On Experimental Research in Sampling-based Motion Planning

Roland Geraerts
Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands, Email: roland@cs.uu.nl

1 Introduction

Motion planning is one of the fundamental problems in robotics. The motion planning problem can be defined as finding a path between a start and goal placement of a robot in an environment with obstacles. Over the past fifteen years, many different researchers have studied sampling-based motion planning techniques such as the Probabilistic Roadmap Method (PRM) [2]. This has led to many variants, each with its own merits. It is difficult to compare the different techniques because they were tested on different types of environments, using different underlying libraries, implemented by different people on different machines.

We have provided a comparative study and analysis of a number of these techniques, all implemented in a single system and run on the same test environments and on the same computer [1]. We encountered many difficulties and pitfalls during this study. We will identify them and discuss our solutions based on our experimental research over the past four years.

2 Methods

General setup Our goal was to create a system that facilitated conducting experiments easily and reducing making errors. We met this goal by creating a single motion planning system called SAMPLE (System for Advanced Motion PLanning Experiments) which we implemented in C++ using Visual Studio.NET 2003 under Windows XP Professional.

In a motion planning experiment, many choices exist for the components that compose a sampling-based motion planning algorithm. SAMPLE provides an easy API (Application Programming Interface) to add techniques (such as a particular sampling or local planning technique) to a component and to add its parameters. We created a GUI to set up an experiment easily (see Figure 1). We considered two types of experiments. The first type compares different techniques while the second type examines the influence of a particular parameter of a technique. An experiment can be saved to/loaded from disk to enable repeating the experiment. We created a command line version of SAMPLE to run the actual experiments. This program can run on a dedicated computer that has no processes running (such as a virus scanner or an internet connection) which could influence the results. Initially, we collected the experimental data manually. As this was quite a tedious job being sensitive to copy/paste errors, we decided to automate this by automatically collecting and processing the data.

We implemented many techniques and added them to SAMPLE. Sometimes it was hard to implement a technique in a way it was intended by the creator since not all details were always present in the paper (which is often due to space limitations). In some cases, we found these details in the source code which was available on the web.

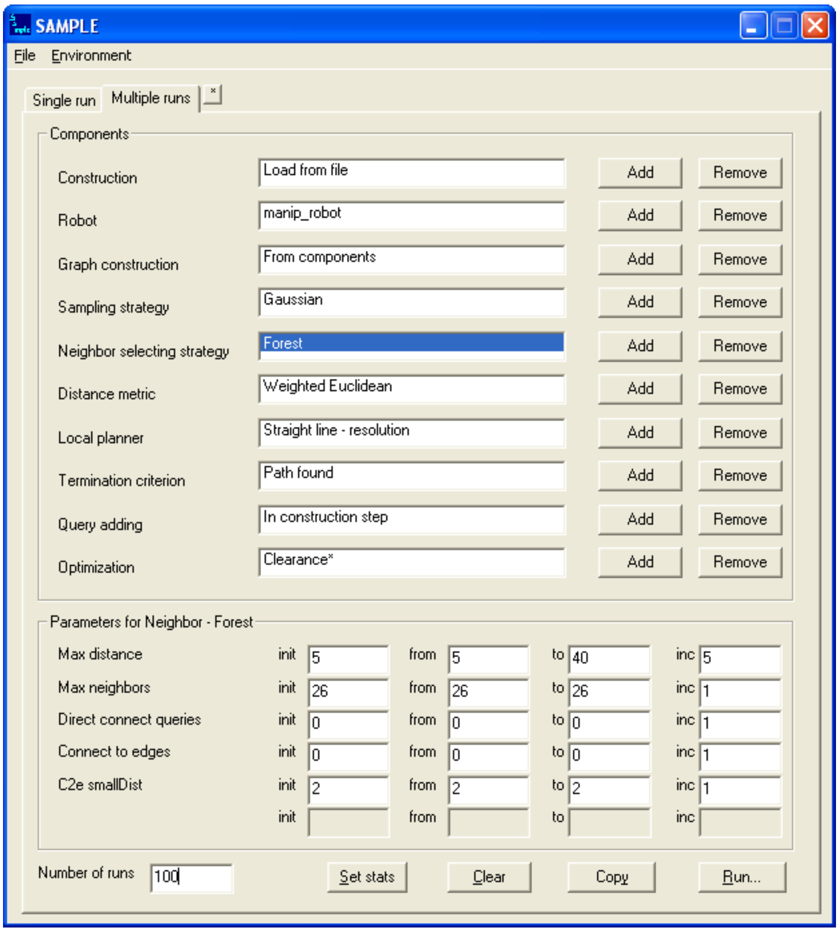

Figure 1 Setting up an experiment in SAMPLE. In this example, the effect the parameter 'max distance' of the 'Forest' neighbor selection strategy is studied. The parameter is varied between 5 and 40. For each different value, the experiment is carried out 100 times.

Representative problems The results of our comparative study indicated that some previous conclusions were too general which was probably caused by considering a set of examples that was too limited. We have to admit that it can be difficult to choose an appropriate set. In our study, we tried to employ environments and robots that resembled a wide range of configuration spaces. That is, we used environments with cluttered obstacles, narrow passages, many/few obstacles and scale differences. In addition, we used both small/large as well as different types of robots.

Some existing papers present a new technique only using examples satisfying its intentional goal. However, we think that also examples should be included that show its worst-case behavior and its limitations. For example, in our research on creating small roadmaps [1], we not only used an environment to show the potential of the algorithm, but also used an environment to show that other algorithms can be faster.

Interchangeability In our research group, initially, every member had its own implementation of his techniques which made it difficult to compare them. In addition, much work was wasted by creating components of software having comparable functionality. We decided to create libraries taking care of common functionalities such as a collision checker, a visualizer, and a graph library [5,6]. These libraries can be downloaded on the web. Also other libraries have been made available such as the Nearest neighbor library [7]. Besides the libraries, we encourage to make the source code of the complete system available such as done by [3,4].

Another important issue is the ability to exchange the geometry of environments and robots, as well as problem descriptions. We resolved this issue by using VRML as language for describing the geometry and XML as language for the descriptions.[1] There are great advantages of using existing languages: They are well-documented, parsers and type checkers are available for all appropriate platforms and programs exist to create and edit these descriptions. We think that the robotics community would benefit by supporting these languages.

3 Results

Evaluation of solution An issue each researcher has to deal with is how to evaluate the results. A common way to evaluate the results is to compare the new technique with existing techniques. In our comparative study, we initially compared techniques based on the time required to solve one relevant witness query. This however did not guarantee that every possible query could be solved by the roadmap that was constructed. We improved the study by evaluating the techniques based on solving every possible query. That is, we provided an analysis tool that indicated when the roadmap was dense enough to solve each query.

Another way to evaluate the results is to compare against the optimal solution. In some cases, we created an experiment for which the optimal solution is known. Unfortunately, such experiments can in general only be conducted for trivial cases. For more complicated experiments (such as the ones used for measuring path quality [1]), we tried to approximate the optimal solution by performing many runs. In addition, we used visual inspection to evaluate the results. In future research, we will also incorporate user evaluation to make the judgments.

[1] Many geometry files can be downloaded on http://www.give-lab.cs.uu.nl/movie/moviemodels.

Statistics Our comparative study showed that the variance in the running times was often large. This phenomenon is undesirable because of two reasons. First, a large variation complicates statistical analysis and can even make it unreliable. Second, it is undesirable from a users point of view, e.g. it can be hard to give a user an indication of how long the method will take to terminate. Hence, we had to be very careful analyzing the results. As the running times could vary extensively, we performed a large number of runs (i.e. 100) for each experiment. In this way we increased its statistical significance. In addition, we created box plots to provide insight in the distribution of the running times. (Such a box plot displays the middle 50% of the data, the average, the standard deviation and the minimum and maximum value.)

It may seem that deterministic techniques do not have such a 'variance problem'. Nonetheless, the study showed that a small change of the environment leads to a comparable amount of variance. Hence, care must be taken when deterministic techniques are involved.

4 Conclusion

It is often difficult to compare and evaluate techniques experimentally, because they were tested on different types of environments, using different underlying libraries, implemented by different people on different machines. By creating a system that facilitates integrating the techniques and automates conducting experiments, many errors can be avoided. To enable a fair and easy implementation of techniques, source code and software components should be made available. In addition, we encourage to use standard file formats (such as VRML and XML) to exchange problems easily.

When techniques have been implemented, they have to be evaluated by considering a large range of examples. Moreover, examples that show their limitations should also be included. A common way to evaluate the results is to compare techniques with existing ones. While such a comparison is often made based on running times, it may not always be convenient to use such a criterion. We think that incorporating user evaluation and user studies may be appropriate.

References

[1] R. Geraerts. *Sampling-based Motion Planning: Analysis and Path Quality.* PhD thesis, Utrecht University, http://www.cs.uu.nl/~roland/motion_planning/thesis.html, 2006.

[2] L.E. Kavraki, P. Švestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580, 1996.

[3] J.-C. Latombe. MPK: Motion planning kit. http://ai.stanford.edu/~mitul/mpk, 2006.

[4] S.M. LaValle. MSL: Motion strategy library. http://msl.cs.uiuc.edu/msl, 2006.

[5] D. Nieuwenhuisen. Atlas. http://www.cs.uu.nl/~dennis/atlas/atlas.html, 2006.

[6] D. Nieuwenhuisen. Callisto. http://www.cs.uu.nl/~dennis/callisto/callisto.html, 2006.

[7] A. Yershova and S.M. Lavalle. MPNN: Nearest neighbor library for motion planning. http://msl.cs.uiuc.edu/~yershova/mpnn/mpnn.htm, 2006.