ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues

Conference Paper · May 2011

DOI: 10.1109/SIECPC.2011.5876943 · Source: IEEE Xplore

Safdar Zaman
COMSATS Institute of Information Technology
12 PUBLICATIONS **84** CITATIONS
SEE PROFILE

Wolfgang Slany
Graz University of Technology
123 PUBLICATIONS **1,047** CITATIONS
SEE PROFILE

Gerald Steinbauer
Graz University of Technology
94 PUBLICATIONS **491** CITATIONS
SEE PROFILE

ROS-based Mapping, Localization and Autonomous Navigation using a Pioneer 3-DX Robot and their Relevant Issues

Safdar Zaman, Wolfgang Slany, Gerald Steinbauer
Institute for Software Technology
Graz University of Technology, Austria
{szaman, wsi, steinbauer}@ist.tugraz.at

***Abstract*— The *Robot Operating System* (ROS) provides operating system-like services to operate robots. Mapping, localization, and autonomous navigation in an indoor environment are popular issues in the field of autonomous robots. Autonomous navigation in a dynamic environment is not only challenging but also uncovers many indoor environmental factors which affect the process of mapping and navigation. The presented work describes how a ROS-based control system is used with a Pioneer 3-DX robot for indoor mapping, localization, and autonomous navigation. Mapping of different challenging environments is presented in this work. Moreover, some factors associated with indoor environments that can affect mapping, localization, and automatic navigation, are also presented. For experiments, three environments (one artificial and two real) have been tested. Some implementation was done in C and Python.**

I. INTRODUCTION:

A robot does not possess natural senses like human beings have. Human beings get information about their surroundings through vision and other sensing powers. A robot cannot explore an unknown environment unless it is provided with some sensing sources to get information about the environment. Different kinds of sensors such as sonars, odometers, laser range finders, inertial measurement units (IMU), global positioning system (GPS) and cameras are used to make a robot capable of sensing a wide range of environments. The map of the environment is a basic need of a robot to perform indoor services like moving room to room, gripping and picking an object from one place and taking it to another place. To perform such type of services, the robot should not only know about the environment but while it is moving it should also be aware of its own location in that environment. Moreover, proper representation of the robot itself in the environment also plays a vital role to solve many issues related to automatic navigation.

A comprehensive overview of the Robot Operating System (ROS) has been presented by Quigley et al. [1]. Li et al. [2] presented map building using sonar sensors and Dezert-Smarandache Theory (DSmT). In this work the authors used computational techniques, such as probability theory, fuzzy sets theory, neutro-sophic theory, and neural networks for map building. They used a Pioneer robot for experiments and created a 3D grid map of the environment. Imthiyas [3] presented an indoor robot localization process. This work used sets of data collected from indoor environments to produce a Gaussian Process (GP) model. During motion, the robot uses this GP model to localize itself in the environment. The author also presented feature extraction process to extract and use geometrical features of the objects to perform obstacle avoidance. In [4] Jogan and Leonardis presented an approach for using omni-directional images for topological localization. Such image-based methods are an intuitive alternative to localization based on range-sensors and metric maps. The authors used principle component analysis to obtain an efficient representation of reference images using eigen-vectors. The matching of actual and reference images are based on the similarity of their eigen-vectors. In this way the method provides robustness against noise and occlusion in the image. The work presented by Thrun et al. [5] presented multi-robot mapping with Pioneer robots using an incremental map construction process. In [6] the authors presented an approach that uses a probabilistic method to build maps of populated environments. They used a joint probabilistic data association filter to track motion of the people within environmental data obtained through sensors of robot. They used this information to improve alignment between successive scans and to filter out corrupted measurements originated from people walking in the sensor's range. In [7] the authors presented a comparative study of robot localization processes. In this work monocular and trinocular cameras, and laser range finder sensor were used for mapping and localization. According to their experimental results the precision obtained by both the cameras (monocular and trinocular) is the same as the precision obtained using a laser rangefinder sensor. Please refer to [8] for a deeper discussion on basic methods for mapping with robots. A recent development for the SLAM problem is the Fast-Slam [9] where a particle filter is used and different possible robot paths and data associations are represented as samples for this filter. For a deeper coverage of the basic principles and methods used in localization and mapping please refer to [10].

II. ROS:

Robot Operating System (ROS)[1] is a Linux based software framework for operating robots. This framework uses the concept of packages, nodes, topics, messages and services.

[1]Please refer to http://www.ros.org for a complete description of ROS.

A *node* is an executable program that takes data from the robot's sensors and passes it to other nodes [1]. The information which moves from node to node is called a *message*. Messages always travel via special ports called *topics*. A node which sends messages on a topic is called a publisher and the receiving node has to subscribe the topic to receive that message, hence it is called a subscriber. All related nodes are combined in one *package* that can easily be compiled and ported to other computers. The packages are necessary to build a complete ROS-based autonomous robot control system.

III. Presented Work:

The presented work is a ROS-based control system for a Pioneer 3-DX robot for mapping, localization, and autonomous navigation in both real and self-created environments. The following sections describe the setup.

A. *ROS setup for Pioneer 3-DX:*

The Pioneer 3-DX robot is one of the most popular research robots. Because of its modest and balanced size combined with reasonable hardware, it is most suitable for in-door navigation. Pioneer 3-DX robots use a differential drive for locomotion. To make the robot fully capable of mapping and localization, one laser range finder (-SICK LMS 200) is also used. Linux (Ubuntu 10.4) based computer system equipped with ROS is used to control the robot. A joystick is also used to teleoperate the robot during the mapping process. Both Laser and Pioneer are connected with the computer via USB ports. At the software side, we have used ROS packages like *ROSARIA, SICKTOOLBOX_WRAPPER, GMAPPING, MAP_SERVER, AMCL, MOVE_BASE, JOY, TELEOP_BASE, RVIZ* and *SIMPLE_NAVIGATION_GOALS*. The *ROSARIA* package is offered by the AMOR[2] group. The nodes provided by these packages are *RosAria, sicklms, slam_gmapping, map_saver, map_server, amcl, move_base, joy_node, teleop_base, rviz,* and *simple_navigation_goals*. The following subsections describe which nodes are required at which stage.

1) Mapping: Mapping is the process of creating a spatial model of the environment surrounding the robot using its sensors. The map is then used for localization and navigation. In order to build the map using ROS, the following commands are executed on the Linux command line. Each command specifies a node and its package:

```
$ roscore
$ rosrun ROSARIA RosAria
$ rosrun joy joy_node
$ rosrun teleop_base teleop_base
$ rosrun ROSARIA transform
$ rosrun sicktoolbox_wrapper sicklms
$ rosrun mapping slam_gmapping
$ rosrun map_server map_saver -f mymap
```

[2]Please refer to http://act.rasip.fer.hr/ for AMOR group

The *sicklms* node should be provided with the proper port parameter. The *transform* is our own created node which transforms one coordinate system to another. All these nodes enable the robotic system to move for creating the map. The robot is moved in the desired environment via joystick. Odometry data from the Pioneer's motor is provided by RosAria node. Laser and odometry data are used by the slam_gmapping node during map creation. The node uses the open-source implementation of grid-based SLAM using Rao-Blackwellized particle filters [11]. After scanning the whole environment it will create two files *mymap.pgm* and *mymap.yaml*. The *PGM* file is the map's image whereas the *YAML* file is the description of this map. The *YAML* file is later used in localization and navigation. Figure 1 shows a map created during mapping:

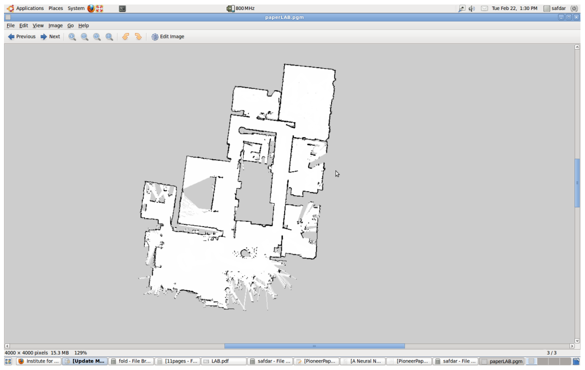


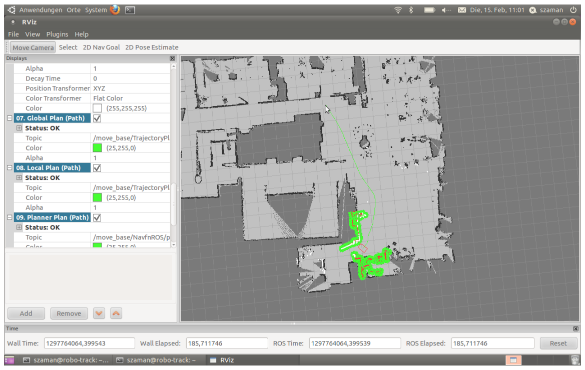
Fig. 1. Map of a real dynamic environment 1

2) Localization: Localization tells the robot where it is in relation to the environment. Localization uses odometry, laser data, and a map. In addition to the nodes **roscore, RosAria, transform, joy_node, teleop_base,** and **sicklms,** the following nodes are also necessary for robot localization:

```
$ roslaunch amcl amcl_diff.launch
$ rosrun map_server map_server mymap.yaml
$ rosrun rviz rviz
```

These nodes enable the robot to localize itself in the environment. The node provided by the *Amcl* package does this localization. The node uses the particle filter based localization method described in [12]. To control the computational demands of the method an automatic adaption of the sample size based on KLD sampling is used [13]. The launch file *amcl_diff* is launched from the *examples* directory of the amcl package. Localization needs the *map_server* node instead of *map_saver* from the package *map_server* for getting the mymap.yaml file. Node *rviz* is used for visualization of the robot localization. Figure 2 shows the self-localized Pioneer robot in the map.

3) Autonomous Navigation: Once mapping and localization are successfully done then navigation can be easily achieved. Package *move_base* is used to accomplish autonomous navigation. This package provides the *move_base* node that uses localization information and



```
Displays
Alpha                   1
Decay Time              0
Position Transformer    XYZ
Color Transformer       Flat Color
Color                   (255,255,255)
07. Global Plan (Path)  ✓
Status: OK
Topic                   /move_base/TrajectoryPl
Color                   (25,255,0)
Alpha                   1
08. Local Plan (Path)   ✓
Status: OK
Topic                   /move_base/TrajectoryPl
Color                   (25,255,0)
Alpha                   1
09. Planner Plan (Path) ✓
Status: OK
Topic                   /move_base/NavfnROS/p

Wall Time: 1297764064,399543   Wall Elapsed: 185,711746   ROS Time: 1297764064,399539   ROS Elapsed: 185,711746
```

Fig. 2. Localization

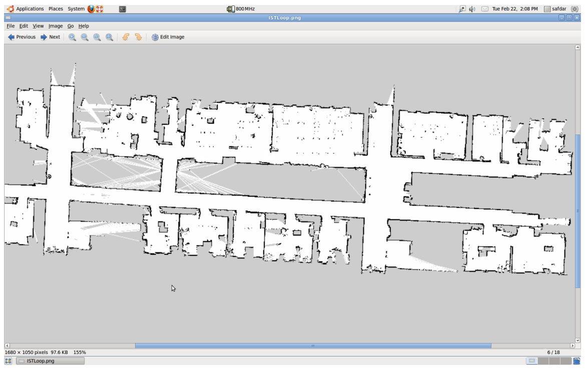


Fig. 3. Map of a real dynamic environment 2

provides move-commands to the robot to move safely in the environment without colliding with stationary or moving objects like humans. This package also maintains two costmaps each for the local and global planner. The approach used for navigation is described in more detail in [14]. The global planner is based on A*-search while the local planner used the dynamic window approach by Fox et. al [15]. Figure 2 also shows the RVIZ tool during robotic autonomous navigation. After executing the nodes **roscore, RosAria, transform, joy_node, teleop_base, sicklms, amcl, map_server,** and **rviz**, we use one more launch file from package *move_base*. This launch file launches the *move_base* node with four yaml files as its parameters. These yaml files specify the robot's footprint, angular and linear velocities limits, different tolerances, and biases, etc. The file is launched as:

$ roslaunch move_base move_base.launch

Four *yaml* files used by move_base package are for the common cost parameters, local costmap, global costmap, and base local planner. These files are used for path planning to help autonomous navigation. Rviz is used as an interactive tool to set the initial pose and goal to navigate the robot to a specific location in the environment. The *Simple_Navigation_Goals* package provides a way to set a sequence of the goals. We created a Python node using the ROS concepts of *Action Server* and *Action Client* for providing sequence goals to navigate the robot automatically.

B. Experiments:

Two real and one self-created environments have been used for experiments. Figures 1 and 3 show the maps of two real dynamic environments. These environments consist of a big hall, many rooms, hallways, different sized doors, and glass walls. The artificial environment shown in Figure 4 is a self-created environment to check some other complicated structures like narrow passes, edges, zigzag ways, and sharp corners, etc. Its map is shown in Figure 5. During mapping and navigation, these environments raised some problems due to factors such as different level of surfaces or orientation of the laser sensor.

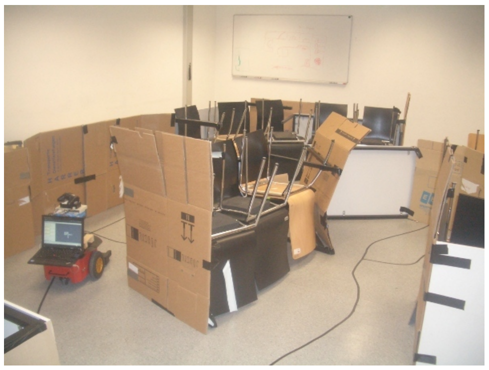


Fig. 4. Self-created artificial environment

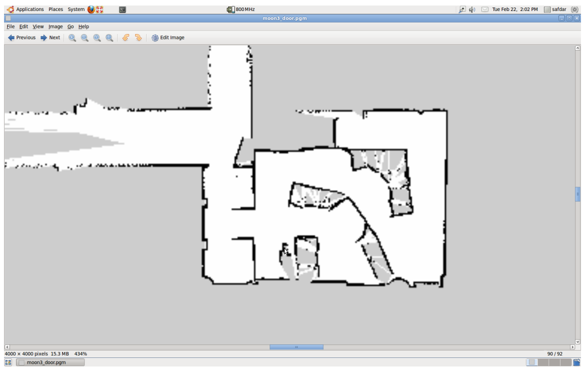


Fig. 5. Map of the self-created artificial environment

C. Issues and Observations:

During this whole process, some aspects of the environments have been detected that affected indoor mapping and autonomous navigation. These aspects are related to both the environment and the robot. In order to have precise mapping and navigation, these factors should be given proper attention. These issues include:

Height and Orientation of the laser sensor:
The position of the laser scanner on the robot is very critical to mapping and navigation. Of course sensors are always mounted on the robot at some height above the ground.

One should carefully select the height and orientation of the laser sensor as these are very critical to mapping and navigation. We mounted the laser sensor on the robot in a straight forward direction with 390mm height above the ground. In such situations all the objects below the laser scan become undetectable during mapping and navigation. The map obtained can be incomplete and hence the robot can collide with objects not recognized by the laser.

Robot Footprint:

The robot's body is represented as its footprint in the map as shown by the small red rectangle in Figure 2. The footprint is used in the navigation to estimate a collision-free path through the environment. The robot's footprint is always specified in such a way that it should not intersect with any point representing objects in the map. Circular and complete square footprints are more suitable than the rectangular ones. We applied both circular and square footprints that gave pretty good results to avoid collision with the edges and sharp corners in the environment. By adding additional hardware to the robot such as a gripper we have to enlarge the footprint. In this situation circular and square footprints become larger than hallways and doors. The rectangular footprint of the robot was suitable but it caused many delays and problems in the navigation at the edge and corner points of the environment. Hence the choice of footprint does not only depend upon the robot's shape but the structure of the environment is also important.

Surface Difference or Inclination:

This factor specially affects the mapping procedure. As shown in Figure 6, the robot bends down due to the surface

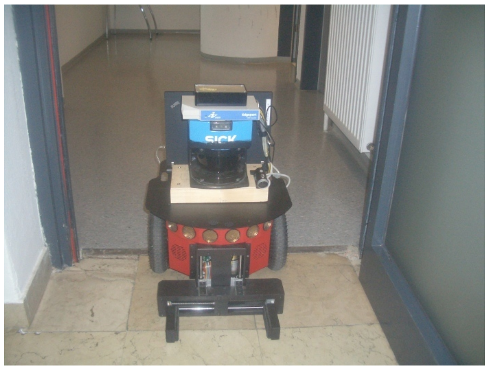


Fig. 6. Surface difference on the floor

difference and the laser beam gets disturbed. Therefore, if the environment contains a surface height difference or inclination on the floor then the laser beam can be disturbed during motion and the obtained map will have structural irregularities. Such types of surface difference are mostly found in doors connecting rooms. A map affected by this problem needs to be edited and aligned by putting black lines where necessary to add structures not automatically recognized during the mapping process.

Objects with wider lower part:

The Objects whose lower part is more expanded than the upper may cause problems during mapping and navigation. Such expanded area is not visible from a certain height of the laser sensor. During path planning the planner may generate a path through this area and the robot can hit this lower expanded area of the object. Figure 7 shows a black seat and a whiteboard whose lower parts are wider and more expanded than their upper detectable area. In such situations

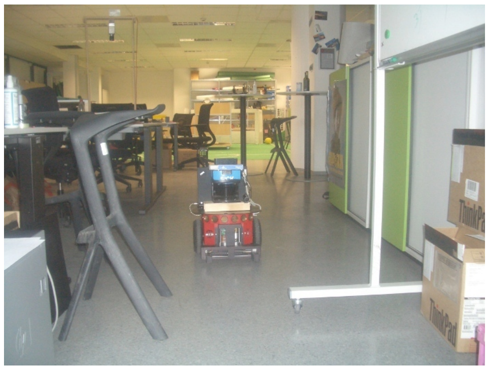


Fig. 7. Objects wider from below area

the robot believes that there is sufficient area to pass through but in reality is going to collide with them. The solution to this issue is either that the laser sensor should be mounted in such a way that it can see such surfaces, or black points should be drawn in the map resembling invisible objects in order to avoid collisions.

Looping during mapping:

Mapping can provide better maps if the robot scans the environment in a loop. A loop is an environmental closed structure where one starts moving from a point and comes back at that point through different way. Such a situation was faced while mapping two rooms having a connecting door. Loop closing methods such as described in [11] are quite complex but can increase the quality of maps significantly.

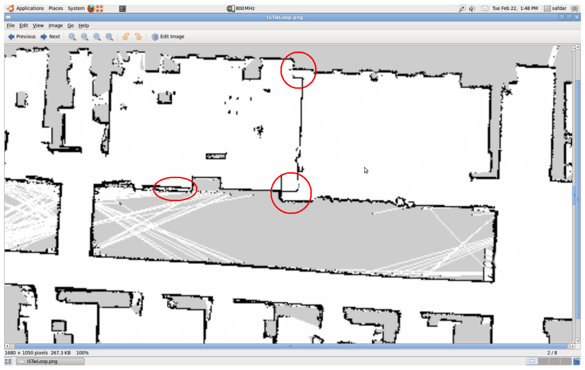


Fig. 8. Map without taking a loop. Misalignment of the map is highlighted.

Figure 8 shows an inaccurate map created without taking the loop through rooms having a closed door in between them. But with the opened door and taking the loop, the problem in the map disappeared as shown in Figure 9.

IV. Discussion & Future Work:

The presented work describes a ROS-based control system of a Pioneer 3-DX robot for mapping and navigation in

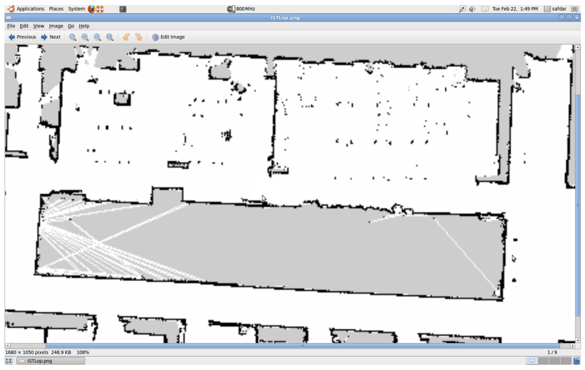


Fig. 9. Map after taking and closing a loop

indoor environments. The maps of two real and one artificial environment are created and tested. A navigation stack comprising different ROS packages is used for autonomous navigation. During mapping and navigation different factors like surface difference, object widening at its lower parts, sensor orientation or loops are also presented. In our future work we will map more dynamic environments to find out more such factors and structures. It is also recommended to look for runtime and fast strategies to cope with these factors without changing the real map of the environment.

ACKNOWLEDGEMENTS:
The first author gratefully acknowledges the support from Higher Education Commission (HEC) of the government of Pakistan funding his PhD studies at Graz University of Technology in Austria.

REFERENCES

[1] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[2] X. Li, X. Huang, and M. Wang. Robot Map Building From Sonar Sensors and DSmT. *Information & Security. An International Journal*, 20:104–121, 2006.

[3] M. P. Imthiyas. Indoor Environment Mobile Robot Localization. *International Journal on Computer Science and Engineering*, 2(3):714–719, 2010.

[4] M. Jogan and A. Leonardis. Robust localization using an omnidirectional appearance-based subspace model of environment. *Robotics and Autonomous Systems*, 45(1):51 – 72, 2003.

[5] S. Thrun, W. Burgard, and D. Fox. A Real-Time Algorithm for Mobile Robot Mapping with applications to filter out corrupted measurements originating from people walking in the vicinity of the robot. In *IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, USA, 2000.

[6] D. Hahnel, D. Schulz, and W. Burgard. Map Building with Mobile Robots in Populated Environments. In *IEEE/RSI, International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, 2002.

[7] J. A. Perez, J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardos. Continuous Mobile Robot Localization: Vision vs Laser. In *IEEE International Conference on Robotics and Automation (ICRA)*, Detroit, MI, USA, 1999.

[8] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[9] M . Montemerlo and S. Thrun. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics (Springer Tracts in Advanced Robotics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[10] S. Thrunn, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. Intelligent robotics and autonomous agents. The MIT Press, 2005.

[11] G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23, 2007.

[12] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128(1-2):99–141, 2000.

[13] D. Fox. Adapting the Sample Size in Particle Filters Through KLD-Sampling. *International Journal of Robotics Research*, 22, 2003.

[14] E. Marder-Eppstein, E. Berger, T. Foote, B. P. Gerkey, and K. Konolige. The Office Marathon: Robust Navigation in an Indoor Office Environment. In *International Conference on Robotics and Automation (ICRA)*, Anchorage, AK, USA, 2010.

[15] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics and Automation*, 4(1), 1997.