Integrating ABSYNTHE autonomous navigation system into ROS

Ángel Llamazares, Eduardo Molinos, Manuel Ocaña
and Fernando Herranz
Department of Electronics. University of Alcalá, Madrid (Spain)
Email: allamazares, emolinos, mocana, fherranz@depeca.uah.es

***Abstract*— ABSYNTHE, which stands for Abstraction, Synthesis, and Integration of Information for Human-Robot Teams, is an interdisciplinary project which aims to develop basic concepts and structures for the abstraction of partial views of the environment and the actions and intentions of teams, as well as the integration of this information into situation assessments. One of our key objectives with this project is to develop the autonomous navigation system to be used in different platforms (big all terrain outdoor and small indoor robots) that can contain a variety of heterogeneous sensors. It represents a challenging topic because we have to develop a robust and safety navigation system that can be used by these different robots. The goal of this paper is to show the integration of the navigation system into Robot Operating System (ROS) platform. Some experimental results and conclusions will be presented.**

I. Introduction

Robots teams are being developed to support a variety of human activities, including planning and decision making. In recent years, these teams have started to include humans as active components that sense, act, and collaborate with their automated counterparts. These human-robot teams are characterized by their ability to acquire vast amounts of data, generally heterogeneous in nature, reflecting the variety of available member platforms and sensing devices. These data must be properly processed, analysed, interpreted, and fused to ensure the right functioning of human-robot teams in terms of coordination, negotiation, distribution, and cooperation.

The competency of humans to collaborate on the solution of complex problems, while exchanging information that facilitates the understanding of system behaviour, may be explained by the use of cognitive tools that produce summarizations and descriptions of complex objects, events, and relations in terms that are easy to comprehend by other humans. In addition, achieving an effective collaboration of human and robots in mixed human-robot environments, is especially important to ensure a robust navigation for complex tasks, such as rescue and guiding, in terms of ensuring human integrity.

The autonomous navigation systems have been studied deeply in the literature [1] [2]. To achieve an autonomous navigation, the system should take into account the following stages: perception, localization, cognitive processes and motors modulation. While the first one uses the on-board sensors to provide information about the surroundings of the robot, the localization stage is usually based on this sensory information and a priori map to determine the position. Once it has been determined the robot position, the cognitive process stage determines what things are needed to do to ensure that the robot gets the target in a safety way. Finally, cognitive processes will provide the output for the motors modulation stage.

When the robot is navigating in outdoors, usually Global Positioning System (GPS) or its enhanced version Differential GPS (DGPS) [3] are used in the localization stage because they provide enough accuracy (a few centimetres in DGPS case). On the contrary, when GPS receiver is in urban environments with high buildings or trees, the signal can suffer multipath fading or even Line-Of-Sight (LOS) blockage. In addition, it is important to remark that GPS signal is not strong enough to penetrate inside tunnels or buildings, then this problem discards this technique in indoor environments.

To solve this kind of problem, robot navigation systems use a combination of a previous map with perception information that comes from a sensor fusion [4] [5] to guide the robot. Maps are usually obtained in a semi-autonomous process known as mapping [6] [7].

Localization and mapping are two processes with similar features and a common problem. It is not possible to build a map if the localization process does not work well, and it is impossible to locate a device with high precision without an accurate map. Some many times, SLAM (Simultaneous Localization And Mapping) techniques are used to fix this problem doing both processes simultaneously and using a variety of sensors, such as GPS and vision [4], vision and laser [8] even using RF range only sensors [9] [10].

While the map, generated by mapping or SLAM algorithms, can be used by the global planner to reach the target, if we want to get a robust and safety navigation it is needed to include an obstacle avoidance system. It ensures that the robot always navigates avoiding obstacles and keeping robot-human integrity [11] [12]. Obstacle avoidance systems can be divided into global or local, depending of the available information about the environment. While the first ones assume a complete model of the environment such as potential field methods [13], local methods can be considered faster and can be programmed like reactive tasks. These reactive methods control the robot when an obstacle is detected to avoid the collision. They use the nearest portion of the environment and update the world model according to the current sensor observation.

In this work, we review some several available navigation, mapping and localization packages in ROS. We describe our own algorithms and how we have integrated them into this platform. We test the combination of several algorithms to achieve a robust and safety autonomous navigation using heterogeneous robots and sensors. Real and simulated experiments are developed to test the performance of these systems.

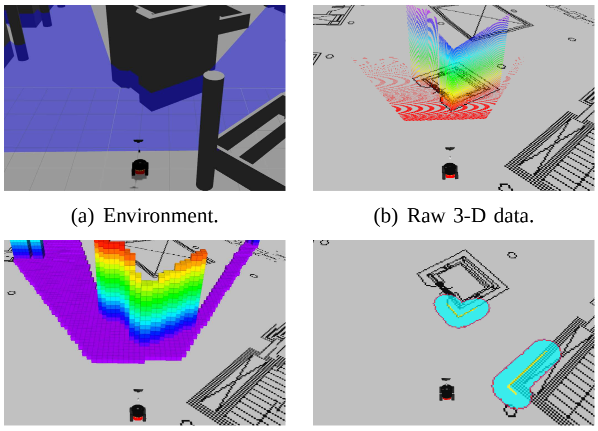

(a) Environment. (b) Raw 3-D data. (c) Octomap. (d) Costmap.

Fig. 1. From perception to mapping

In addition, we test the effect of dynamic obstacles on our navigation system.

The rest of the paper is organized as follows: section 2 shows a selection of related works; proposed method is shown in section 3; section 4 describes the test-bed and some experimental results; and finally, section 5 enumerates conclusions and future works.

II. Related works

This section provides a description of available packages and some previous algorithms developed by the authors that have been integrated in ROS platform. We describe a briefly introduction about several algorithms related to the next stages: mapping, localization and navigation.

A. Mapping stage

The mapping is used to obtain a model of the real world based on the data provided by the sensors. These maps can be global or local depending of the knowledge about the actual robot position (absolute or relative). The following methods have been evaluated in this paper:

- Octomap [14] builds a probabilistic 3D representation of the world based on Octree structures using a Point Cloud data. This Point Cloud can be directly obtained from 3D range sensors (3D laser ranger or Kinect depth camera) or using 2D range sensors (laser or sonar) and the appropriate information about the pose of the robot. Octomap is available in ROS under BSD License.
- Costmap [15] obtains a probabilistic 2D representation based on occupancy grids. It needs 2D range data or 3D Point Cloud projected to the floor. It is also able to enlarge any occupied cell to a specified radius, which is recommended for safety navigation purposes. It is available in ROS under BSD License in the navigation package.
- DOMap (Dynamic Occupancy Mapping) [16] is a dynamic occupancy grid where it is stored the probability of occupation of each cell and an estimation of object's velocity. While the occupancy is obtained by a Bayesian Occupancy Filter (BOF), the velocity estimation is based on a Kalman Filter tracking and a detection of the obstacles movement using the pyramidal implementation of Lukas Kanade optical Flow [17]. The algorithm is able to obtain not only a more robust position of the obstacle but also its velocity. This algorithm was developed by the authors and has been integrated into ROS.

Figure 1 shows an example of mapping using Octomap and Costmap. The environment 1(a) is perceived by a depth camera and in this way the 3D range raw data 1(b) is obtained. Using this Point Cloud, Octomap obtains the map shown in 1(c) and projecting this Point Cloud to the ground Costmap obtains the map shown in 1(d).

B. Localization stage

Global localization is the process that provides the position of a robot in a map, while local localization is relative to a starting position and increments the position of the robot by mean of integrating the odometry information. This is obtained by the encoders of the robot or by fusing information from encoders with Inertial Measurement Unit (IMU). A package to fuse this information using an extended Kalman filter is available in ROS (Robot_Pose_EKF) and can use odometry, IMU or even GPS (if available) to improve the localization of the robot.

If there is a priori good map of the environment it is possible to use an Adaptive Monte-Carlo Localization (AMCL) system [18] to obtain the global position of the robot in the map. This package is available in ROS under BSD License in the navigation package. On the contrary, if a map is not available, it is also possible to perform the Localization and Mapping stages simultaneously to obtain a map and the localization of the robot at the same time. SLAM processes can be found in the ROS package GMapping [7].

C. Navigation stage

The navigation stage must be divided into global and local stages. While the first one obtains the best path to get the target, the second one tries to avoid the possible obstacles of the route.

Global navigation calculates the best path from one point to another that the robot should be able to follow. This stage can be tackled by the Dijkstra algorithm [19] to calculate the best path into a static map given. Dijkstra is a graph search algorithm that solves the single source shortest path problem.

The local navigation controls the robot velocity to follow the path created by the global navigation. In addition, it avoids the obstacles found in the environment that are not included in the static map. There are so many algorithms to tackle this stage, some examples of them:

1) Dynamic window approach [20] tries to deal with the dynamic of the robot in a real world. This algorithm builds a search space of possible velocities without collision and search the best velocity to follow. It is available in ROS under BSD License in the navigation package.

2) Elastic band approach [21] fuses control and navigation into the same stage. It builds a map of interconnected spheres between the robot and the goal that can adapt their volume to the environment and selects the best path to travel from one sphere to another. It is available in ROS under BSD License in the navigation package.
3) VFH+ (Vector Field Histogram Plus) [22] is a reactive only algorithm. It was developed to use with laser sensors and divide the environment into angular sectors and the choose the best sector to navigate according to a cost function with different parameters such as the angular difference from the actual position or distance to the goal. Smooth Nearness Diagram [12] also divides the environment into angular sectors but in this case can have different sizes. This algorithm maximizes the security and usually navigates at the middle of the best sector selected by a cost function. Both algorithms were released under BSD License for the Player Stage Project and were integrated in ROS by the authors.
4) Curvature Velocity Method [23] works with velocity states instead of position states. This algorithm builds all the curvature arcs that the robot can follow and selects the best arc to follow based on a cost function that depends on parameters such as free distance to obstacle or angular difference to the orientation of the robot. Lane Curvature Method [24] and Beam Curvature Method [11] are two extensions of the previous method that add a previous stage where the environment are divided (into rectangular lanes or angular sectors) and select a middle goal based on another cost function that CVM should reach. These algorithms were first released under Carmen Project and they have been integrated in ROS by the authors.

One of the main problems of these methods is that most of them do not take into account the dynamic information of the environment, they only consider that all obstacles are static. This characteristic is especially important when the robot deals with a high uncertainty over the position, shape and velocity of the obstacles and it is still a challenge for real world applications [25] [26]. In addition, these approaches only take into account the two-dimensional information and do not take care of the height of them.

III. Proposed Method

In this section we show our proposal of autonomous navigation avoiding dynamic obstacles into the ABSYNTHE project framework. We focus on an improvement of the local mapping stage to tackle with dynamic obstacles using DOMap, as it has been presented before.

The system is divided into several stages which are connected with each other as Figure 2 shows. It is important to notice that the main goal of our system is to navigate through an environment that is composed of human-robot teams. Therefore, the robot have to reach the goal introduced by a human on a mobile device application and it has to do autonomously and safely, keeping the integrity of both human and robots. Figure 2 shows the stages involved in the navigation and the data flow between them.

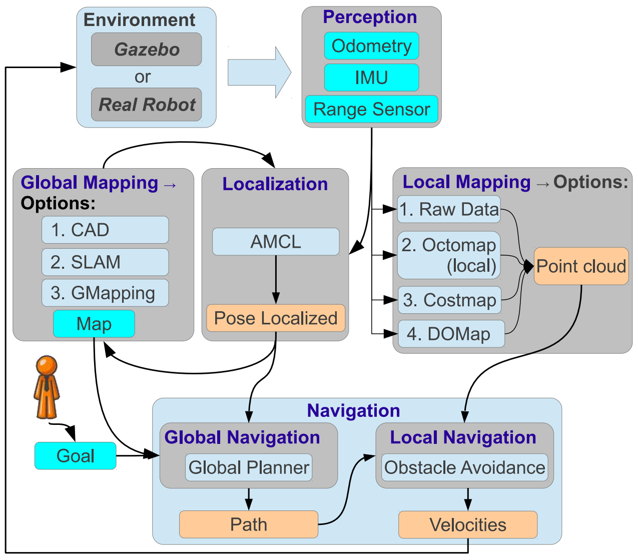


Fig. 2. ABSYNTHE's autonomous navigation flow diagram

In this work we are going to test the combination of obstacle avoidance algorithms with the four local mapping methods: raw data from the sensors, Octomap based mapping, Costmap and DOMap. In order to do that, we process the data provided by the mapping systems to give the same type of input to all avoidance algorithms. We have selected a Point Cloud structure because it represents 3D information but easily can be modelled like 2D information. The 3D information will be:

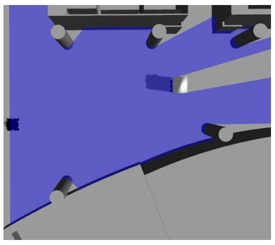
1) Raw data directly obtained from a 3D range laser or a depth camera. Also it can be obtained from a 2D range sensor and transform into 3D if the position of the sensor into the robot is known.
2) Costmap and Octomap divides the environment into cells (bi-dimensional or three-dimensional) so we transform the centre of each occupied cell into a 3D point.
3) DOMap also divides the world into cells of regular size, but instead of transforming occupied cells into a point, we transform the future estimation of the occupied cells in order to predict the movement of dynamic obstacles. Figure 3 shows how the environment is clustered into different objects, their relative velocities to the robot calculated and a Point Cloud built from this information.

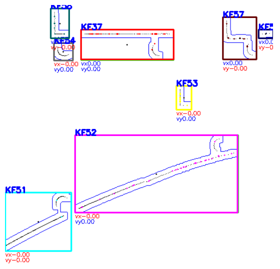
IV. Implementation and Results

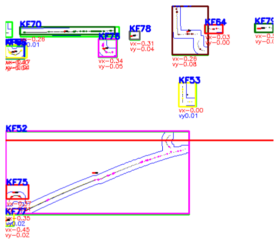
This section describes some implementation features and the experimental results obtained with the designed tests. Firstly we describe the test-bed and the platforms that we are using to test our proposal.

A. *Test-bed*

The environment to test our system was established indoor. We use several robots with different capabilities to test the whole system: Pioneer 3-DX, Pioneer 3-AT and a big outdoor


(a) Environment.


(b) DOMap: Static environment.


(c) DOMap: Dynamic environment.

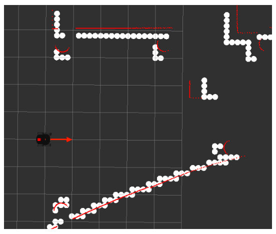

(d) DOMap: Output.

Fig. 3. DOMap: Perception stages. Fig. 3(a) Show the environment. Fig. 3(b) Show how the objects are detected, classified in blobs and tracking with Kalman Filters. Fig. 3(c) Show how the velocities is obtained and the Fig. 3(d) show the output of the system with the prediction poses of the objects.

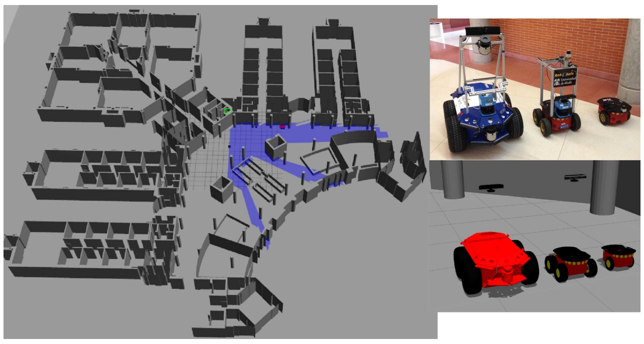


Fig. 4. Environment and robots used

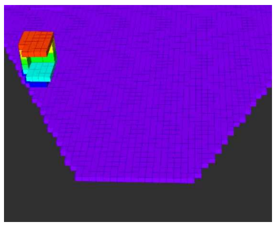
robot Seekur Jr., all of them made by MobileRobots. The in-door environment dimensions are approximately 60x60 meters. Figure 4 shows the environment and the different robots that has been modelled into Gazebo/ROS for simulation and test purposes.

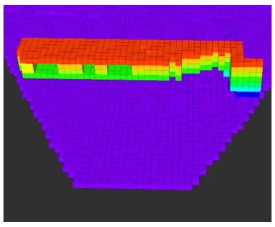
The platforms are equipped with different configurations: Pioneer 3-DX has 6 sonar and a Hokuyo URG-04LX laser parallel to the ground; Pioneer 3-AT is equipped with 12 sonar rangers, a Sick LMS 200 laser parallel to the ground, a Hokuyo URG-04LX laser that can be used parallel or angled to the ground, a Colibri IMU attached to the laser to obtain its attitude, and a Kinect sensor that can provide 3D information; Seekur Jr. is equipped with two SICK LMS 151 laser range finders, one of them angled to the ground, bumpers, and an IMU to reduce the uncertainty of the odometry. The control of each robot is made from a laptop with Ubuntu 12.04 LTS distribution and ROS Hydro. We also have a database server to provide the different targets and tasks that the robots should execute.

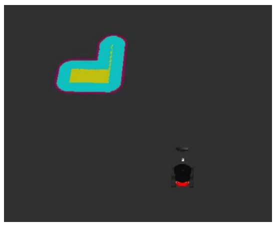
B. *Results*

We have tested different scenarios: local mapping, local obstacle avoidance and global navigation.

1) Local Mapping: Octomap and Costmap present some issues in the map building when moving obstacles appear due to these two methods uses a similar system to mark a cell as occupied or free: if there is a number of sensor impacts into a cell it is marked as occupied, and to mark this cell again as clear an impact behind this cell is needed. Figure 5 represents an obstacle moving perpendicularly to the robot and an obstacle getting closer to it, as there are no walls behind the obstacle in the field of view of the sensor, each cell marked as occupied can not be marked again as free. This problem produces a trail of the object that marks as occupied cells that are really free and it can affect to the local navigation. Due to the DOMap decreases the occupancy probability of the cells without laser impacts, and the lack of memory of Raw Data, they have not these issues.


(a) Octomap: Obstacle begins to cross.


(b) Octomap: Obstacle ends to cross.


(c) Costmap: Obstacle begins to approach.

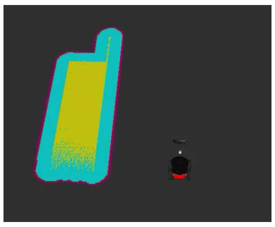

(d) Costmap: Obstacle ends to approach.

Fig. 5. Octomap and costmap mapping moving obstacles

2) Local Obstacle Avoidance: We have tested the behaviour of local obstacle avoidance with moving obstacles. In order to do that, we select an obstacle avoidance algorithm (Curvature Velocity Method) and tests four different mapping systems in three different scenarios: a moving obstacle getting closer to the robot, a moving obstacle crossing the robot's path and a moving obstacle overtaking the robot. The algorithm has the same configuration at each scenario, with its maximum velocities limited to 0.4 m/s and 40 degrees/s. The moving obstacle's size is 0.7 meters long, 0.4 meters width and 1.5 meters height, and moves at 0.4m/s when it is crossing the robot's path and 0.5m/s when it is overtaking the robot. On each scenario the robot should reach a target situated 6 meters ahead of it. We use a Pioneer 3-AT with a parallel laser to the floor.

For the obstacle getting closer to the robot scenario, figure 6 shows paths followed by the robot. The paths followed were similar but the algorithm's behaviour was slightly different. The Raw Data mapping does not have any memory of the obstacle, so the obstacle is avoided when it comes near the robot, what is potentially dangerous. Octomap and Costmap present the obstacle trail issue in the map building as we have probed before, so the obstacle is transformed into a wall separating the path from the ideal path. DOMap predicts the position of the obstacle in the future and its avoidance manoeuvre begins before Octomap or Costmap and follows a path closer to the ideal path.

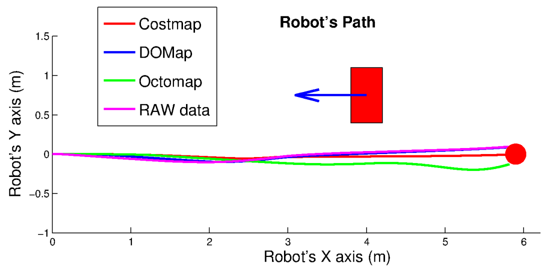


Fig. 6. Obstacle getting closer to the robot scenario: paths followed

For the obstacle crossing robot's path scenario, The Costmap and Octomap problem with trail obstacles is more acute in this case. Octomap is not able of clearing the trail and the algorithm avoid the path followed by the obstacle as if it was a wall. DOMap improves the path followed by Raw Data Mapping: the avoidance manoeuvre starts before because the algorithm predicts that the obstacle is not going to be in the path of the robot. Figure 7 shows the paths followed by the robot.

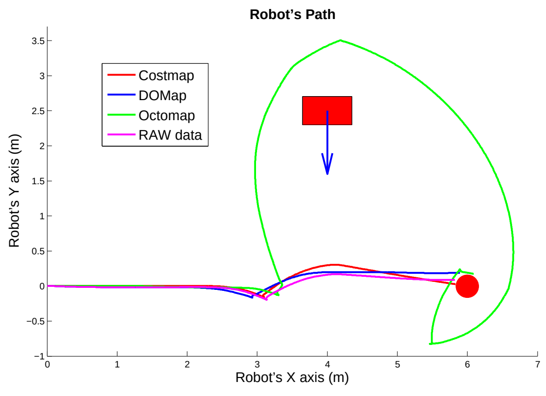


Fig. 7. Obstacle crossing robot's path scenario: paths followed

For the obstacle overtaking the robot scenario, Raw Data Mapping can be dangerous because it can crash with the obstacle due to its lack of memory. Octomap and Costmap have a better performance in this case because an obstacle moving away from the robot does not generate tail (cells are clearing as the obstacle moves away). DOMap predicts the future position of the obstacle and gets away from the trajectory of the obstacle making the obstacle avoidance manoeuvre longer but safer. Figure 8 shows the paths followed by the robot.

3) Global Navigation: We have tested the capabilities of the system to navigate into a full environment simulating a real ABSYNTHE scenario: robot starts in a known position and has to reach an office situated in a small corridor. The environment is static so we choose costmap local mapping stage and test seven different obstacle avoidance algorithms.

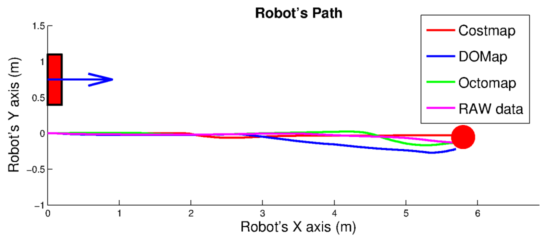


Fig. 8. Obstacle overtaking the robot scenario: paths followed

Figure 9 shows the path planned using a Dijkstra algorithm and the path followed by each obstacle avoidance algorithm. Every algorithm were able to ensure the security of the robot but only three of them were able to reach the target into the corridor: SND, Dynamic Window Approach and CVM.

Reactive algorithms like VFH+ can easily fall into local minimum and were not able to move, in this case it was stuck into a U-shape obstacle. The localization of the robot was not perfect and it affected the behaviour of the algorithm, especially to Elastic Band. This algorithm tried to follow the path planned but due to the robot was too close to an obstacle the algorithm failed and stopped the robot.

CVM based algorithms (CVM, LCM and BCM) were able to reach the corridor's entrance, but due to the intermediate planning stage of LCM and BCM, the entrance of the corridor was never the best selected as a target due to the size of its corridor. SND was able to reach the goal but its velocities were very low due to the complexity of the environment (especially in the corridor). Table I shows the smoothness and lengths of the paths and prove that Dynamic Window approach obtained the best results following the smoothest and fastest path.

TABLE I

GLOBAL NAVIGATION DATA

Algorithm	BCM	CVM	DW	Eband	LCM	SND	VFH
Lin. speed avg (m/s)	0.336	0.319	0.386	0.282	0.192	0.134	0.260
Rot. speed avg (rad/s)	0.019	0.010	0.011	0.021	0.043	0.004	0.050
Distance	53.28	51.59	49.52	12.61	40.75	48.12	15.12
Reach the goal	No	Yes	Yes	No	No	Yes	No

V. CONCLUSIONS AND FUTURE WORKS

In this work we have presented a revision of some several available navigation, mapping and localization packages in ROS, and we have described the integration of our own algorithms. The autonomous navigation of several platforms in simulated and real environments combining local and global algorithms have been tested in order to demonstrate that is robust and safe. Nowadays we are doing more real tests to increase the number of experiments and for generating more multimedia content that will be available in our Robesafe

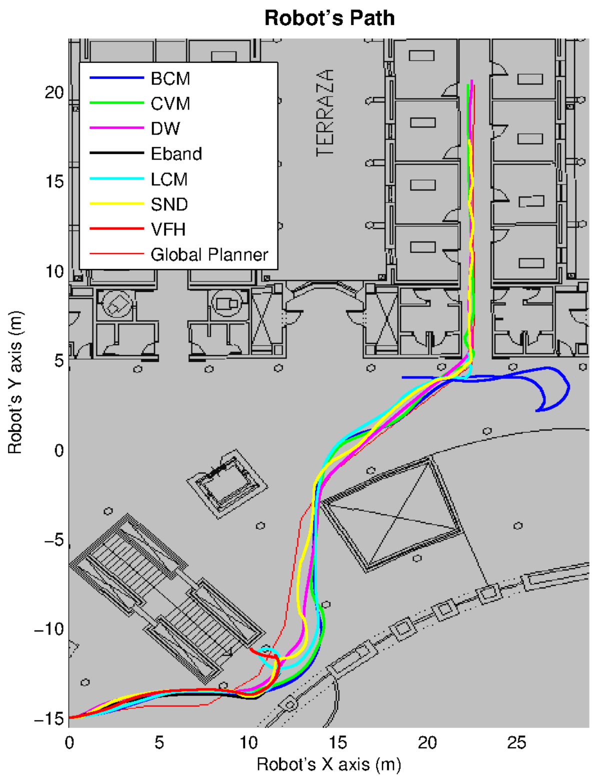


Fig. 9. Global navigation: paths followed by the robot

YouTube Channel (*www.youtube.com/Robesafe*). We are also working on adapting the algorithms to work in 3D environments distinguishing between navigable and non-navigable areas using a 3D range scanner.

VI. Acknowledgements

This work has been funded by TIN2011-29824-C02-01 and TIN2011-29824-C02-02 (ABSYNTHE project) from the Ministerio de Economía y Competitividad.

References

[1] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The stanford entry in the urban challenge," *Journal of Field Robotics*, 2008.

[2] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Winning the darpa grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[3] P. Enge and P. Misra, "Special issue on global positioning system," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 3 –15, jan 1999.

[4] D. Schleicher, L. M. Bergasa, M. Ocaña, R. Barea, and E. López, "Low-cost GPS sensor improvement using stereovision fusion," *Signal Processing*, vol. 90, no. 12, pp. 3294 – 3300, December 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168410002197

[5] M. Hentschel, O. Wulf, and B. Wagner, "A gps and laser-based localization for urban and non-urban outdoor environments," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, sept. 2008, pp. 149 –154.

[6] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[7] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, p. 2007, 2007.

[8] P. Newman, D. Cole, and K. Ho, "Outdoor slam using visual appearance and laser ranging," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, may 2006, pp. 1180 –1187.

[9] K. E. Bekris, M. Click, and E. E. Kavraki, "Evaluation of algorithms for bearing-only slam," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA'06)*, 2006, pp. 1937–1943.

[10] J. L. Blanco, J. A. Fernandez-Madrigal, and J. Gonzalez, "Efficient probabilistic range-only slam," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'08)*, 2008, pp. 1017–1022.

[11] J. L. Fernández, R. Sanz, J. A. Benayas, and A. R. Diéguez, "Improving collision avoidance for mobile robots in partially known environments: the beam curvature method," *Robotics and Autonomous Systems*, vol. 46, no. 4, pp. 205–219, 2004.

[12] J. W. Durham and F. Bullo, "Smooth nearness-diagram navigation," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, September 22-26, 2008, Acropolis Convention Center, Nice, France*. IEEE, 2008, pp. 690–695.

[13] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," *Robotics and Automation, IEEE Transactions on*, vol. 8, no. 1, pp. 23–32, 1992. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=127236

[14] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," in *In Proc. of the ICRA 2010 workshop*, 2010.

[15] S. Thrun and A. Buecken, "Integrating grid-based and topological maps for mobile robot navigation," in *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, 1996.

[16] Á. Llamazares, V. Ivan, E. Molinos, M. Ocaña, and S. Vijayakumar, "Dynamic Obstacle Avoidance Using Bayesian Occupancy Filter and Approximate Inference," *Sensors*, vol. 13, no. 3, pp. 2929–2944, mar 2013. [Online]. Available: http://www.mdpi.com/1424-8220/13/3/2929

[17] J. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[18] P. Pfaff, W. Burgard, and D. Fox, "Robust monte-carlo localization using adaptive likelihood models," in *EUROS*, 2006, pp. 181–194.

[19] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[20] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles." in *ICRA*. IEEE, 2007, pp. 1986–1991. [Online]. Available: http://dblp.uni-trier.de/db/conf/icra/icra2007.html

[21] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," in *In Proceedings of the International Conference on Robotics and Automation*, 1993, pp. 802–807.

[22] I. Ulrich and J. Borenstein, "VFH+: reliable obstacle avoidance for fast mobile robots," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2, may 1998, pp. 1572 –1577 vol.2.

[23] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *Robotics Automation Magazine, IEEE*, vol. 4, no. 1, pp. 23 –33, mar 1997.

[24] N. Y. Ko and R. Simmons, "The lane-curvature method for local obstacle avoidance," in *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, vol. 3, oct 1998, pp. 1615 –1621 vol.3.

[25] C. Coué, C. Pradalier, C. Laugier, T. Fraichard, and P. Bessiere, "Bayesian Occupancy Filtering for Multitarget Tracking: an Automotive Application," *Int. Journal of Robotics Research*, vol. 25, no. 1, pp. 19–30, Jan. 2006, voir basilic : http://emotion.inrialpes.fr/bibemotion/2006/CPLFB06/. [Online]. Available: http://hal.inria.fr/inria-00182004/en/

[26] C. Fulgenzi, A. Spalanzani, and C. Laugier, "Dynamic obstacle avoidance in uncertain environment combining pvos and occupancy grid," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Rome, April 2007. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2007/FSL07