Towards Quantitative Comparisons of Robot Algorithms: Experiences with SLAM in Simulation and Real World Systems

Benjamin Balaguer, Stefano Carpin
School of Engineering
University of California, Merced, USA
E-mail: {scarpin,bbalaguer}@ucmerced.edu

Stephen Balakirsky
National Institute of Standards and Technology
E-mail: stephen@nist.gov

Abstract— **Autonomous robotics has been plagued by the lack of quantitative comparisons between different solutions for the same problem. The situation arose due to a lack of theoretical background, recognized benchmarks, and the existence of a culture that is not oriented towards the free sharing of ready-to-use code for scientific research. In this paper we leverage a recent paradigm shift, and contrast different algorithms for Simultaneous Localization And Mapping (SLAM) readily available to the scientific community. In particular, we have run the same algorithms in two different settings. The first one is based on a P3AT robot operating inside a large building hosting office space and research labs. The second scenario is a virtual replication of the identical floor plan, implemented inside the USARSim simulation environment. In other words, the simulated scenario features the exact models of the environment and robot. The experimental setup offers a matrix where weaknesses and strengths of different SLAM algorithms can be contrasted in real and virtual environments, also outlining the degree to which the simulated results can be extrapolated to measure or predict real world systems performance. We conclude that the availability of open source algorithm implementations, data sets, and simulation environments is the key to promote accelerated research in autonomous robotics. In particular, it appears that available SLAM implementations are robust and easy to use for environments like those used in our experiments, and therefore research efforts should be accordingly re-modulated.**

I. Introduction

One of the cornerstones of the scientific method is *repeatability*. Experimental tests confirming or disproving a certain theory should be carried out by different researchers and lead to the same conclusions, within the defined error bounds, provided that the operative conditions are the same. Robotics has not yet enjoyed such a rigorous approach, a fact that can be explained by a multitude of reasons. Robots are complicated systems composed of many interacting units, each of them characterized by its own behaviors and errors. Robots' observed behaviors do not only depend on the software and hardware, but also on the surrounding environment. Evidently, even if two researchers buy the same robot and perform the same experiment with the same software, they are likely to observe very different findings. Another problem that has not helped in making the situation better, but that finds roots in the same issues, is the lack of a widely shared base of reusable code that is maintained and exploited by different research groups. While it is true that certain middle-ware softwares are enjoining significant popularity [1] [2], they are basically interfaces to gain portable access methods to different sensors and actuators. Code exchange for algorithms solving general purpose tasks is still a fairly rare occurrence. All of the above has contributed to a situation where quantitative comparisons between different approaches is still missing. As a consequence, it is still too often observed that when a new project is started, certain tasks are coded again from scratch, rather than relying on existing libraries. The situation described clearly is a detriment to the development of more capable robots.
Fortunately, in recent years a shift has been observed, and the direction seem to be changing. Among these events we list the following:

- the commercial success of certain robotic platforms has created a situation where many research labs use the same robots
- the establishment of on-line repositories of sensor data, like the Robotics Data Set Repository [3], allow different groups to execute different algorithms on exactly the same data set, thus outlining strengths and weaknesses
- the open-source approach is starting to become more accepted in the robotics community, with the creation of repositories of code for similar purpose [4]

In this paper we propose to start a systematic investigation and comparison of different algorithms for the well known simultaneous localization and mapping problem (SLAM). In particular, we present a juxtaposition between real world validation and experimental runs within the high fidelity USARSim simulator [5]. The use of a simulator is particularly appealing for quantitative comparisons of SLAM algorithms because it allows the generation of huge data sets without investing too many resources. Moreover, the simulator is highly configurable and allows one to specify different noise levels for the various sensors, thus permitting a careful evaluation of robustness with respect to noise sources.
The paper is organized as follows. In section II we provide an overview of the USARSim software, while in section III we illustrate and contrast three SLAM algorithms. Section IV reports on the experimental setup that was implemented and the results that were produced. Finally, conclusions are offered in section V.

II. THE USARSIM FRAMEWORK

The current version of Urban Search and Rescue Simulation (USARSim) [6] is based on the UnrealEngine2 [1] game engine that was released by Epic Games as part of Unreal Tournament 2004. The engine may be inexpensively obtained by purchasing the Unreal Tournament 2004 game. The USARSim extensions may then be freely downloaded from [7]. The engine handles most of the basic mechanics of simulation and includes modules for handling input, output (3D rendering, 2D drawing, and sound), networking, physics, and dynamics. USARSim uses these features to provide controllable camera views and the ability to operate multiple robots. In addition to the simulation, a sophisticated graphical development environment and a variety of specialized tools are provided with the purchase of Unreal Tournament.

The USARSim framework builds on the Unreal game engine and consists of:

- standards that dictate how agent/game engine interaction is to occur,
- modifications to the game engine that permit this interaction
- an Application Programmer's Interface (API) that defines how to utilize these modifications to control an embodied agent in the environment
- 3-D immersive test environments
- models of several commercial and laboratory robots and effectors
- models of commonly used robotic sensors

While there exists quite a few robotic simulators, USARSim was chosen for many different reasons, the most important of which being its accuracy. Indeed, a lot of time and research is spent every year improving the robotic platforms, authenticating the sensors, building additional robots, sensors, and environments, and validating the physics engine. More specifically, [8] [9] [10] [11] [12] provide details about USARSim validation both quantitatively and qualitatively. Additionally, USARSim provides the same robotic interface as the real P3AT, allowing researchers to run two robots, one in USARSim and one in the real world, with a single input (e.g. a joystick).

A simple but effective command-and-message interface is used to interact with the USARSim robots: string commands are sent to the robot and string messages are sent by the robot. The USARSim interaction standards consist of items such as robot coordinate frame definitions and unit declarations while the API specifies the command vocabulary for robot/sensor control and feedback. Both of these items have become the *de facto* standard interfaces for use in the RoboCup Rescue Virtual Competition which utilizes USARSim to provide an annual Urban Search and Rescue competition. In 2007 this competition had participation from teams representing 5 countries.

[1]Certain commercial software and tools are identified in this paper in order to explain our research. Such identification does not imply recommendation or endorsement by the authors, nor does it imply that the software tools identified are necessarily the best available for the purpose

Both laboratory and commercial vehicles with different mobile platforms (skid-steered, Ackerman-steered, omni drive, legged, humanoid, nautical, and aerial) are incorporated within USARSim. Additionally, a set of robotic arms and plan-tilt mechanisms can effortlessly be mounted and utilized on any robot. Figure 1 shows a small collection of some of the robots that USARSim has to offer. The list of available sensors and effectors is also quite extensive and includes range scanners, sonars, cameras, grippers, RFID tags, and INS sensors.

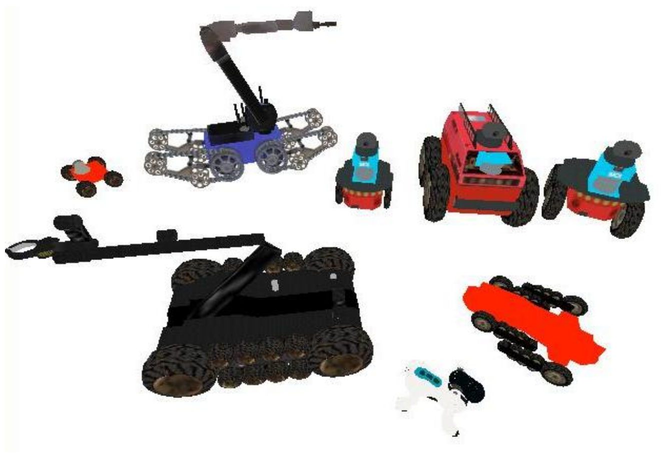

Fig. 1. A subset of available robots in the current version of USARSim.

Highly realistic environments are also provided with the USARSim release, ranging from simple planar mazes to multi-level collapsed structures. The environments encompass challenging robotic problems that cover many areas of research including mapping, planning, mobility, cooperation, communication, image-processing, and victim detection. Furthermore, the environments accommodate all the USARSim robotic platforms by providing indoor buildings, urban roads and highways, lakes and rivers, and large flying spaces. Example indoor and outdoor environments may be seen in Figure 2. In addition, an editor delivered for free with the game engine and the ability to import models simplifies the creation of worlds.

It is worthwhile to note that USARSim does not supply a robot controller. In other words, USARSim simply provides a well-defined interface to communicate with the robots and it is the researcher's responsibility to appropriately use the interface to achieve desired results. Researchers do not have to write a controller from scratch, however, since several open source controllers may be freely downloaded. These include the community developed Mobility Open Architecture Simulation and Tools (MOAST) controller [13], the player middle-ware [1], and any of the winning controllers from previous RoboCup competitions. Winning controllers, from RoboCup 2006, may be found on the robocuprescue wiki [14]. A description of the winning algorithms may be found in [15].

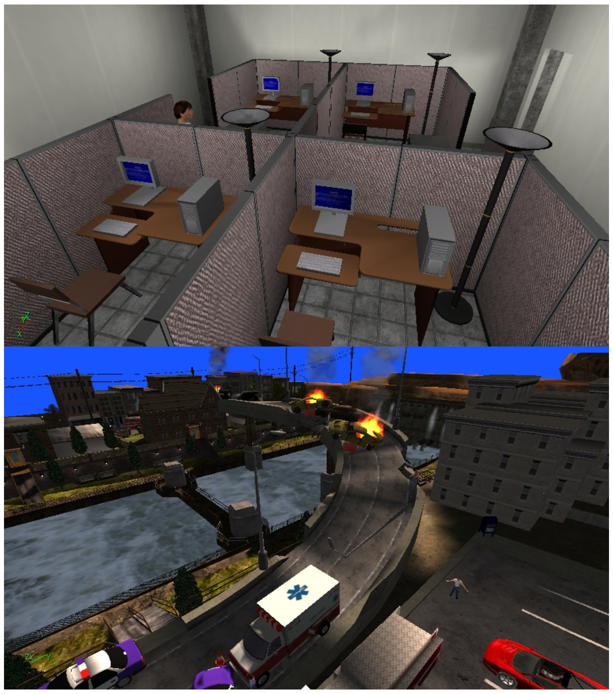

Fig. 2. Examples of indoor (top) and outdoor (bottom) environments used during the 2007 RoboCup Virtual Rescue Competition.

III. SLAM ALGORITHMS

In this section we briefly describe the three algorithms that were compared. We selected three of the seven packages currently available on the OpenSlam website. The selection was mainly driven by the desire to compare algorithms with similar characteristics in terms of requested input data, rather than by the desire to perform a comprehensive comparison.

A. GMapping

The GMapping algorithm produces a grid map and takes a particle filter approach [16] [17]. In particular, each particle is associated with a possible map. The main challenge for the algorithm, therefore, is to reduce the number of particles, because of the significant overhead associated with each particle. The algorithm uses a so-called Rao-Blackwellizzed filter and its major contribution is in the definition proposal distributions and resampling techniques that allow one to decrease the number of particles without incurring in problems related to undersampling.

The GMapping implementation available on OpenSlam is coded in C++ and processes data logs encoded using the Carmen log format [18]. Basically, the algorithm requires time-stamped odometry (pose, translational and rotational velocities) and time-stamped readings from the SICK laser.

B. GridSlam

GridSlam also uses a Rao-Blackwellizzed filter, and is particularly aimed to mapping environments with loops, a problem known to be challenging [19]. GridSlam also tries to decrease the number of particles used in the filter, but takes a different approach from GMapping. In GridSlam, a model of the residual error from scan registration is learned on the fly and used to contract the number of particles.

The GridSlam implementation is also coded in C++ and, similarly to GMapping, processes data provided in the Carmen log format.

C. DPSlam

DPSlam also uses a particle filter to estimate the robot's pose and maps [20]. However, unlike the former approaches that aim to carry along a restricted set of candidate maps, DPSlam exploits a peculiar map representation that allows one to track a huge set of candidates (thousands, according to the authors).

DPSlam is also implemented in C++ and requires the same data as the former algorithms (i.e. odometry and laser scans) although not encoded in the Carmen log format.

IV. EXPERIMENTAL SETUP AND RESULTS

The experimental setup aims to not only compare different SLAM algorithms, but also assess the fidelity of the simulation engine. In fact, if we are able to show that results extracted in the simulation environment can be safely extrapolated to real wold scenarios, we have then installed a very powerful tool to generate a massive amount of test data with minimal effort. The methodology developed to conduct this twofold evaluation will be described shortly. For real world validation, we use a P3AT platform. The robot is equipped with odometry sensors and a SICK PLS range finder. A wireless-capable laptop is mounted on the robot, and the robot is controlled using the Player middleware [1]. The robot used in simulation is the corresponding P3AT model available in USARSim. The simulated robot is also controlled using Player. Data collection for the real robot took place in the hallway of the School of Engineering of UC Merced. For the simulated experiments, we developed a model of the same building, using the original blue prints provided by the architects. Figure 3 shows matching screenshots, in simulation and the real world, of the robots collecting data.

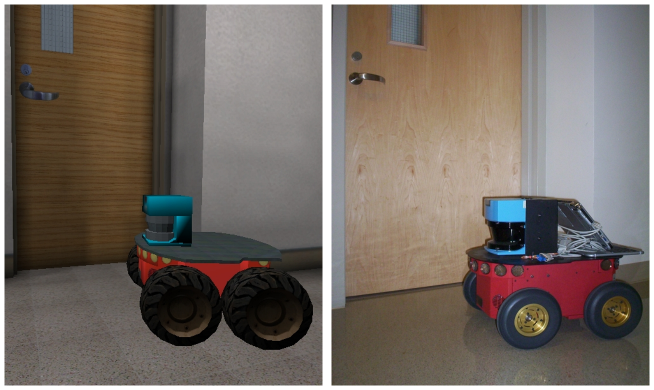

Fig. 3. Corresponding screenshots of the P3AT in simulation (left) and in the real world (right), as they are performing a data collection.

In order to keep a close alignment between the real world and its simulation, we avoided entering offices or research labs, especially for experiments aiming at assessing the simulation accuracy. The reason for this constraint stem from the lack of detailed footprints of lab and office furniture. Data collection was performed in parallel. A control application gets input from a user via a joystick and then sends the same commands, i.e. rotation and translation speeds, to the two robots. The user does not directly see any of the robots, but rather controls them by observing the output coming from the SICK sensor equipped on the real robot. The described approach requires careful tuning of the robot model in simulation in order to match the performance of the real platform.

We preliminary compared the performance of the various algorithms while processing data coming from the real robot and from the simulation. Figures 4 and 5 show the output of the GMapping algorithm for data collected by the real robot and the simulator, respectively. Figures 6 and 7 show the same results for the GridSlam algorithm. Finally, figures 8 and 9 show the same results produced by the DPSlam algorithm.

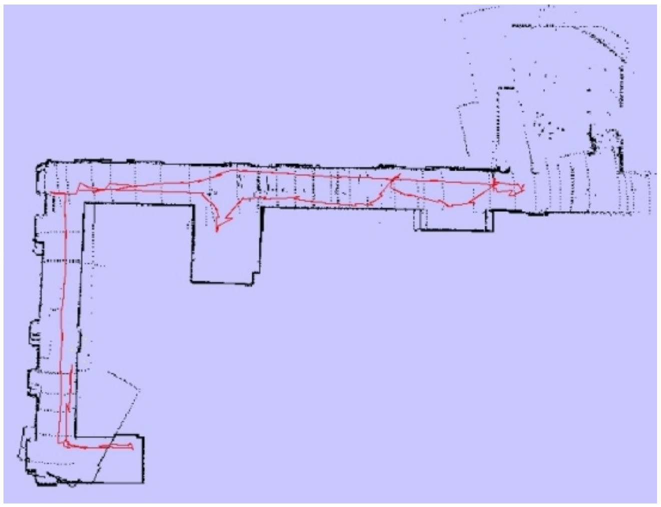
Fig. 4. A map produced by the GMapping algorithm on a data set collected by the real P3AT robot during daytime with people walking by the robot.

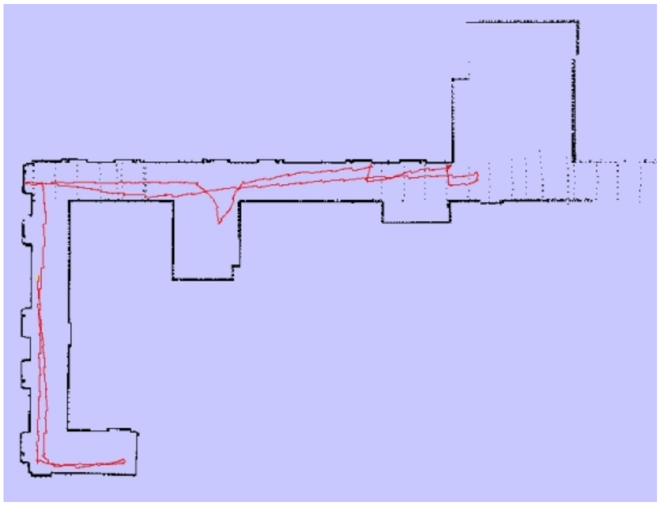
Fig. 5. A map produced by the GMapping algorithm on a data set produced by a simulated P3AT robot.

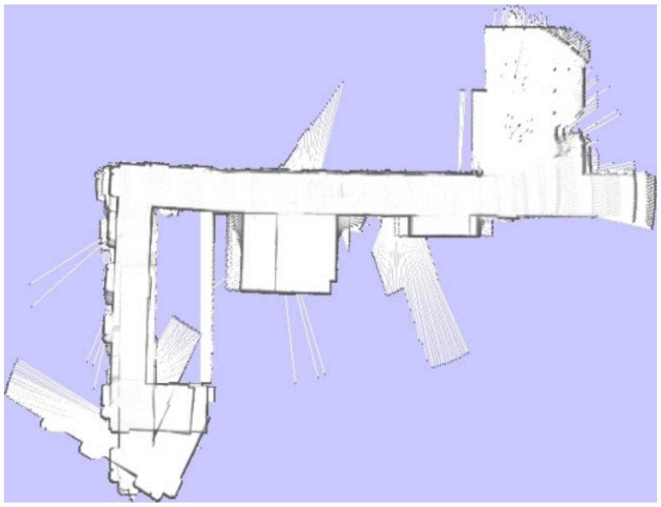
Fig. 6. A map produced by the GridSlam algorithm on a data set produced by the real P3AT robot.

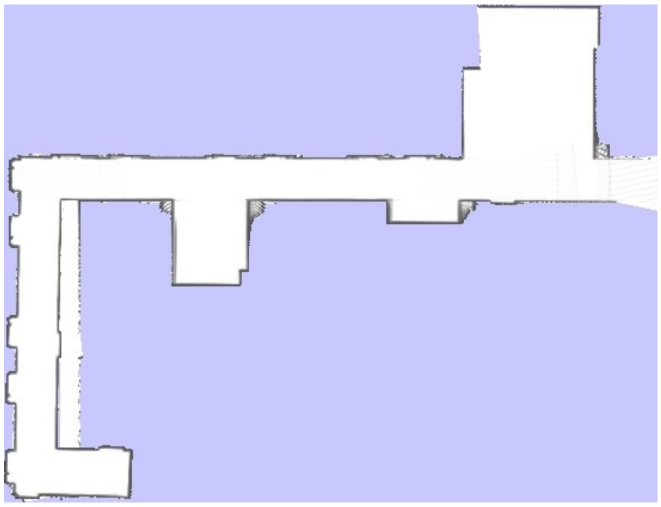
Fig. 7. A map produced by the GridSlam algorithm on a data set produced by a simulated P3AT robot.

It is important to stress that, in this set of tests, we did not strive to find the best fine tuning for the algorithms, but rather to assess the similarity between results produced in simulation and with the real robot. A few observations can be made from the figures:

- There is a good correspondence between maps produced in simulation and in reality by the GMapping algorithm. In both cases the produced map and the tracked path basically agree with ground truth.
- For the considered dataset, the DPSlam algorithm seems to fail both for real and simulated data. This does not mean that DPSlam is not working properly in general, but rather that both specific set of simulated and real

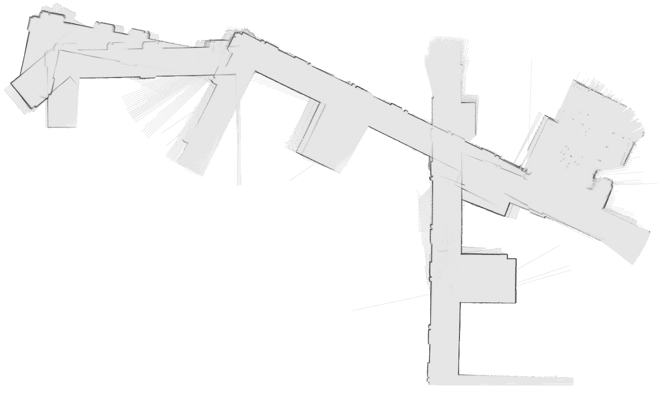

Fig. 8. A map produced by the DPSlam algorithm on a data set produced by the real P3AT robot.

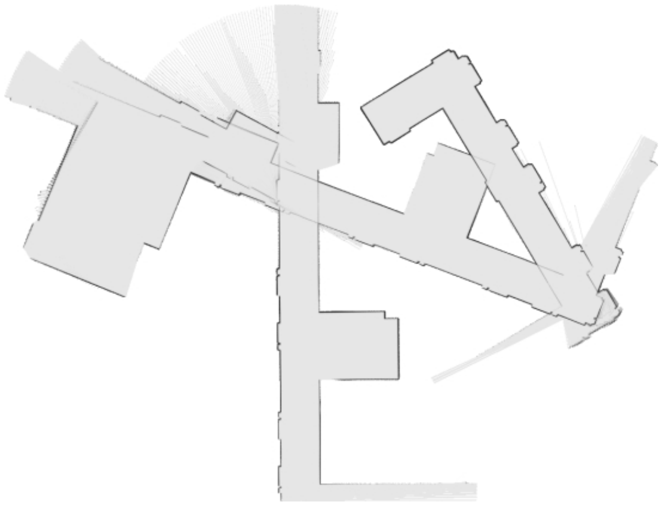

Fig. 9. A map produced by the DPSlam algorithm on a data set produced by a simulated P3AT robot.

data exhibit some characteristic hard to deal with for this algorithm. Successive runs show good performance.

- GridSlam algorithm exhibits an intermediate behavior. The map produced with real world data shows an inconsistency on the lower left corner and an incorrectly-handled opening in the horizontal corridors (the opening is much wider in figure 6). The map produced with data coming from the simulator does not show these problems, though there is a problem with the vertical corridor since it appears to be wider than it is in reality (its width should be the same as the horizontal corridor).

The above results illustrate that there is a reasonable correspondence between results produced with simulated and real world data. Such correspondence is fairly strong for GMapping and DPSlam (in terms of success or failure) but less evident for GridSlam.

The next set of tests aims to measure the robustness of the three algorithms with respect to signal noise. Previously illustrated runs were obtained under the following simulated conditions. Readings from the SICK laser were affected by an additive noise with intensity 0.1% while the odometry was affected by Gaussian noise with 0 mean and 0.1 covariance. It is understood that Gaussian noise is highly suboptimal when it comes to reproduce data coming from odometry, a fact that needs to be better addressed within the USARSim framework. We then executed the three SLAM algorithms on a data set produced by a simulated robot whose SICK laser was affected by an additive noise of intensity 5%. This means that every value d_i returned by the sensor is altered accordingly to the following formula

$$d'_i = d_i(1 + 0.05x)$$

where x is a random variable with uniform distribution over the interval $[-1, 1]$. Produced maps are illustrated in figures 10, 11 and 12.

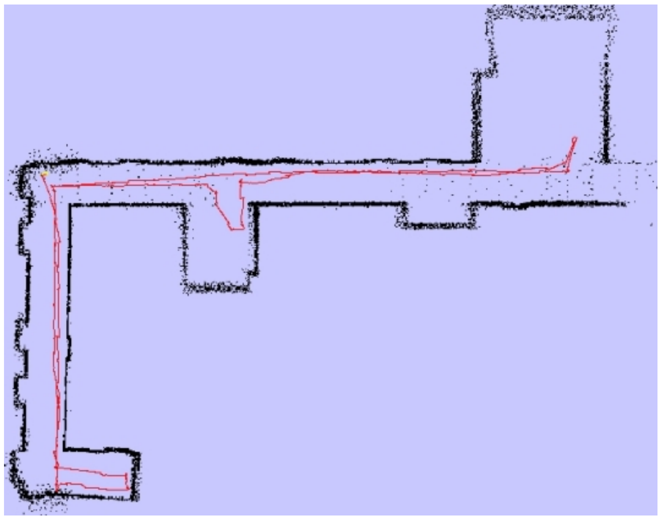

Fig. 10. A map produced by the GMapping algorithm on a data set produced by a simulated P3AT robot.

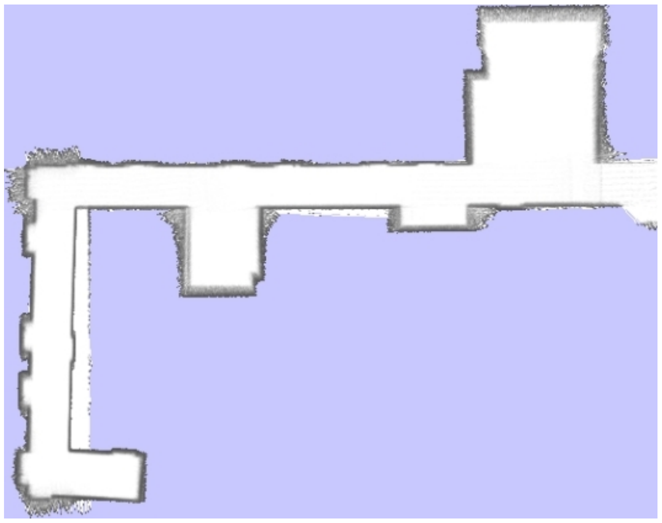

Fig. 11. A map produced by the GridSlam algorithm on a data set produced by a simulated P3AT robot.

The final set of tests was produced in a similar setting, with an additive noise of 10%. Resulting maps are illustrated in figures 13, 14 and 15.

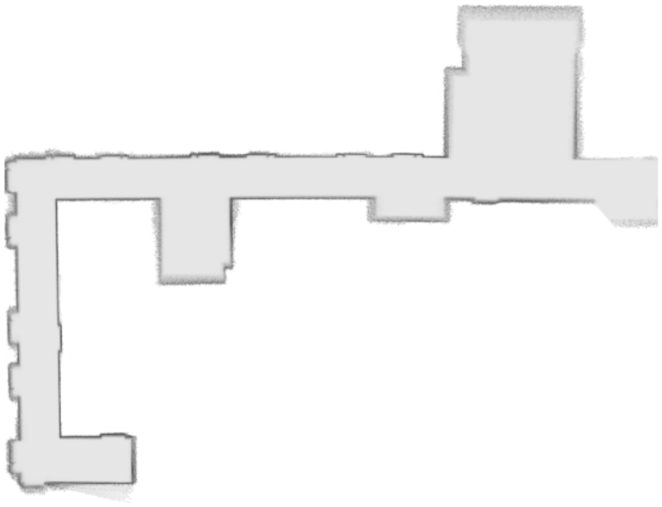

Fig. 12. A map produced by the DPSlam algorithm on a data set produced by a simulated P3AT robot.

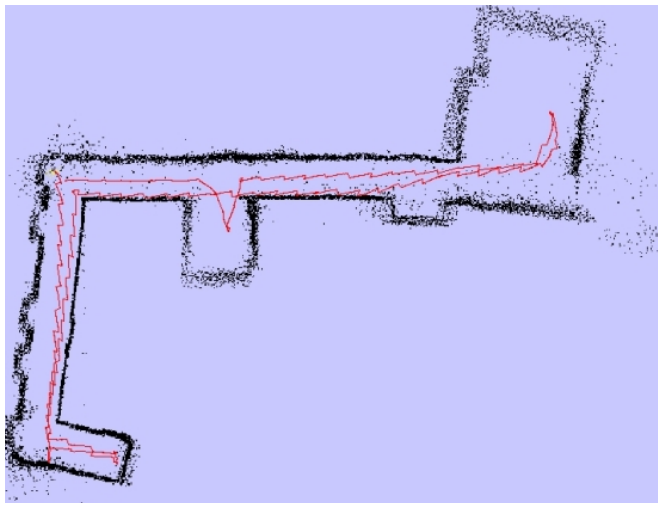

Fig. 13. A map produced by the GMapping algorithm on a data set produced by a simulated P3AT robot.

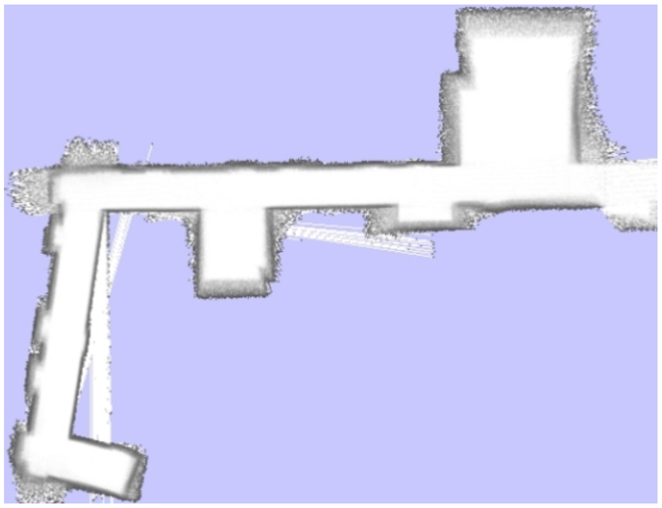

Fig. 14. A map produced by the GridSlam algorithm on a data set produced by a simulated P3AT robot.

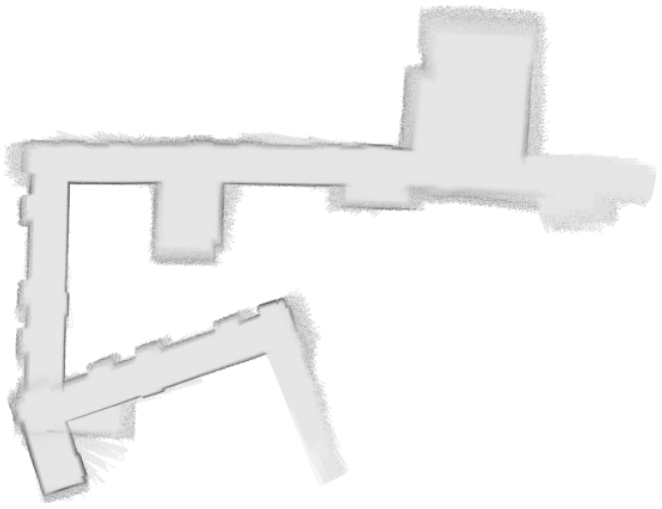

Fig. 15. A map produced by the DPSlam algorithm on a data set produced by a simulated P3AT robot.

V. Conclusions

Some general conclusions can be drawn. First, the use of the USARSim framework in order to compare different SLAM algorithms appears appropriate. Indeed, results obtained in simulation nicely translates to real robots. The performance of the simulated SICK laser is comparable to the real one, and an additive noise of 0.1% seems a reasonable choice. It appears necessary to develop a better model for odometry noise in order to accommodate the incremental nature of this disturbance. The three algorithms seem to be equally and reasonably robust to noise in the SICK laser. A level of 5% noise that visually appears much larger than anything observed in real world systems can still be dealt with by the algorithms. Both GMapping and DPSlam suffer from a lack of accuracy when noise is increased to 10%, but this is a level that is hardly ever observed in reality.

Comparing the different algorithms, GMapping emerged to be the more stable one in terms of performance, seldom incurring in severe problems of map consistency. Along a different line for comparisons, DPSlam required the most time when processing batch data logs offline and, as acknowledged by the authors, is very demanding in terms of memory. It is important to note that we have not fine-tuned the algorithms, resulting in a possibly unsatisfactory set of parameters. Different sets of parameters could have easily produced different results. However, if the robotics community aims to a wide sharing of open source algorithms, the availability of easy to use and tune algorithms is a must.

An aspect that still seems to be underconsidered is the availability of well defined metrics for mapping algorithms to, for example, quantitatively measure the correlation between a produced map and the corresponding ground truth. *Visual inspection* still seems to be the most widely used means of performing such evaluation, but a more rigurous approach is needed. Tools coming from the field of computer vision, related to image similarity, could be useful but have yet to enjoy significant popularity.

References

[1] "Player/stage project," http://playerstage.sourceforge.net, 2007.

[2] "Orca robotics," http://orca-robotics.sourceforge.net, 2007.

[3] Radish, "The robotics data set repository," http://radish.sourceforge.net, 2007.

[4] "Openslam," http://www.openslam.org, 2007.

[5] S. Carpin, M. Lewis, J. Wang, S. Balarkirsky, and C. Scrapper, "Usarsim: a robot simulator for research and education," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007, pp. 1400–1405.

[6] S. Balakirsky, C. Scrapper, S. Carpin, and M. Lewis, "Usarsim: providing a framework for multi-robot performance evaluation," in *Proceedings of the Performance Metrics for Intelligent Systems Workshop*, 2006.

[7] "Usarsim project," http://sourceforge.net/projects/usarsim, 2007.

[8] S. Carpin, T. Stoyanov, Y. Nevatia, M. Lewis, and J. Wang, "Quantitative assessments of usarsim accuracy," in *Proceedings of the Performance Metrics for Intelligent Systems Workshop*, 2006.

[9] S. Carpin, J. Wang, M. Lewis, A. Birk, and A. Jacoff, "High fidelity tools for rescue robotics: results and perspectives," in *Robocup 2005: Robot Soccer World Cup IX*, ser. LNCS, 2006, pp. 301–311.

[10] A. Birk, S. Carpin, W. Chonnaparamutt, V. Jucikas, H. Bastani, I. Delchev, I. Krivulev, S. Lee, S. Markov, and A. Pfeil, "The iub 2005 rescue robot team," in *Robocup 2005: Robot Soccer World Cup IX*, ser. LNCS. Springer, 2006.

[11] J. Wang, M. Lewis, M. Koes, and S. Carpin, "Validating usarsim for use in hri research," in *Proceedings of the 49th meeting of the Human Factors and Ergonomics Society*, 2005, pp. 457–461.

[12] M. Zaratti, M. Fratarcangeli, and L. Iocchi, "A 3d simulator of multiple legged robots based on USARSim," in *Robocup 2006: Robot Soccer World Cup X*, 2007.

[13] "MOAST project," http://sourceforge.net/projects/moast, 2007.

[14] "Robocup rescue wiki," http://www.robocuprescue.org/wiki, 2007.

[15] S. Balakirsky, S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, and V. Ziparo, "Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from robocup rescue," *Journal of Field Robotics*, To appear.

[16] S. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005, pp. 2432 – 2437.

[17] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping iwht Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 36–46, 2007.

[18] "Carmen," http://carmen.sourceforge.net.

[19] D. Haehnel, D. Fox, W. Burgard, and S. Thrun, "A highly efficient fastslam algorithm for generating cyclic maps of large-scale ennvironments from raw laser range measurements," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 206–211.

[20] A. Eliazar and R. Parr, "DP-SLAM: fast, robust simultaneous localization and mapping wuthout predetermined landmark," in *International Joint Conference on Artificial Intelligence*, 2003.