Learning the Voronoi Centers of a Mobile Robot's Configuration Space

N. A. Vlassis G. Papakonstantinou
P. Tsanakas

Digital Systems and Computers Laboratory
National Technical University of Athens
15773 Zografou Campus, Athens, Greece
E-mail: nvlassis@dsclab.ece.ntua.gr

Abstract

Partitioning the configuration space of a mobile robot is essential for the robot path planning task. However, most existing techniques either rely upon precise geometrical descriptions of the environment, or are static by nature. In this paper we propose a method by which the robot dynamically builds a Voronoi tessellation of its configuration space. In order to do this, we apply an adaptive k-means clustering algorithm to the robot's free space, while letting the robot explore its environment. The cluster centers, corresponding to the centers of the respective Voronoi cells, are connected into a mesh by using the Delaunay triangulation. Then, we show how the basic robot tasks can be integrated on the resulted graph.

1 Introduction

Exploring the free space and building a good map representation is a challenging mobile robot problem [1, 2, 3]. In this paper we propose a method by which the robot explores and learns its environment without any prior information about it. The notion of 'learning' implies a way of modeling the free space so that subsequent tasks, like path planning or obstacle avoidance, can easily be carried out based on the previously gathered information.

To accomplish this task we develop an adaptive clustering technique [4] that partitions the robot's free configuration space in a number of regions, the clusters, each one characterized by its *means* and its *variance*. Under certain assumptions, the cluster centers can be regarded as the centers of a *Voronoi diagram* [5] of the robot's free space.

To exploit this clustering, a way for connecting the cluster centers in pairs must be found. For that, we triangulate the set of clusters centers by using the *Delaunay triangulation* [6], to produce a mesh on which subsequent tasks can easily be carried out. This mesh is a non-directed graph on which path planning can be performed by the usual graph searching techniques [7]. In our case, we apply a modified $A^\star$ algorithm on the graph, like in [8]. We demonstrate some of the results of our method running the Khepera simulator [9].

2 Clustering the Robot's Free Configuration Space

Let us assume a mobile robot's configuration space be a subset of $\mathbb{R}^2 \times [0, 2\pi)$, which corresponds to all different positions (x, y) and orientations θ of the robot in its environment [10]. We make two assumptions; first, we restrict ourselves to a *holonomic* mobile robot, i.e., a disc-shaped robot whose degree of freedom of turning around its central axis is independent of its current position in the plane. In that case, the robot's total configuration space is $\mathcal{C} \subset \mathbb{R}^2$, and the free space, composed of all obstacle-free configurations of the robot, is $\mathcal{C}_{free} \subset \mathcal{C}$. Second, we assume a correct positioning system whereby the current robot's position $(x, y) \in \mathcal{C}_{free}$ is continuously adjusted by an appropriate localization procedure, and thus is considered accurate.

Then, we let the robot explore its environment. The contiguous robot movements provide us with a sequence of n 2-dimensional samples $\{\xi_1, \dots, \xi_n\}$ from $\mathcal{C}_{free}$, with n unrestricted at the moment. Our task is to group all ξ_i into K clusters, so that each cluster contains neighboring points in $\mathcal{C}_{free}$, as measured by some metric distance which may be the Euclidean or may be not. Since we don't have any prior knowledge about the robot's environment, we want K to change dynamically.

To this direction, we employ a general statistical clustering schema [11] which assumes that samples from cluster k follow a 2-*variate normal* probability density function $\mathbf{N}_2\{\mu_k, \sigma_k\}$, i.e.,

$$p_k(\xi_i) = \frac{1}{2\pi\sigma_k^2} e^{-\frac{1}{2\sigma_k^2}[(\xi_i - \mu_k)(\xi_i - \mu_k)^T]}, \qquad (1)$$

parametrized over the *means* μ_k and *variance* σ_k^2 of the cluster. In this model, a cluster can be regarded as a circle centered at μ_k, while σ_k^2 gives a measure of its size.[1] Also we assume that each cluster k has a *prior* probability π_k to be preferred over the other clusters.

The *Bayes decision rule* [11] (p. 19) assigns a future sample ξ_i to cluster k if the *posterior* probability

$$p(k|\xi_i) = \frac{\pi_k p_k(\xi_i)}{\sum_{j=1}^{K} \pi_j p_j(\xi_i)} \qquad (2)$$

that ξ_i belongs to cluster k is maximized over all other clusters. Under the above framework, the problem is translated into estimating the parameters μ_k, σ_k^2 and the prior probability π_k of each cluster, based on the sequence of input samples $\{\xi_i\}$, $i = 1, \dots, n$.

[1]Note that this schema does not necessarily imply a Gaussian uncertainty model in the robot's position estimates; it rather eases the statistical computations.

3 The Probabilistic Growing Cell Structures Algorithm

In this section we describe the *probabilistic growing cell structures algorithm* [4], a probabilistic clustering algorithm for arbitrary n-dimensional input domains. In our case, the input domain is the robot's free configuration space $\mathcal{C}_{free} \subset \mathbb{R}^2$.

Initialization. The initial position $\xi_1 = (x_1, y_1)$ of the robot becomes the center (means) μ_1 of the first cluster, whereas σ_1^2 is set to 0. The prior probability π_1 of this cluster is set to 1.

Adaptation. According to the Bayes decision rule, a new input ξ_i is assigned to the cluster k with the maximum $p(k|\xi_i)$, as computed by eq. 2. This cluster is called the *winning* cluster. Ignoring the normalization factor in the denominator of eq. 2 and applying the logarithm to the nominator, this corresponds to minimizing $\delta(\xi_i, \mu_k)^2 + 2\log(2\pi\sigma_k^2) - 2\log\pi_k$, where $\delta(\xi_i, \mu_k) = [\frac{1}{\sigma_k^2}(\xi_i - \mu_k)(\xi_i - \mu_k)^T]^{1/2}$ is the *Mahalanobis* distance from ξ_i to the center of cluster k.

Following [4], we use the optimal under *Maximum Likelihood* estimations of the parameters μ_k and σ_k^2 of the winning cluster k, which for the n-th sample ξ_n read

$$\begin{aligned} \mu_k &= \mu_k + \eta_k(\xi_n - \mu_k), \\ \sigma_k^2 &= \sigma_k^2 + \eta_k[(\xi_n - \mu_k)(\xi_n - \mu_k)^T - \sigma_k^2], \end{aligned}$$

where η_k is the 'learning rate' approximated by $p(k|\xi_n)/(n\pi_k)$, and $p(k|\xi_n)$ is the posterior probability that sample ξ_n belongs to cluster k. It is interesting to note here the similarity of the above formulas to the self-organizing learning schema of Kohonen's SOM algorithm [12].

Updating. The prior probabilities of *all* clusters are updated according to [4]

$$\pi_k = \pi_k + \frac{1}{n}[p(k|\xi_n) - \pi_k].$$

The appearance of n, the number of the input samples appeared so far, in the denominators of the above formulas accounts for a *stochastic approximation*, i.e., a convergence towards a steady state, which in turn implies a static environment. Since we want to incorporate dynamically changing environments into our algorithm, we substitute in the above formulas n with l, a restriction over the l most recent samples $\{\xi_{n-l}, \ldots, \xi_n\}$. This implies a 'forgetting' schema in which old samples affect the learning process less than recent ones.

Cluster insertion. After a fixed number of steps, and if a pre-defined maximum number of clusters is not exceeded, we find the cluster q who maximizes the quantity

$$\phi(q) = \alpha_\pi \pi_q + \alpha_\sigma \frac{\sigma_q^2}{\sum_{j=1}^K \sigma_j^2},$$

with $\alpha_\pi, \alpha_\sigma$ appropriate coefficients that are case-dependent. We create a new cluster r between q and its direct neighbor f with the maximum $\phi(f)$. We estimate the means, variance and prior probability of the new cluster as

$$\mu_r = \frac{\phi(q)\mu_q + \phi(f)\mu_f}{\phi(q) + \phi(f)},$$
$$\sigma_r^2 = \frac{\phi(q)\sigma_q^2 + \phi(f)\sigma_f^2}{\phi(q) + \phi(f)},$$
$$\pi_r = \frac{\phi(q)\pi_q + \phi(f)\pi_f}{\phi(q) + \phi(f)}.$$

Clusters q and f change their variances and prior probabilities appropriately as

$$\sigma_q^{2(new)} = (1 - \frac{\phi(q)}{\phi(q) + \phi(f)})\sigma_q^2,$$
$$\sigma_f^{2(new)} = (1 - \frac{\phi(f)}{\phi(q) + \phi(f)})\sigma_f^2,$$
$$\pi_q^{(new)} = (1 - \frac{\phi(q)}{\phi(q) + \phi(f)})\pi_q,$$
$$\pi_f^{(new)} = (1 - \frac{\phi(f)}{\phi(q) + \phi(f)})\pi_f,$$

where $(\cdot)^{(new)}$ denotes the new value.

Cluster deletion. After a fixed number of steps we remove the cluster c with the lowest $\phi(c)$.

4 Delaunay triangulation

With the above algorithm, a clustering of the robot's free configuration space $\mathcal{C}_{free}$ is feasible. Moreover, the cluster centers can be viewed as the centers of a Voronoi diagram [5] of $\mathcal{C}_{free}$. The Voronoi diagram of a set of points in $\mathbb{R}^2$ is a partition of the plane into polygonal cells, one for each input point, so that the cell of an input point p contains all those loci whose nearest input point according to the Euclidean distance is p.

In order for the clustering to be of practical use, connections must be established between pairs of cluster centers. To this direction, we apply the Delaunay triangulation (DT) [6] to the cluster centers. The DT of a point set is the planar dual of the Voronoi diagram of that set. Each edge of a DT triangle is the perpendicular bisector of a Voronoi cell edge, whereas the triangle vertices correspond to the Voronoi centers. The resulting mesh of triangles forms a non-directed graph, on which path planning can be performed by one of the usual graph searching techniques [7].

5 Results-Discussion

We implemented and tested our method on the Khepera simulator [9]. This software simulates the environment of a disc-shaped holonomic mobile robot that is equipped with eight proximity sensors and a simple sensor and kinematic model. Our exploration strategy was a random walk, whereas, as mentioned above, we assumed accurate knowledge of the robot's (x, y) position at any moment. In Fig. 1 we show the results after applying our method in a typical indoor environment. The left part shows the clustering of the input space, whereas the right part depicts the resulting mesh after the Delaunay triangulation. For the latter we used the well-known Fortune's sweepline algorithm [13] for building the Voronoi diagram first and then taking its planar dual.

In order to perform path planning on the resulting graph we employed a modified $A^\star$ algorithm [8] that takes into consideration the curvature of a candidate trajectory and the prior probability of each cluster to pick up paths that are piecewise smooth, and potentially entail little uncertainty. Given the current's robot position s and

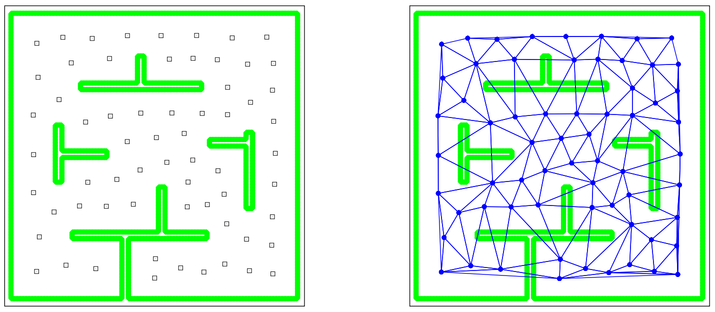
a. Clustering b. Delaunay triangulation

Figure 1. Clustering and triangulating the input space.

a target position t, two local paths are found from s and t to their nearest cluster centers, and then the path planning algorithm selects the best path in the graph that connects these two centers.

As mentioned in section 3, the global update procedure of the clusters' prior probabilities implies a 'forgetting' schema; clusters that have not been 'visited' for a long time are likely to minimize their prior probability to a certain extent and thus be less probable for subsequent plannings. This is analogous to the human planning system, in which recent successful routes are usually preferred over routes that have not been used for a long time.

Finally, as shown in Fig. 1b, the graph that the DT method produces may contain edges that intersect obstacles, even if their respective vertices lie in the free space. To handle these cases we employ a trial-error technique. If a path over an illegal edge is planned, the robot starts following it and when its sensing mechanism perceives a blocking situation the robot stops, appropriately updates the respective map connection (legal-illegal), and replans a path to the goal position.

6 Conclusions

We presented a technique for clustering and triangulating the configuration space of a holonomic mobile robot in order to build a graph appropriate for path planning. We developed a probabilistic clustering schema that is capable of building the Voronoi diagram of the robot's free space, and showed how the Delaunay triangulation method can be applied to the resulting cluster centers to construct a graph.

In the course of our method we made two assumptions: the robot is holonomic and a precise localization procedure is available. Under the latter assumption, we found that a random walk was adequate as an exploration schema. However, the relaxation of the position accuracy assumption should give rise to more elaborate exploration strategies, e.g., like in [14].

We are currently working on incorporating a position uncertainty model to our clustering schema, and on the other hand enhancing the robot with a typical map-based localization procedure [15] to check the validity of our method in more realistic situations.

References

[1] B. J. Kuipers and Y.-T. Byun, *A robust qualitative method for robot spatial learning*, Proc. AAAI (1988) 774–779.

[2] M. J. Matarić, *Integration of representation into goal-driven behavior-based robots*, IEEE Trans. on Robotics and Automation **8**(3) (1992) 304–312.

[3] U. R. Zimmer, C. Fischer and E. von Puttkamer, *Navigation on topologic feature-maps*, Proc. 3rd Int. Conf. on Fuzzy Logic, Neural Nets and Soft Computing, Fuzzy Logic Systems Institute, Iizuka, Japan (1994) 131–132.

[4] N. A. Vlassis, A. Dimopoulos and G. Papakonstantinou, *The probabilistic growing cell structures algorithm*, Proc. 7th Int. Conf.

on Artificial Neural Networks, Lausanne, Switzerland (Oct. 1997).

[5] F. Aurenhammer, *Voronoi diagrams—a survey of a fundamental geometric data structure*, ACM Computer Surveys **23** (1991) 345–405.

[6] M. Bern and D. Eppstein, *Computing in Euclidean Geometry*, World Scientific, 2nd edn. (1995), Du, D.-Z. and Hwang, F. K. (eds.).

[7] N. J. Nilsson, *Principles of Artificial Intelligence*, Springer-Verlag, New York (1980).

[8] N. A. Vlassis, N. M. Sgouros, G. Efthivoulidis, G. Papakonstantinou and P. Tsanakas, *Global path planning for autonomous qualitative navigation*, Proc. 8th IEEE Int. Conf. on Tools with AI, Toulouse, France (Nov. 1996).

[9] O. Michel, *Khepera Simulator version 2.0, User Manual*, University of Nice Sophia-Antipolis (1996).

[10] J. C. Latombe, *Robot Motion Planning*, Cluwer Academic Publishers, Boston, MA (1991).

[11] B. D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, Cambridge, U.K. (1996).

[12] T. Kohonen, *The self-organizing map*, Proceedings of the IEEE **78**(9) (Sep. 1990) 1464–1480.

[13] S. Fortune, *A sweepline algorithm for voronoi diagrams*, Algorithmica **2** (1987) 153–174.

[14] B. R. Donald, J. Jennings and R. Brown, *Constructive recognizability for task-directed robot programming*, Tech. rep., Computer Science Department, The University of Michigan, Ithaca, NY (1991), TR91-1253.

[15] J. Borenstein, B. Everett and L. Feng, *Where am I? sensors and methods for mobile robot positioning*, Tech. rep., The University of Michigan (1996).