2013 European Conference on Mobile Robots (ECMR)
Barcelona, Spain, September 25-27, 2013

Teletesting: Path Planning Experimentation and Benchmarking in the Teleworkbench

Andry Tanoto[1], Javier V. Gómez[2], Nikolaos Mavridis[3], Hanyi Li[4], Ulrich Rückert[4] and Santiago Garrido[2]

Abstract— **Experimental evaluation of navigation algorithms requires physical robots as well as position sensing devices. The common alternative is to use simulations to run the experiments. However, simulation often does not provide an accurate prediction of real-world behavior. Therefore, in this paper, we present an innovative approach towards evaluation of navigation algorithms, which does not need physical robots and position sensors to be present at the experimenter's site, but relies on a special remote internet-accessible physical testbed, the "Teleworkbench", which can be used in order to evaluate as well as uniformly cross-compare algorithms with no need of spending money on hardware or simulation software. More specifically, in this paper we are using the Teleworkbench to evaluate three different path planning algorithms, and compare it with simulation. Different metrics are proposed, such as the path execution time, smoothness and path clearance deviations. Our results clearly illustrate the superiority of the Teleworkbench as an evaluation platform in comparison to simulation, which does not provide an accurate prediction of actual physical performance, and thus illustrate both the viability as well as the power of our novel approach.**

I. INTRODUCTION

Traditionally, experimental evaluation of navigation algorithms requires physical robots, as well as position sensing devices, to be available at the experimenter's lab. As an alternative, many authors have used simulation in order to run such experiments. However, simulation often does not provide an accurate prediction of real-world behavior. Therefore, in this paper, we present an innovative approach towards evaluation of navigation algorithms, which does not need physical robots and position sensors to be present at the experimenter's site, but relies on a special remote internet-accessible physical testbed, the "Teleworkbench" [1], which many remote experimenters can use in order to evaluate as well as uniformly cross-compare their algorithms with no need of spending money on hardware or simulation software.

One of the first attempts to come up with a benchmark definition in path planning is given in [2]. Many benchmarks for motion planning have been already proposed, but these approaches are very dependent on the used algorithms: probabilistic planners [3], humanoid problems [4], GPU-based algorithms [5] and so on. Recently, a generic simulation infrastructure has been proposed for benchmarking mobile manipulators path planning algorithms [6].

In spite of all these efforts, the benchmarks never go further than simulations. Benchmarking with real robots could be a very complex, time-consuming task and the performance is not comparable among robots and implementations, because control parameters can highly influence the results. Why is benchmarking in real robots important? There are factors that simulations can hardly take into account: perturbations (both spatial and temporal), deviations due to errors, noise, etc. Some metrics such as plan execution time or deviations between real and simulated plans are required in order to check the reliability of the different algorithms. The path following and control algorithm for the real robot play a very important role in this benchmarking, and also other subsystems of the robot, such as localization, odometry, etc.

The Teleworkbench (TWB) offers a controlled environment in which users in any location can execute, test, and compare their algorithms and programs using real robots. The TWB also provides functionality for assisting researchers and developers in several aspects of experimentation using robots: (i) integration with a robot simulator, (ii) download and execution of users' robot programs, (iii) automatic environment building, (iv) data logging, (v) position tracking of up to sixty-four robots, and (vi) a visualization tool for experiment analysis. As experiments run in a controlled and repeatably rebuilt environment, researchers can reproduce and compare the results of the experiments. In this paper we are using the Teleworkbench to evaluate three different path planning algorithms and compare them with the simulation results. The paths are computed offline in *a priori* known, static map. Therefore, the scope is to evaluate the planning algorithms independently, without taking into account onboard sensors of the robots. The position of the robots is provided by the TWB.

The paper is organized as follows: Section II describes the architecture we propose for remote experimentation of navigation algorithms. In Section III the complete setup of the simulation and experiments is detailed. The results are shown in Section IV. Finally, in Section V the main conclusions of the paper are outlined.

---

*This work is partially supported by the spanish Ministry of Science and Innovation under the projects DPI2010-17772 and CSD2009-00067 and by the German Research Foundation DFG under the project CoE 277: Cognitive Interaction Technology (CITEC).

[1] A. Tanoto was with System and Circuit Technology, Heinz Nixdorf Institute, University of Paderborn, Fürstenalle 11, 33102 Paderborn, Germany `andry.tanoto@hni.upb.de`

[2] J.V. Gómez and S. Garrido are with RoboticsLab, Carlos III University of Madrid, Av. de la Universidad 30, 28911, Madrid, Spain `jvgomez@ing.uc3m.es`

[3] N. Mavridis is with SKEL group at the Institute of Informatics and Telecommunications, NCSR Demokritos, Agia Paraskevi Attikis, P.O.Box 60228, 153 10, Athens, Greece `nmavridis@iit.demokritos.gr`

[4] Hanyi Li and Ulrich Rückert are by the Cognitronics and Sensor Systems Group, Cognitive Interaction Technology Centre of Excellence (CITEC) Bielefeld University, 33615 Bielefeld, Germany `hli,rueckert@cit-ec.uni-bielefeld.de`

978-1-4799-0263-7/13/$31.00 ©2013 IEEE

II. SYSTEM ARCHITECTURE

We focus on the point of view of the user who wants to test a path planning algorithm. The most desirable characteristic is that the experimentation procedure should be fully transparent for the user. In this way, the user uploads a program with the path planning algorithm to the Teleworkbench Server and executes it remotely, as shown in Figure 1. The user only has to care about the environment dimensions adaptation and to output the computed path in a specific format: a string sent via TCP/IP.

This string can be received by the robot controller, which subsequently communicates it to either the Player/Stage simulator [7] or the robots in the Teleworkbench. When the message with the path is received, the robot localization and path execution is done automatically. While the experiment is running, the user is able to follow it thanks to video streaming. Once the experiment finishes, a log is created in which different data are collected such as robot positions, sensor data, execution time, etc.

In the following, we describe the different modules of the implemented architecture.

A. Path Planning - User

This module encapsulates the path planning algorithm and all the additional steps which are necessary for the computation of the path: environment adaptation, path trimming and conversion, etc. The user of the Teleworkbench only has to deal with this module since it depends only on the algorithm.

Any path planning algorithm can be used, in any programming language as long as the TCP/IP output satisfies the established format (as described in Figure 1, we have used Matlab compiled code in this paper). The focus of the proposed architecture is mobile robot navigation. Hence, 2-dimensional planning should be done, or 3-dimensional if the heading angle θ is wanted to be include in the planning. Depending on the algorithm employed, previous steps could be required. For example, when running algorithms in which the robot is taken into account as a point with no dimensions, it is recommendable to dilate the obstacles of the environment by the radius or the robot. Otherwise the robot will collide.

Once the path is obtained, it has to be adapted so that it can be sent via TCP/IP commands to the robots. This adaptation applies, for example, path trimming (uniform sampling of the path so that it is not necessary to send all the points of the path), heading computation for every point of the path (if required), and so on.

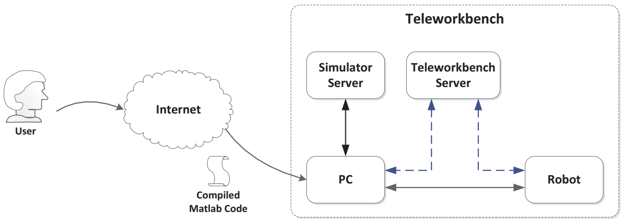


Fig. 1. The deployment diagram of the experiments with Stage simulator or BeBot minirobot.

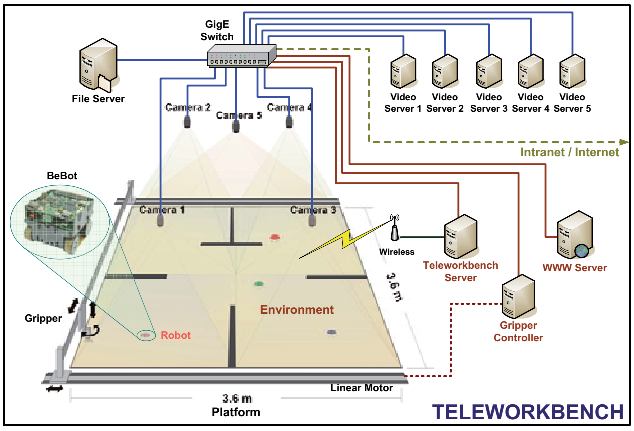


Fig. 2. The diagram showing the general system architecture of the Teleworkbench system.

B. Teleworkbench

The distributed system architecture of the Teleworkbench is shown in Figure 2. Earlier papers [1], [8] describe the Teleworkbench System in more detail. In this paper, we will briefly describe the system and its main components.

The Teleworkbench comprises a main experiment field of 3.6×3.6m that is partitionable into four sub-fields, each of which can be used for an experiment independently. A gripper module with four degrees of freedom (3 translational and one rotational) enables automatic environment building by using plastic blocks. Additionally, it can also be used for placing robots at predefined locations and orientations. Three different robotic platforms are currently supported by the Teleworkbench: *Khepera II*, *Khepera III* [9], and the *BeBot* [10]. Five 1-megapixel Gigabit-Ethernet cameras are mounted above the experiment field, four of which are assigned to the sub-fields and the other one monitors the entire experiment field. Each camera is connected to a video server that processes the video data to provide the GPS-like position and orientation information of the robots as well as to record and stream the video. Currently, up to 64 robots can be identified and tracked by means of barcode-like markers that are placed on top of the robots. One server, called the Teleworkbench Server, is responsible for scheduling, queuing and execution of experiments. Additionally, the server handles wireless communication among robots, e.g. with Bluetooth or WLAN. Another server is assigned for intermediating users and the Teleworkbench System. A website is provided to support users in performing different activities, e.g. experiments setup and execution, experiment data acquisition, or live-monitoring. A file server is deployed to store all data that accumulates during experiments.

The Teleworkbench aims to provide a seamless transition from simulation to experiments with real robots. The same environment model that is used in the simulator can be used for the experiment. When the experiment is set and ready, the defined environment model is realized by using plastic blocks arranged by the gripper module. Afterwards, the uploaded programs are deployed and executed.

During experiments, the communicated messages among agents are logged and can be retrieved after the end of the experiment. At the same time, users can also observe the experiment using the developed graphical user interface (GUI) that can display the streamed live-video overlaid by some robot information such as robot symbol, robot path, sensor information, and exchanged messages (see Figure 3).

C. Robot Platform

The experiments and simulations detailed in this paper are carried out with BeBot minirobots [10]. The robot controller uses a modular and flexible robot software architecture (see Figure 4), which is based on the schema-based architecture of Arkin [11]. In this architecture, multiple concurrent processes, called motor schemas, generate different behaviours that are represented by a vector, whose value and angle corresponds to the speed magnitude and orientation respectively. The developed software architecture also provides an abstraction of the robot controller, which allows the easy deployment of different robot controllers on the robot. Additionally, the robot software architecture is composed of modules, each of which realizes one specific functionality. Furthermore, the software architecture enables us to instill a certain safeguard mechanism on the robot to tackle problems due to erronous behaviours during runtime. For this study, the robot controller contains one path follower schema that enables the robot to traverse a given list of positions.

A server is also deployed on the robot to provide access to the robot, e.g. to send commands to the robot. A robot communication protocol has been defined, consisting of a list of commands that the robot supports. The command that is of interest in this study is the one for sending a list of points that the robot has to traverse: $T, T, P, px_1, py_1, pa_1; px_2, py_2, pa_2; ..., px_n, py_n, pa_n$, where px_n,py_n, and pa_n are the position (x and y) and the orientation (pa_n) of the target point.

D. Simulation Server

A simulation server is deployed to support simulation of the robot controller before executing it on the BeBot. The server runs a Linux operating system with the Player/Stage robot simulator [7]. One robot server with access to the aforementioned robot controller is also deployed on the same machine. As in the case of the server running on the robot, this robot server is also programmed to receive the same commands, e.g. the robot path. The command will be sent to the robot controller, which in turn translates it to the command understood by the Stage simulator.

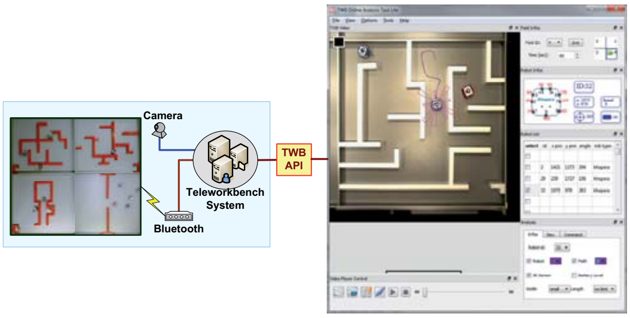


Fig. 3. The GUI for online analysis tool. The API is used to communicate with the Teleworkbench System as well as with the robots.

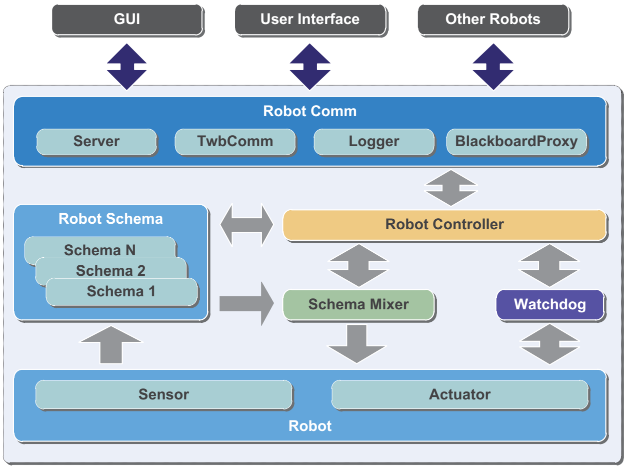


Fig. 4. The robot software architecture based on motor schema architecture.

III. SIMULATION AND EXPERIMENT DESCRIPTION

The proposed benchmarking architecture contains three different steps: computation, simulation and experimentation. The first stage, computation, comprises the path planning algorithm as detailed in section II-A. In our case, this is done with Matlab compiled packets.

The TWB supports the interoperability with the Stage robot simulator. Depending on the IP address specified in the previous step, the path will be simulated or executed in the TWB. During our experiments, we simulated the paths in order to verify them before proceeding with real experimentation.

Finally, the last stage is the experimentation with BeBot minirobots in the TWB. During the experiments, all the necessary data for the metrics detailed in the previous section are recorded.

The experiments comprise a total of 24 real trajectories. Two different environments are used: a room like environment and one with blocks and several intersections, as shown in Figure 8. Two different plans (set of start and goal points) were requested in each environment. Also, for each plan, three different path planning algorithms have been used. Finally, each path planning algorithm was executed twice.

The path planning algorithms chosen are the Fast Marching Method (FMM), Fast Marchig Square (FM^2) [12] and Probabilistic Road Maps (PRM) [13], implemented with the Robotics Toolbox [14]. These algorithms were chosen since they are quite different from each other. FMM provides optimal paths in terms of distance, but with sharp curves and runs too close to obstacles. Paths computed with FM^2 are very smooth, but longer. And PRM provide stochastic paths which are not smooth but faster in high-dimensional spaces.

A. Metrics Employed

The metrics we have included in this paper are those related to the execution of the path and to the comparison between the computed path P_0 and performed path P_r (performed in the simulation as well as in the TWB):

- **Path execution time** - The time t (in s) the robot took to follow the path from the initial given point until the target is reached.
- **Path deviation (error)** - Path deviation e_p (in mm) is measured by dividing both paths into n points. For each point of initial path another one is chosen on the real path. This point is chosen so that the Euclidean distance d_E (error) is minimum.
- **Path smoothness** - The smoothness κ' can be measured in many different ways. We will use the smoothness metric given in [6], which represents the standard deviation of the angles along the path. Let α_i be the angle between two consecutive segments of a path divided into m segments. Therefore, $\kappa' = \sqrt{\frac{1}{m-1}\sum_{i=2}^{m}\alpha_i^2}$. The angle taken into account is illustrated in Figure 5.
- **Path length** - The path length l is approximated by dividing the path into n points $P = \langle p_1, p_2, ..., p_n\rangle$ and computing $l = \sum_{i=1}^{n-1} d_E(p_i, p_{i+1})$, where d_E stands for the Euclidean distance.
- **Minimum Obstacles clearance** - The metric d_n contains the deviation of the minimum distance of the points along the path to the closest obstacles of the environment.
- **Average speed** - This metric (given in m/s) is computed as follows: $v = l_r/t$.

IV. RESULTS

Graphs in Figure 6 show the results of the simulation with Stage robot simulator and the experimentation with the Teleworkbench in terms of the metrics described in section III-A. In path deviation we also calculate a direct comparison between the results of Stage simulation and the Teleworkbench.

In Figure 7 the results for smoothness, clearance and path length are shown as the ratio with regards to the initial, computed path. The objective is to show the deviation between the computed and performed paths (in both simulations and real executions).

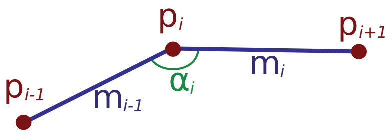


Fig. 5. The angle between two consecutive segments of a path.

Many interest conclusions can be extracted from the results. First, the duration of the real path executions in the TWB take longer than in the simulated environment (the top left graph in Figure 6), so the average velocity is lower in real experiments (the top middle graph in Figure 6). In these two cases, the variation is not constant among the algorithms. The top right graph in Figure 6 shows the higher path deviation in real experiments is comparison to the one in simulation. However, in this case the variation among algorithms is almost constant. The results with the minimum clearance follow the same pattern.

It is interesting that the path length deviation (the bottom middle graph in Figure 6) shows similar results in simulation and in real experiments. Finally, regarding path smoothness (the bottom right graph in Figure 6), the controller is not able to reproduce the paths as smooth as planned. The most interesting point here is the smaller deviation between simulation and TWB that occurs in the case of the PRM algorithm, where the smoothness is lower.

Focusing on the ratios with the initial path, Figure 7, the main conclusions can be extracted. The real experiments always reported worse results than the simulation: path length ratio is always over 1, which means that it has increased, and path length and smoothness are decreased. There is only one exception, the smoothness of the PRM algorithm is improved in this case. This is because the implemented controller is not able to follow the sharp curves which a PRM path is characterised for.

Therefore, the main conclusion of the result is that the simulation of path planning algorithms is useful, but benchmarking with results obtained only through simulation is not enough. The application of the different algorithms in the real world can have different results than those provided in simulation. This is a known problem: simulations can be as close to reality *as desired*, but to represent all the external factors that influence the real performance is very complex (and most of the times not worthy) task. Also, the main problem that arises when executing algorithms in real robots is that the deviations between simulations and real world are not constant, as shown in the results of this paper.

Figure 8 shows the computed, simulated, and TWB-generated robot path generated by the three algorithms in two different environment configurations. The results show that the simulated and TWB-generated paths are close to the computed one. However, some overshoots are visible which results from the inability of the P-controller used as the path follower in this study to keep the robot always on track. This issue is more prevalent in the results of the Teleworkbench. The experiments in the Teleworkbench produce longer robot path, as is shown in the path length graph of Figure 6.

In Figure 9, we can see the snapshots of the same experiment running in the Stage simulator and the TWB. The robot path is overlaid on the picture as well as on the video. Using the Teleworkbench GUI, this can be done either online (during runtime) or offline (after the experiment).

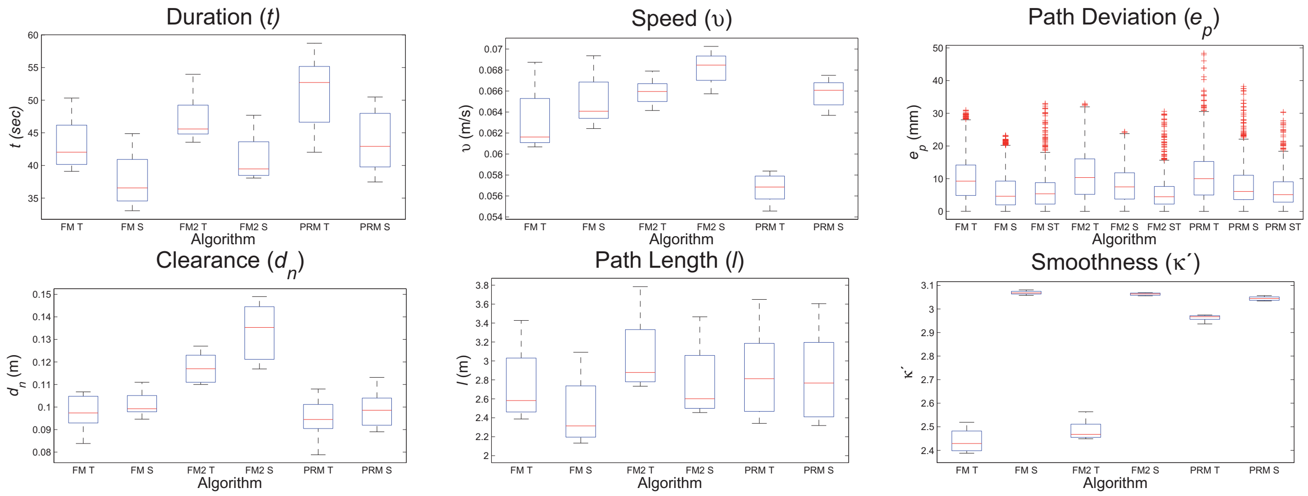


Fig. 6. The results of the simulation with Stage simulator (with suffix S) and the experimentation with the Teleworkbench (with suffix T). Suffix ST is to indicate the results of direct comparison between the simulation and experimentation with the Teleworkbench.

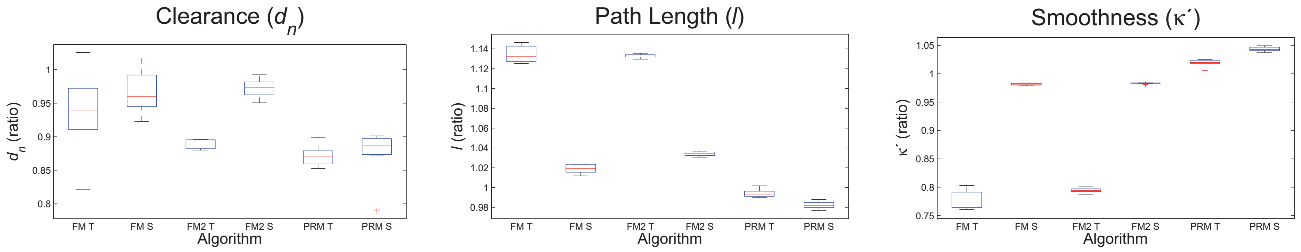


Fig. 7. Results for the length, smoothness and clearance shown in terms of ratios (performed path/initial path) of the simulation with Stage simulator (with suffix S) and the experimentation with the Teleworkbench (with suffix T)

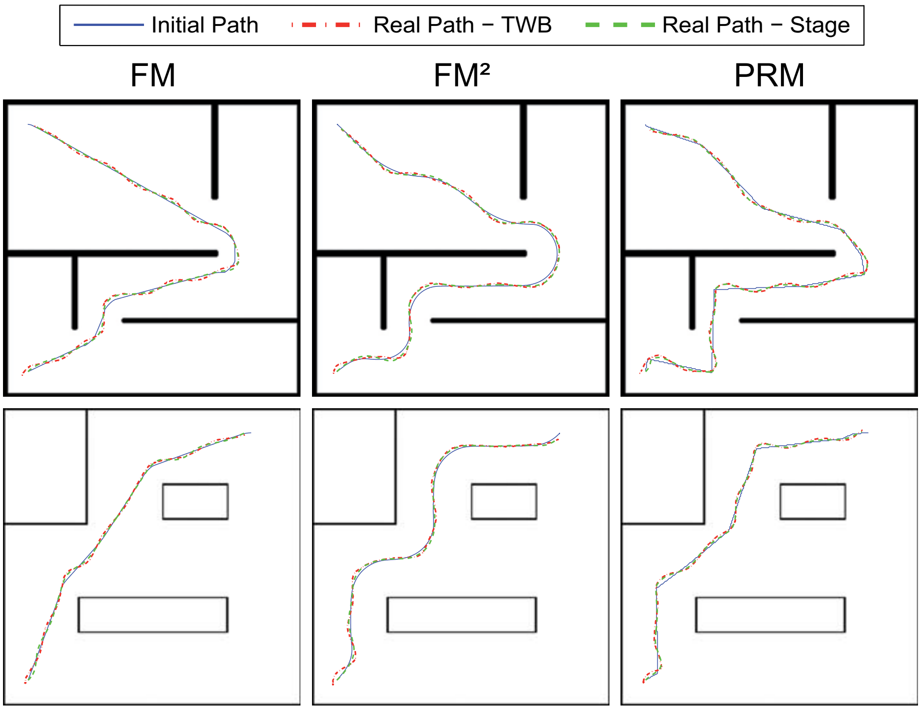
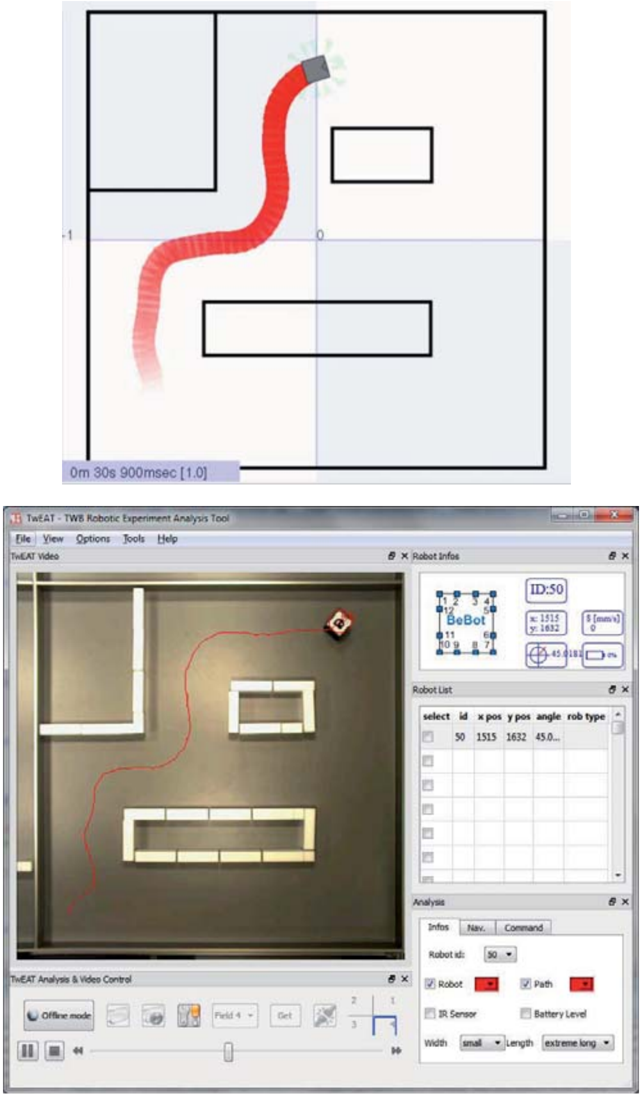


Fig. 8. The results of the simulation in Stage robot simulator and the Teleworkbench for both types of environment and different sets of start and target positions.



select	id	x pos	y pos	angle	rob type
☐	50	1515	1632	45.0...	
☐					
☐					
☐					
☐					
☐					

Robot id: 50

☑ Robot 🟥 ☑ Path 🟥

☐ IR Sensor ☐ Battery Level

Width small Length extreme long

Fig. 9. The snapshots of the experiments running on Stage simulator and the Teleworkbench. The path of the robot is embedded on the video using the Teleworkbench GUI.

V. CONCLUSION

In this paper we have introduced a novel architecture to remotely test and benchmark path planning algorithms using the Teleworkbench. Six different metrics have been proposed in order to take into account the quality of the implementations of path planners into real robots.

This infrastructure allows people around the world to test a path planning algorithm very easily, without spending a lot of efforts for implementing the algorithms in real robots and dealing with the typical implementation problems.

Results show that when dealing with the implementation of path planning algorithms in real robots the metrics obtained in simulation are not completely valid in the real world. Although this is highly dependent on the control strategy employed, if the same controller is applied in all the experiments similar results are expected.

For example, in our case the controller is not able to follow such smooth paths as those given FM^2. However, this is not a problem since the paths computed with FM^2 have a higher average speed than those computed with FM or PRM. When benchmarking path planning algorithms in simulation, these issues are not usually taken into account, but they can be very important in the real applications.

This infrastructure is also valid for testing and comparing path following algorithms and motion controllers. In that case, using the same path planning algorithm the same metrics can be employed in order to compare the quality of the controllers.

The future work focuses on including more algorithms to the test and extending the benchmarking to other algorithms such as multirobot path planners and planning with dynamic obstacles. Also, the proposed schema is applicable to benchmark the influence of sensor noise and inaccuracies of the control in the paths. In addition, it could be interesting to compare the performance of the sensor models in simulation with the real sensors.

REFERENCES

[1] F. Werner, U. Rückert, A. Tanoto, and J. Welzel, "Teleworkbench - a platform for performing and comparing experiments in robot navigation," in *IEEE Intl. Conference on Robotics and Automation*, 2010.

[2] J. Baltes, "A Benchmark Suite for Mobile Robots," in *IEEE/RSJ Intl. Conference on Intelligent Robots and systems*, vol.2, pp. 1101–1106, 2000.

[3] R. Geraerts, "Sampling-based motion planning: Analysis and path quality," Ph.D. dissertation, Utrecht University, 2006.

[4] J. Kuffner, S. Kagami, M. Inaba, and H. Inoue, "Performance benchmarks for path planning in high dimensions", in *JSM Conference on Robotics and Mechatronics*, June 2001.

[5] J. Pan, C. Lauterbach, and D. Manocha, "g-Planner: Real-time Motion Planning and Global Navigation using GPUs," in *AAAI Conference on Artificial Intelligence*, 2010.

[6] B. Cohen, I. A. Şucan, and S. Chitta, "Generic Infrastructure for Benchmarking Motion Planners," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, pp. 589–595, 2012.

[7] B. P. Gerkey, R. T. Vaughan, A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. In Proceedings of the International Conference on Advanced Robotics (ICAR), pp. 317–323, 2003.

[8] A. Tanoto, U. Rückert, and U. Witkowski, "Teleworkbench: A teleoperated platform for experiments in multi-robotics," in *Web-based Control and Robotics Education*, vol. 38, chapter 12, pp. 287–316. Springer Verlag, 2009.

[9] K-Team Corp., "Khepera III, http://www.k-team.com/mobile-robotics-products/khepera-iii.

[10] S. Herbrechtsmeier, U. Witkowski, and U. Rückert, "Bebot: A modular mobile miniature robot platform supporting hardware reconfiguration and multi-standard communication, in *Progress in Robotics*, Springer Berlin Heidelberg, vol. 44, no. 5, pp. 346–356, Aug. 2009.

[11] R. C. Arkin, "Motor Schema-Based Mobile Robot Navigation, in: The International Journal of Robotics Research, vol. 8, August, no. 4, 92–112, 1989.

[12] A. Valero, J. V. Gómez, S. Garrido, and L. Moreno, "Fast Marching Method for safer, more efficient mobile robot trajectories," accepted in *IEEE Robotics and Automation Magazine*.

[13] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic Roadmaps for Path Planning in High-dimensional Configuration Spaces," in *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[14] P. Corke, "Robotics, Vision and Control - Fundamental Algorithms in MATLAB®," in *Springer Tracts in Advanced Robotics*, vol. 73, Ed. Springer, 2011.