. A neural network for collision-free path planning. *Artificial Neural Networks* 2(1), 591-4, 1992.

[4] W.G. Han, S.M. Baek and T.Y. Kuc. Genetic algorithm based path planning and dynamic obstacle avoidance of mobile robots. *Proc. IEEE International Conference on Systems, Man, and Cybernetics.* 3, 2747-51, 1997.

[5] R. Kunigahalli and J.S. Russell. Visibility graph approach to detailed path planning in cnc concrete placement. *Automation and Robotics in Construction XI.* 1994.

[6] T. Braun and N. Tay. Combining configuration space and occupancy grid for robot navigation. *Industrial Robot.* Vol. 28(3), 2001.

[7] M. Lepetic, G. Klancar, I. Skrjanc, D. Matko and B. Potocnik. Time optimal planning considering acceleration limits. *Robotics and Autonomous Systems.* Volume 45, 199-210, 2003.

[8] vJ. Holland. *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press. 1975.

[9] J. Tu, S.X. Yang, M. Elbs and S. Hampel. Genetic algorithm based path planning for a mobile robot. *Proc. of the 2003 IEEE Intern. Conference on Robotics and Automation.* pp. 12211226. 2003.

[10] A. Elshamli, H.A. Abdullah and S. Areibi. Genetic algorithm for dynamic path planning. *Proc. Canadian Conference on Electrical and Computer Engineering.* IEEE - Piscataway, NJ, USA. pp. 677-80, 2004.

[11] J. Kennedy and R.C. Eberhart. Particle swarm optimization. *Proc. International Conference on Neural Networks IEEE.* Vol. 4. pp. 1942-1948, 1995.

[12] F. Aurenhammer and R. Klein. *Voronoi diagrams. Hand book of Computational Geometry.* Elsevier Science Publishers. Amsterodam, 2000.

[13] J. Ye and R. Qu. Fairing of parametric cubic splines. *Mathematical and Computer Modelling.* Volume 30, 121-31. 1999.

[14] M. Saska, M. Macas, L. Preucil and L. Lhotska. Robot path planning using partical swarm optimization of ferguson splines. *Proc. 11th IEEE International Conference on Emerging Technologies and Factory Automation.* 2006.