Interactive Benchmark for Planning Algorithms on the Web: http://www.piaggio.ccii.unipi.it/benchplanning.html

Simone Piccinocchi, Massimo Ceccarelli, Federico Piloni, Antonio Bicchi

Centro "E. Piaggio"
Università di Pisa

Abstract

This paper presents an interactive environment available on the WorldWide Web intended to allow fair and thourough comparison of different techniques to solve a basic problem in nonholonomic motion planning. By connecting to the server, the user, potentially unaware of the technical subtleties of the planning problem, but well conscious of his application needs, can design the benchmark problem that is most significant to his purposes. The user can then obtain different solutions from several algorithm providers, and compare them both qualitatively (by graphic display), and quantitatively. Providers implement their own algorithms at their sites, with wide freedom of choice in programming language, computational architecture, etc., while complying with few simple protocol conventions. It is believed that similar usage of the Web, easily extendable to domains other than NHMP, can usefully contribute to the fair comparison of results among researchers, as well as to the diffusion of advanced research results towards application–oriented users.

1 Introduction

Work reported in this paper was motivated by the observation that in many domains of advanced research, traditional means of communicating and evaluating results are not as effective as they should be to allow an efficient dissemination of knowledge beyond the boundaries of rather strict, sometimes esoteric communities. This problem is naturally connected with the very nature of scientific thought, and with the fact that any scientifically mature discipline tends to create its own technical *jargon* to increase the efficiency of communication within its community. However, particularly for engineering disciplines, the lack of understanding of advanced research results causes many frustrations to both parties. The proliferation of the scientific literature in terms of number of books, journals and conferences recently occurred in many fields, is also a factor in distracting the attention of industry towards academic research.

Furthermore, even within the specialist clubs, it is sometimes very hard for a researcher to decide which algorithm solves a given problem best. Take for instance the problem of finding paths in a cluttered environment for a car–like vehicle (this example is going to be used as a paradigma throughout the paper). A "good" solution may be considered such based on the fact that it may achieve the shortest possible path, or the most regular one, or that it can compute it fastest. Also, an algorithm performing well in a much cluttered environment can be very awkward in a simpler situation; and it may so happen that, for all problems in a class of "reasonable" applications, an exponential–time algorithm performs better than one which is provably polynomial.

It is clear how having the possibility of seeing several different algorithms "at work" on specific problems would largely help in solving these problems. However, no fixed benchmark problem can exhaust the possibilities of variations in application problems. It would be important that the user could test algorithms based on a close replica of his application specifications, if he is from industry, or to build his own counterexample, if he is to explore weaknesses of existing solutions. Also, it seems highly desirable that algorithms to be compared are not implemented by a single, albeit impartial, programmer. Rather, each author or group should provide their own implementation, which often has been optimized through years and for which they can take the responsibility. Opennes to new algorithm providers is a key consideration in easing free competition and flow of ideas in the community.

A solution to the above requirements for an interactive benchmarking tool appears to be feasible today thanks to the development of net communications and programming technologies. Members of the scientific comunity have always regarded the Internet as a powerful means to exchange informations and resources, originally using it mostly for electronic mail and

for file transfers. More recently, we assisted to a quick growth of the net and, thanks to the introduction of graphic browsers, an even larger number of scientists currently use the web to present and retrieve information on ongoing research, consult libraries, watch real–time events.

Among the most innovative applications of the WWW facilities to robotics and related disciplines, we mention few who share some features with the one presented in this paper. *FixtureNET* by Goldberg *et al.* ([1]) and *CMU online fixturing system* by Mattikalli and Khosla ([2]) provide the user with site–specific algorithms to be applied to the solution of a grasping/fixturing problem interactively posed by the user. ARPA is funding the *ACORN* project to provide network based testbeds for manufacture and design algorithms ([3]). The Geometry Center at the University of Minnesota maintains a large collection of interactive geometric algorithms ([4]). Interactive experimentation with physical robots has also been explored, albeit somewhat preliminarily, in several sites (see e.g. the *Telegarden* installation of Goldberg *et al.*, [5]).

In the rest of this paper, we describe an extension of similar concepts to incorporate many of the specifications that we regard as necessary for a customizable interactive benchmarking environment for comparison of multiple algorithms on similar problems. In order to make the tool concrete, we implemented it for a specific research domain, i.e., motion planning for car–like vehicles. The background of such domain is briefly resumed in the next section.

2 Planning for car–like vehicles

The study of algorithms for motion planning of autonomous car–like vehicles under nonholonomic constrains has been one of the most active fields in recent years. Simply put, the problem is to find a path (possibly the shortest one) that takes a car–like vehicle from a given start point to a given final configuration, so as to avoid obstacles present in the environment. Most practical vehicles moving on wheels are not free to follow arbitrary paths, their motions having to comply with constraints on the direction of instantaneous velocity (which has to be consistent with the steering angle), as well as with bounds on the turning radius. Such nonholonomically constrained vehicles are usually not trivial to steer even for human drivers, to whom some training is necessary in order to learn maneuvers as e.g. parallel parking of a car in limited space.

The attention towards motion planning for car–like vehicles is due to both its application potentials (automated factories and highways, assisted parking maneuvering, etc.) and its theoretical challenges. The presence of lower bounds on the minimum turning radius involves curvature constraints on feasible trajectories that deeply affect the geometry of the problem. The problem of finding the shortest path between two configurations in the plane with curvature limitations is an interesting geometric problem *per se*, that was solved first by Dubins [8] for smooth trajectories, and, more recently,with reversals allowed, by Reeds and Shepp [16], Boissonnat, Cerezo, and Leblond [7], and Sussmann and Tang [17]. These theoretical results ignited research on new methods tending to find nonholonomic paths with bounded curvature amidst obstacles. Jacobs and Canny [9] developed an algorithm that determines a smooth trajectory for a car–like robot, composed of canonical trajectory pieces ("jumps"). Kanayama and Hartman [10] developed smooth local dynamic planners based on clothoids and cubic spirals. Laumond *et al.* [12] developed an efficient planner based on an iterative procedure that provides a feasible path starting from a free holonomic path. Mirtich and Canny [13] presented an algorithm to build a maximal clearance skeleton to be used as a roadmap, employing a Reeds/Shepp metric. Bicchi, Santilli, and Casalino [6] described a method that, under some particular conditions, provides a shortest path for a car–like vehicle among obstacles. Overmars and Svestka [14], and Kavraki and Latombe [11] presented efficient, probabilistically complete path planners. Paromtchik and Laugier implemented a real–time path planner on a modified commercial automobile, and experimented it in real parking problems [15].

As of today, it can be fairly stated that no sigle algorithm outperforms all others in this field. Rather, which algorithm is the best solution for a given problem is a matter of the specific problem, and of the user needs. This makes path planning for car–like vehicles a natural candidate domain for application of the interactive benchmarking architecture we are interested in exploring.

3 Structure of the interactive benchmark

The Interactive Benchmarking (IB) server realized at our WEB site is a graphic interface allowing users to design nonholonomic motion planning problems and to observe the behaviour of different algorithms as applied to that problem.

After connecting to the server, the user is presented with a workspace and forms that allow him to describe obstacles and the initial and final positions of the car, to build a specific problem that resembles a typical environment for his application, or a challenge to exi-

sting algorithms. Obstacles at present are restricted to be polygons with an arbitrary number of sides. Once the workspace is defined, the user can choose from a list of available algorithms. If the chosen algorithm is implemented in a third, remote site, then a concise description of the problem is sent to the remote server running the relative code, and a solution is sent back to the IB server, which shows it to the user. Besides a graphic display of the path found, such data as path length and time of execution can be displayed.

The user can subsequently choose another algorithm from the list, and apply it on the same problem. He can compare the two solutions and thus decide which one best suits his problem. It is also possible to save a particular problem that has shown important peculiarities of one of the algorithms (for example a problem whose solution an algorithm couldn't find), to point it to the authors or to other users. Some default problems are available, for users wishing to familiarize with the system.

Experienced users and researchers can provide their own algorithm and add it to the list of available choices so that other users can test it on their problems. The algorithm can be run on any computer connected to Internet, with limited requirements in terms of graphic browser support (any version of Netscape will suffice). Any such remote algorithm should adhere to a very simple input–output protocol; the remote computer should also execute a server daemon that accepts requests of algorithm activation from the net and sending back the results. Both the server code and the protocol are documented in our site.

Any programming language and operative system can be used to implement the remote algorithms. Obviously, as the remote algorithm code resides on the provider's own hardware, possible proprietary concerns are fully guaranteed by this mode of proceeding.

4 HTML interface description

After connecting to the interactive benchmark server, the introductory page of figure 1 appears on the user's browser. From this page one can either access, via the HELP button, an HTML guide of the application, or can start the application with the START button. Users that wish to provide their own algorithm for other users' tests can access, by the ADD button, on–line instructions on how a new algorithm can be added to the list of the available techniques and about the protocol to be followed to execute the algorithm remotely (in case the provider wishes to run his code on his own hardware).

By pushing the START button, the HTML page of figure 2 appears, with an empty rectangular workspace and a few buttons. A polyhedral obstacle

Figure 1: Introduction page.

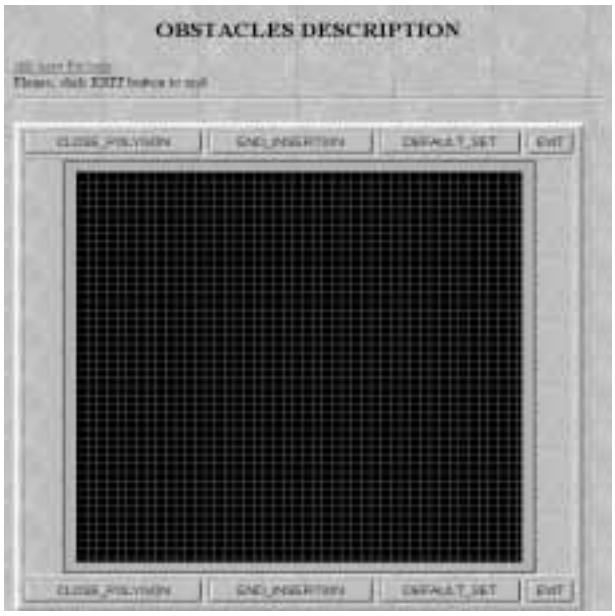

Figure 2: Ostacles description page.

can be introduced as a sequence of vertices, specified by the user as a sequence of points entered by clicking the mouse on the workspace. Hitting the button CLOSE_POLYGON, the sequence is ended, and vertices are automatically linked by edges in an clockwise order. A filled polygon, representing the obstacle, is then drawn on the workspace in an updated page. In order for the user to see the obstacle, it is necessary

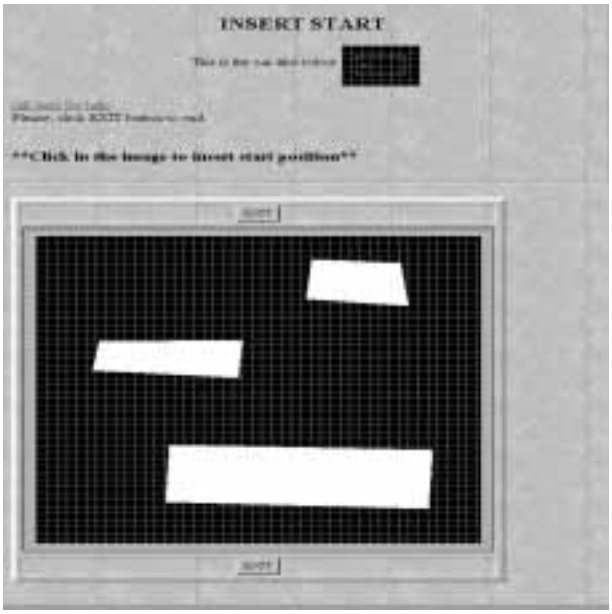

Figure 3: Start point description page.

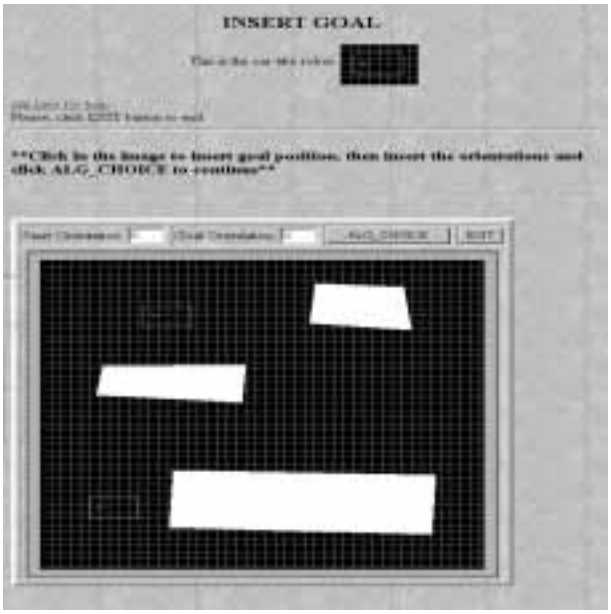

Figure 4: Goal point and orientation description page.

that the updated page is reloaded from the server. To force such refreshing, we adopted the technique of "client pull" linked to every CLOSE_POLYGON event.

An arbitrary number of such polyhedral obstacles can be inserted. Alternatively, an option for working in one of the default obstacle sets can be chosen by pushing the button DEFAULT_SET. After finishing the workspace description phase by pushing the END_INSERTION button (or by choosing a default workspace), the new HTML page of figure 3 is presented. The user is prompted to click the mouse on the portion of the workspace free from obstacles to choose the start position. This operation changes page to the one shown in figure 4, where the workspace with obstacles, the start position and two text field areas are shown, allowing the user to choose the goal position, and the start and goal orientations (in degrees). Having thus finished the problem description, by pushing the ALGORITHM_CHOICE button the page shown in figure 5 is brought forward, where a list of available algorithms is presented. After some minutes, depending on the problem complexity and the chosen algorithm, as well as on the net communication rate, a new page shows to the user the results as in figure 6.

In such result page, one or more pictures can be presented, representing the path found by the algorithm and possibly some auxiliary graphic construction used during computation and useful for analysis purposes. If the algorithm is executed locally, an information

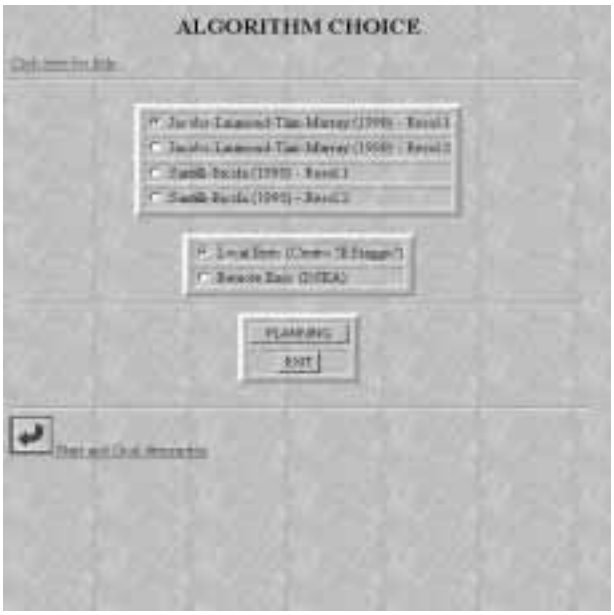

Figure 5: Algorithm choice page.

about the execution time, and a simple animation is also presented, showing the car-like robot following the path found as in figure 7. If the algorithm is implemented remotely, these features are momentarily not in effect. Using the BACK button on the interactive HTML pages, the user can now go back to choosing another algorithm to test on the same problem, or modify the problem changing the start and/or goal

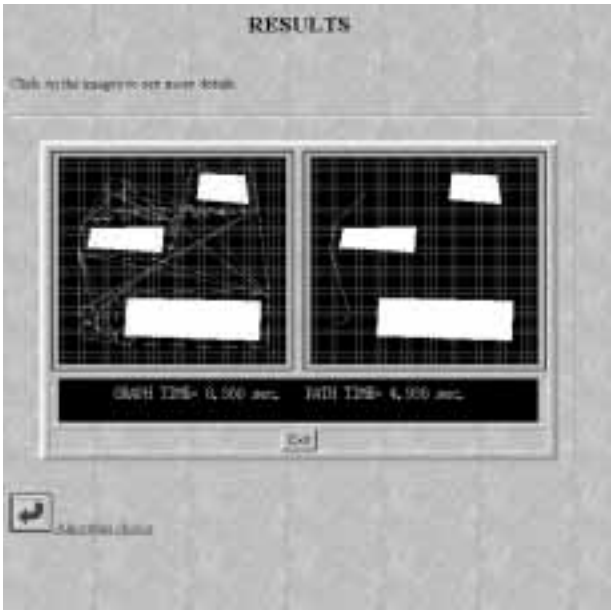

Figure 6: Results page.

Figure 7: Animation page.

position and/or orientation, by going upstream to the appropriate page.

5 Hardware & software architecture

The WEB NCSA HTTPD 1.5.1 server is running on a workstation Sparc 10 with 32Mb of RAM and a 2Gb SCSI hard disk. Our application is based on a set of forms, typical of the HTML language, and on

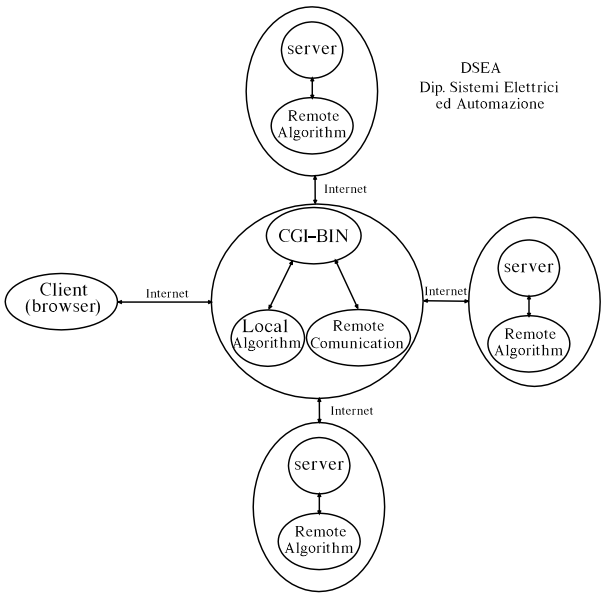

Figure 8: Structure of the IB server.

a set of CGI-BIN scripts (coded in C) that are executed by the server as a consequence of a request by the client. One of these scripts run the algorithm chosen by the users if it is local, otherwise it connects by the socket and TCP/IP protocol with the remote server where the code of the requested algorithm is resident. The protocol to be followed by users wishing to make remotely available their algorithm, is very simple: the remote host should run a server that receives, from the central IB server, a file containing the problem description (named `polyseg.inf`), launches the algorithm program execution, and sends a file (named `path.inf`) containing data about the solution back to the IB server. The program implementing the algorithm can be written in any language, and actually existing code can most often be used to this purpose, with the proviso that its I/O are redirected to read from file `polyseg.inf` and to write to file `path.inf`.

A schematic diagram of the application is reported in figure 8.

The structure of the problem description given in the file `polyseg.inf` is the following:

$$n\ v_1\ x_{11}\ y_{11}\ \cdots\ v_n\ x_{1n}\ y_{1n}\ \cdots\ s_x\ s_y\ s_o\ g_x\ g_y\ g_o$$

where:

n number of obstacles

v_i number of vertices of the i-th obstacle

$x_{ij}\ y_{ij}$ coordinate of the i-th vertex of j-th obstacle

s_x s_y g_x g_y coordinates of the start and goal points

s_o g_o orientations of start and goal configurations

The description of the solution path found by the remote algorithm relies at present on the assumption that such path is decomposable in a sequence of linear, circular and elliptical segments. This assumption is justified because of the above mentioned theoretical results on shortest paths of bounded curvature, but could be easily removed or modified for wider generality. Pointwise path descriptions are also acceptable, using sequences of one resolution–length linear segments. The `path.inf` file only contains the character "N" if the algorithm did not find a solution; otherwise, its structure is the following:

$$\begin{array}{l} \cdots \\ L\ x_i\ y_i\ x_f\ y_f \\ \cdots \\ A\ x_c\ y_c\ r_1\ r_2\ \alpha\ \beta \\ \cdots \end{array}$$

where:

$L\ x_i\ y_i\ x_f\ y_f$ represent a line from $x_i\ y_i$ to $x_f\ y_f$

$A\ x_c\ y_c\ r_1\ r_2\ \alpha\ \beta$ represents an arc of an ellipse centered in $x_c\ y_c$ with principal axes of length r_1, r_2, drawn counterclockwise from angle α to angle β

6 Present limitations and future developments

The prototype application we developed is by no means satisfactory of all specifications of a complete interactive benchmarking environment, although it clearly demonstrates its feasibility and potential usefulness. Among the contingent limitations of the present version of the IB server, there is the limited flexibility in designing the motion planning problem, with particular regard to the shape and kinematics of the car–like vehicle (in particular, position of the center of rotation w.r.t. the body). Allowing polyhedral obstacles only could be restrictive in replicating some environments. No provision is made for uncertain or mobile obstacles, for algorithms incorporating sensing, for multiple cars in the same workspace or for dynamic vehicle models.

The remote algorithm execution protocol does not at present incorporate the possibility of getting quantitative data on the solution provided, such as the length of the path and the number of reversal maneuvers employed. Another very useful feature for comparison would be the computational time required by remote algorithms to solve the problem. This datum is not at present considered in the remote execution output protocol, as bare execution time figures for codes running on different platforms can not be meaningfully compared.

At the moment of writing, the application incorporates two algorithms, namely those reported in [12] and in [6]. These can be executed both locally (i.e., on the same machine that runs the server), or remotely, at another site of our University. The code implementing the algorithm of [12] is not the original authors' version, and is actually far from being as efficient (the original code from the LAAS laboratory in Toulouse should be available soon).

Finally, the interactive environment description phase intensively uses net resources, so that, particularly at net's rush hours, it may be somewhat time consuming. Users are welcome to report further problems and/or suggestions about the IB server, using the authors' addresses or the messaging facilities in the application.

7 JAVA applet

Many of the above described limitations of the first version of the IB server are inherited from those of the tools used, i.e. the HTML language and CGI-BIN scripts. In fact, the first generation of WWW browsers were thought for the visualization of hypertext links, following which one can reach a remote WEB server that sends information to be displayed on the browsers area following the HTML rules. It is using these links that we simulated the interaction between the browser and the IB server.

On the other hand, the new generation of browsers (such as Hotjava, Netscape Navigator 3 and Microsoft Internet Explorer 3) allows local execution of pseudo-binary, platform–independent pieces of code, called "Java applets". The development of these new software technologies is extremely fast, and no version of the language nor of its development tools is stable at the time of writing. Nonetheless, the application perspectives are very exciting, and the technology suit interactive benchmarking almost perfectly. We therefore undertook implementation of a second version of the IB server employing the new JAVA technology. The structure of the IB application with JAVA is completely different from the one described above. By choosing the JAVA option in the opening page `www.piaggio.ccii.unipi.it/benchplanning.html`, the user's browser (which has to be recent enough to support JAVA applications) receives the device–independent code of an applet, which is launched on the user's host and produces the graphic output of figure 9. Construction of obstacles in the workspace is done within this interactive page by the user,

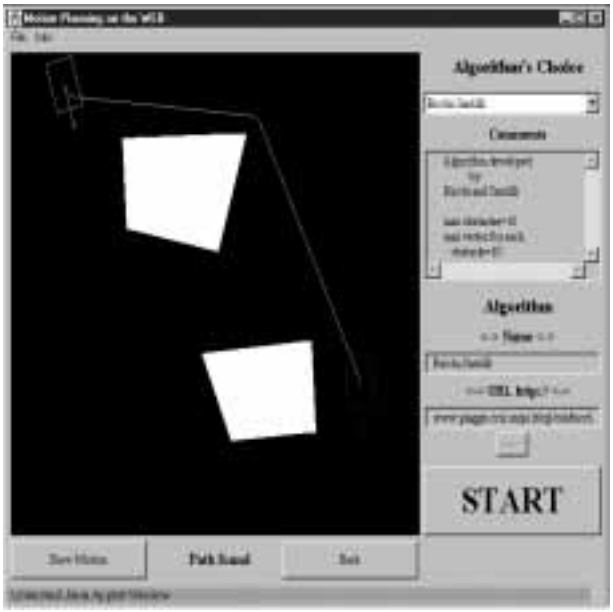

Figure 9: Java interface page.

without resorting to any communication with the IB server. Obstacle description is much faster and more flexible, allowing the user to edit previously introduced polyhedra by adding or deleting vertices, displacing the obstacle, etc. The start and goal positions are fixed by clicking with the mouse in the free workspace, and the orientation is chosen by dragging the mouse while watching in real time at the display of the environment. Once the problem has been setup, by clicking on the START button a string with the problem data is sent to the server running the requested algorithm. After execution, the latter sends results back to the applet on the users' browser, which displays them by graphic animation and synthetic data.

8 Conclusions

We reported on an interactive WWW application intended for allowing potential users of motion planning algorithms for car–like vehicles to examine existing algorithms directly on an application–specific problem, so as to allow fair and effective comparison in close–to–real conditions. Notwithstanding the limitations of this first version (part of which are being solved in the second version using JAVA technology), results are encouraging towards development of similar intefarces for this and other problems in robotics and algorithm design at large.

References

[1] `http://teamster.usc.edu/fixture`

[2] `http://www.cs.cmu.edu/user/rajum/www/fix4.html`

[3] `http://www.eit.com/projects/acorn/`

[4] `http://www.geom.umn.edu:80/apps/`

[5] `http://cwis.usc.edu/dept/garden/`

[6] A. Bicchi, G. Casalino, and C. Santilli: "Planning Shortest Bounded–Curvature Paths for a Class of Nonholonomic Vehicles among Obstacles", Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1349–1354, 1995.

[7] J.D. Boissonnat, A. Cerezo, and J. Leblond: "Shortest Paths of Bounded Curvature in the Plane", Proc. IEEE Int. Conf. on Robotics and Automation, pp.2315–2320, 1992.

[8] L. E. Dubins: "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents", American Journal of Mathematics, vol.79, pp.497–516, 1957.

[9] P. Jacobs, J. Canny: "Planning Smooth Paths for Mobile Robots", *IEEE International Conference on Robotics and Automation*, AZ, 2–7, 1989.

[10] Y. Kanayama and B.I. Hartman: "Smooth Local Path Planning for Autonomous Vehicles", Proc. IEEE Int. Conf. on Robotics and Automation, pp. 1265–1270, 1989.

[11] L. Kavraki and J.-C. Latombe: "Randomized preprocessing of configuration space for fast path planning", in Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2138–2145, 1994.

[12] J. P. Laumond, P. E. Jacobs, M. Taix and R. Murray: "Fast and Exact Trajectory Planning for Mobile Robots and Other Systems with Nonholonomic Constraints", Technical Report 90318, LAAS/CNRS, Toulouse, France, September 1990.

[13] B. Mirtich, J. Canny: "Using Skeletons for Nonholonomic Path Planning among Obstacles", *IEEE International Conference on Robotics and Automation*, pp. 2533–2540, May 1990.

[14] M. Overmars and P. Svestka: "A Probabilistic Learning Approach to Motion Planning", Proc. First Workshop on Algorithmic Foundations of Robotics, pages 19–37. A.K. Peters, Boston, MA, 1994.

[15] I.E. Paromtchik and C. Laugier: "Motion Generation and Control for Parking an Autonomous Vehicle", Proc. IEEE Int. Conf. on Robotics and Automation, pp. 3117–3122, 1996.

[16] J. A. Reeds, R. A. Shepp: "Optimal Paths for a Car that Goes both Forward and Backward", Pacific Journal of Mathematics, vol. 145(2), 1990.

[17] H. J. Sussmann and G. Tang: "Shortest Paths for the Reeds–Shepp Car: a Worked Out Example of the Use of Geometric Techniques in Nonlinear Optimal Control", SYCON report 91–10, 1991.