

Incorporating Learning in Motion Planning Techniques

Luca Maria Gambardella and Marc Haex

IDSIA, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
Corso Elvezia 36 - CH - 6900 Lugano
Phone: +41 91 22 88 81 Fax: +41 91 22 89 94
Email:luca@idsia.ch marc@idsia.ch

Abstract

Robot motion planning in a cluttered environment requires knowledge about robot shape and size. These robot characteristics influence system performance eventhough most motion planning methods do not consider them. This paper presents an ongoing work that studies general motion planning techniques in combination with knowledge related to robot shape and size. The system acquires knowledge and learns strategies to avoid local collitions and to make global decisions. A neural network is presented that learns local behavior and a learning technique based on a reinforcement method is presented to overcome problems of local minimum.

Introduction

This paper describes a way to combine motion planning algorithms with learning methods in order to improve the performance of a system that drives a robot in a cluttered workspace. One of the needs in this situation is to consider robot shape and size but motion planner methods usually do not use these robot characteristics. Some methods use a C space approach [6]. This method first computes the space of free configurations and reduces the robot to a point in this space for planning. Though this has obvious advantages for the planning phase, the process of computing the free space is difficult and time intensive. Some other systems use the real workspace to represent the robot using general heuristics like potential fields to drive robot motion [5]. The methods proposed in this area are usually based on general purpose planners that suffer from implications of robot shape and size. In this paper we want to show how learning techniques can be used to improve these general planners with specific heuristics depending on the shape and size of the robot.

Moving a robot requires two kind of decisions. At the lowest level the system computes local motions in order to avoid collisions with obstacles. At the highest level the system has to solve problems related to general strategies that allow the robot to go from the starting position to the goal. The role of the high-level planner is to find the path throught workspace positions and the role of the low-level planner is to decide the right actions to follow it. In potential field methods, which we use in this work, this two decision level tightly interact. Step by step the high-level planner decides the direction of the motion and the low-level planner choses the action. If the low level planner is unable to find a satisfying action it returns control to the high level planner which computes a new path using a backtracking technique.

The system we are studying consists of a general planning mechanism based on grid potential fields which performs the low and high-level planning. We have attached to the system two learning mechanisms that improve the power of both the planning phases (See Figure 1).

In this paper we will discuss motion planners that are based on grid potential field models. In this case the low level planning consists of following an artificial field using control points defined on the object. We propose to use a neural network for learning this functionality. The high level planning consists of monitoring the way the object choses motion directions starting a back-tracking process when the object appears to be at a local minimum. We propose a learning technique based on a reinforcement learning strategy that is able to predict dead ends and to chose the most convinient path.

General Planning Strategy

Grid potential field methods are a class of potential field methods for which the potential force is computed using a numerical field generated over the workspace which is represented in a grid [1]. To plan the motion of a certain object a number of control points are defined onto its surface. The potential of the object is computed as the combination of the control point values. The trajectory search consists in finding subsequent collision-free configuration changes that minimize the potential value of the object. Collision checking is done in a fast way because it is performed by simulating the robot configuration in the workspace. During the search, backtracking techniques are used to overcome local minima which can occur because of conflicting attractions of different control points and because of certain characteristics of the object shape.

Though this field can be used for motion planning it will cause objects to move very close to the obstacles and therefore generate many collisions. To avoid this problem the numerical potential field can be generated starting from a voronoi-skeleton derived from the workspace grid. A voronoi-skeleton (see [7] for a detailed discussion) consists of those cells in the grid that have equal distance to two or more obstacles in the grid. In other words the voronoi-skeleton contains the paths that maintain maximum distance from the obstacles. A voronoi-based field is created by first propagating the field value over the skeleton and next from the skeleton to the rest of the free space. The advantage of this kind of field is

twofold. First, because the object follows the skeleton, low-level planning will suffer less from collision avoidance. Second, the nodes of a voronoi skeleton can act as possible backtracking points for the high level planning. The computation of the voronoi skeleton in a grid can be computed in an effective way, see e.g. [10][9].

We have used grid potential field planning successfully for planning the motions of a robotic gripper in order to grasp arbitrary objects[4][3]. One of the interesting benefits of the method is that the planning is only based on camera images of the workspace containing the target object. An analysis of the object contour to find the target contact points is avoided. Instead, appropriate attractive fields are defined on the image and a stable grasp is obtained as the result of combining motion planning for the gripper and force-closure checking.

The overall performance of this method depends largely on the number of local minima and hence on the number and size of the backtracking moves. By enhancing the planner with learning methods it is possible to avoid many of these backtracking steps.

Low Level

The purpose of the low-level planner is to compute a robot collision free position that minimizes the potential fields. The system computes the total potential for all the neighbours configuration of the robot position. For each configuration the total field is a weighted combination of the control points values and the configurations with a total value not greater than the actual robot value are selected for planning.

These configurations are sorted according to their values and are executed in the simulated environment and the first collision free configuration is added to the robot path. If no configurations are found the control is returned back to the high-level planner that performs backtracking strategies.

$$V(q) = \sum \rho_i V(c_i) \tag{1}$$

This computation is executed using the same technique for every robot without considering any robot characteristics. The planner always checks many configurations that are not collision free before finding the right solution. Analysing the system behavior it is clear that the choice of the right free configuration is influenced by the field and largely by robot shape and size.

The purpose of the research is to associate to the planner a learning module able to capture the notion of the robot shape and size. The aim of the low-level learning process is to avoid the computation of all the neighbouring configurations in order to select, in one step, the next right collision free position.

The first need is to find a way to capture the notion of shape and size. Our solution is to place robot control points on the contour of the robot, usually in the corners, instead of distributing them over the robot surface.

We then propose to use a neural network for learning the relation between a specific robot and the way the robot has to be moved in a cluttered environment. The idea is to

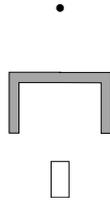


Figure 1: The neural network structure for learning the low level planner capabilities.

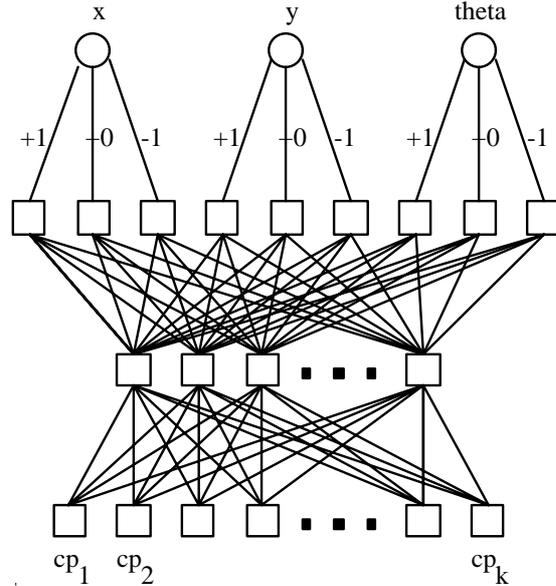


Figure 2: The neural network structure for learning the low level planner capabilities.

start from a general planning module and to build a new system dedicated to the motion of a particular robot.

The neural network is defined considering the input layer as set of nodes that encode the field values of the control points. The output nodes contain a distributed representation of the possible configuration changes for every degree of freedom.

The network can be trained using a set of examples and, after the convergence, it is associated to the planner. If the network fails generating a free configuration, the planner takes control and performs the traditional strategy.

We are currently investigating a way to incrementally improve network performance during the simulation and a mechanism that dynamically select the planner or the network according to system results.

High Level

Grid potential field planning generally applies a best-first technique for searching the solution path. The task of the high level planner is to keep track of this path and in case of a

local minimum (dead-end) to backtrack to a point on this path. A local minimum is reached when the low level planner is not able to chose a configuration change that does not collide which results in a lower total potential. This can occur because of conflicting attractions of the different control points or because there are obstacles blocking the path of the object. The general method to overcome this problem is to backtrack to the previous configuration on the path thereby keeping track of the complete search tree and marking the edges that result in a local minimum. The disadvantage of this method is that many backtracking steps are needed to get out of a deep local minimum. On the other hand this depth-first exhaustive search will always find a solution if one exists. Another solution to local minima is random search, i.e. at a local minimum a number of random paths are generated until a configuration with lower potential value is found. From here again a best-first search is applied.

The learning mechanism that we use to foresee local minima and to propose backtracking points is based on a reinforcement learning strategy. Reinforcement learning is the learning of a mapping from situations to actions so as to maximize a scalar reward or reinforcement signal. (see [8] for a list of articles and [2] for an application in obstacle avoidance). In our model we will learn the relation between a *sensor state* of a robot and the actions that will finally result in a local minimum. The idea is to acquire, during the computation, the relation between a sequence of sensor states and a local minimum situation. This knowledge, which is updated and maintained using a reinforcement learning algorithm, improves the capability of the high level planner.

A *sensor state* S is a n-tuple of parameters (P_i, T, F, R) . P_i are the potential field values of the sensor control points. T is the total potential of the sensor state as computed by equation 1. The boolean value F is the failure value that indicates whether is is possible to continue the search from that state. This value is set to *true* when a collision is detected for the configuration of the robot. For non-colliding states the value can be updated later by the learning algorithm. The parameter R is used by the reinforcement learning algorithm and is related to the failure value.

Sensor states are organized as nodes in a tree for which the arcs between two different sensor states are marked with the action (configuration change) that makes the system change from the parent sensor state to the child sensor state. The set of children nodes are ordered according to the total potential value T and the leaves of the tree are sensor states for which the failure parameter F is *true*.

When a new node S is created, the reinforcement parameter $R(S)$ is set to 0. This parameter is a measure of search failures that have occurred in the state S itself or in one of its children nodes.

When a child of a node S fails the value of $R(S)$ is increased using the reinforcement strategy. If $R(S)$ has reached the threshold value $V(S)$, which is related to the number of children nodes, the failure value $F(S)$ is set to *true* indicating that no more search is allowed from this state, i.e. the planner fails immediately.

In this system a local minimum is described by a tree of sensor states in which at least one state, which is not a leaf in the tree, has the failure value *true*. That means that all

offspring nodes of this state have failed. The size of this tree is a parameter n that can be chosen at the beginning.

Local minima trees are obtained from the main tree which is generated by the high level planner. Whenever a local minimum is detected and a backtracking point is required a subtree of length n is pruned from the main tree and added to the set of local minima trees. E.g. in Figure 4 a tree for the local minimum at sensor state S_i is presented.

At the end of a learning phase the set of local minima trees contains the relations between sensor states, robot actions and local minima that have been encountered for a certain object and this extra knowledge is used for further decisions at the high level planner. It is used to predict local minima and to chose backtracking points before the actual local minimum has been reached.

When the planner reaches a sensor state which is a root in one of the local minima trees, the next action is computed depending on the failure value F and the reinforcement value R of the off-spring nodes. In this way a different search strategy than best first is applied. If F is true the node is not considered, otherwise the action related to vertex that generates the sensor state with the lowest R value is chosen from the list of sorted offspring nodes. In the case a new failure occurs the parameters of the nodes in the tree are updated, i.e. no new tree is created.

It is important to notice that the sensor states and therefore local minimum trees are independent of the geometrical position of the robot in the workspace because the same sensor state can be found at different places in the workspace field.

In this way the knowledge acquired by the planner, using reinforcement learning, can be used in different situations. While the neural network provides local knowledge for obstacles avoidance, a local minimum tree provides a way to improve the performance of the general planner enhancing the system with the specific heuristics to overcome local minima.

Conclusions

In this paper we have described a way to enhance a general motion planner with learning techniques. In this way it is possible to improve the performance of the planner by learning the heuristics that consider an objects' size and shape. The method is especially interesting for grid potential field planning because potential field methods do not consider the objects size and shape and suffer from local minima.

For low level planning, neural network architectures with backpropagation learning are an appropriate technique because they are able to learn the relation between the potential field value as perceived through sensor points on the robot and the necessary action to minimize the potential. In this way the planner avoids to try all possible neighbouring configurations and immediately choses the optimal action. In addition it will generate only those actions that do not result in a collision. To obtain this functionality the network has observed the general planner during a learning period and has acquired the specific knowledge about the consequences of moving a robot with a certain shape in a workspace with obstacles.

The backtracking knowledge used at the high level planning is obtained by using a technique based on reinforcement learning. This technique relies on associating a failure value with each state of the robot and every possible action. This failure value is updated using a reinforcement strategy. Robot sensor states are organized in a tree and for each local minimum a subtree is stored. This set of subtrees enables to predict a local minimum and provides a mechanism to avoid it.

References

References

- [1] J. Barraquand and J. Latombe. Robot motion planning : A distributed representation approach. *The International Journal of Robotics Research*, 10(6), 1991.
- [2] J. del R. Millán and C. Torras. Learning to avoid obstacles through reinforcement: Noise-tolerance, generalization and dynamic capabilities. In *Proc. IEEE International Conference on Intelligent Robots and Systems IROS '92*, July 7-10 1992.
- [3] L. M. Gambardella and M. Haex. Grasps planning for automatic assembly tasks using artificial fields. In *Proc. European Conference on Artificial Intelligence - ECAI '92*, Vienna, Austria, August 3-7 1992.
- [4] M. Haex and L. M. Gambardella. Stable grasps by path planning using artificial fields. In *Proc. IEEE International Conference on Intelligent Robots and Systems IROS '92*, Raleigh, North Carolina, July 7-10 1992.
- [5] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1), 1986.
- [6] T. Lozano-Perez. Automatic planning of manipulator transfer movements. *IEEE Trans. Systems, Man, Cybernetics*, (SMC-11), 1981.
- [7] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations - Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, 1992.
- [8] R. S. Sutton, editor. *Reinforcement Learning*. Kluwer Academic Publishers, 1992.
- [9] L. Vincent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Machine Intell.*, 13(6), 1991.
- [10] Y. Xia. Skeletonization via the realization of the fire front's propagation and extinction in digital binary shapes. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 11(10), October 1989.