Technikum Wien's entry in the Robotour'11 competition

Thomas Koletschka
University of Applied Sciences Technikum Wien
A-1200, Vienna, Austria
Email: thomas.koletschka@technikum-wien.at

Alexander Hofmann
University of Applied Sciences Technikum Wien
A-1200, Vienna, Austria
Email: alexander.hofmann@technikum-wien.at

***Abstract*—This paper describes the system for an autonomous ground vehicle developed by a team at the University of Applied Sciences Technikum Wien. The goal is to deploy this robot in a city park in vienna and have it navigate autonomously from one point to another without colliding with any objects or driving off the permitted paths. We describe the hardware used by the robot and the software system developed for this competition which is based on the Robot Operating System.**

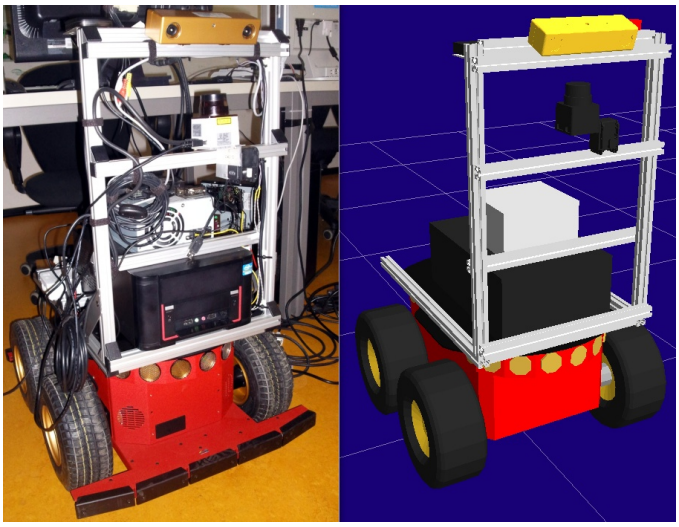

Fig. 1. Our entry in the Robotour 2011 competition (left) and its 3D model (right). The Robot is based on a Pioneer 3AT and additionally equiped with a Bumblebee2 stereo camera and a tilting Hokuyo.

I. Introduction

Robotour [1] is an outdoor delivery challenge for small sized fully autonomous vehicles. The goal of this competition is to navigate within a city park with the only available prior information being an OpenStreetMap map. In four different runs the robots need to carry a 5l beer barrel from a random location within the park to a random target destination. Points are awarded for traveled air distance towards the goal.

This paper describes our approach in conquering this goal with a robot based on a Pioneer 3-At (P3AT), displayed in figure 1 along with its simulation model. The robot will mainly be used on paved park roads, but we want it to be able to navigate on all different kinds of drivable areas such as pedestrian areas or sidewalks. Using this approach we hope to develop a more reliable system which is able to deal with different situations and is therefore able to achieve a good result.

This robot has been developed as part of an undergraduate project at the University of Applied Sciences Technikum Wien.

II. Hardware

A. Base Platform

The robot, as seen in figure 1, is based on a Pioneer 3-AT (P3AT), a robust four-wheel skid-steering robot suitable for outdoor use.
The P3AT comes with an array of each 8 sonar sensors and 5 bumpers on the front and back of the vehicle. Furthermore the platform provides access to its batteries to connect additional sensors and a serial port to control the robot with an external computer.

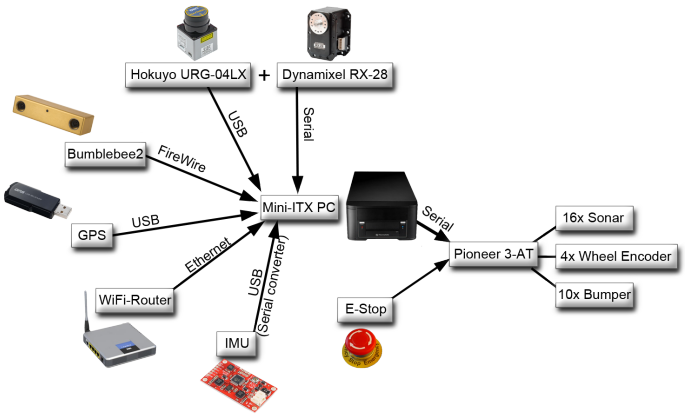

Fig. 2. Hardware Architecture

The robot's computer system consists of a 2.5GHz Intel i5 quad core CPU, 4GB of RAM, a 120GB SSD drive and an additional PCI-E FireWire card to connect the Bumblebee2 camera. The components are placed inside a Mini-ITX tower which is mounted on top of the P3AT above the front axle. That way we have enough room to place the required 5 liter

barrel above the rear axis which should provide an equal distribution of weight and a low center of mass.
Currently the computer system is connected to P3AT's battery system. This is sufficient for short testruns but an additional external battery system that powers the computer only is required to increase the duration to the required minimum of four 30 minute runs.

B. Sensor System

As shown in figure 2 the robot is equipped with 7 different kinds of sensors. The advantage of using multiple kinds of sensors is that one can build a more reliable sensor system by fusing the measurements of different sensors. Combining multiple sensors also helps in covering a wider area which allows the robot to navigate more reliably. Figure 3 illustrates the robot's (red rectangle) sensor coverage. While most of the sensors are oriented in driving direction the sonars on the back of the vehicle also allow for a rough estimation of what's happening behind the robot. This might be useful when passing other robots or when the need arises to back off a couple of centimeters.

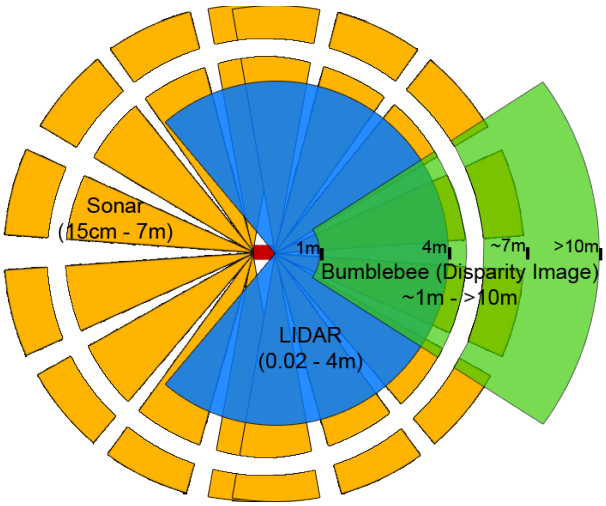

Fig. 3. Sensorcoverage

For inertial navigation, a CH Robotics UM6 Orientation Sensor, consisting of 3-axis gyros, accelerometers and magnetic sensors, is used in combination with a simple GPS receiver, P3AT's wheel odometry and visual odometry [2] computed from the stereo camera's images.
Based on the nature of the robot's environment we assume that GPS signals will be blocked off or heavily distorted most of the time. Furthermore the P3AT uses skid-steering so the odometry calculated from its wheel encoders is not reliable enough either. Therefore it is important to combine those measurements with more reliable sensors such as an inertial measurement unit (IMU) or visual odometry. As shown in [3] the combination of those sensors significantly improves the performance of an inertial navigation system.

The robot uses two main sensors to perceive its environment, a Bumblebee2 stereo camera and a tilting Hokuyo URG-04LX laser scanner.
The Bumblebee has a resolution of 640x480 pixels, a focal length of 3.8mm and a horizontal field of view of 66°. This stereo camera is the main sensor used in the robot's perception system. Its colour images are used for image based path detection and by using the images of both cameras a disparity map can be calculated which is then used for further caluclations such as obstacle detection. Additionally the camera images are used to calculate a visual odometry to improve the overall localization.
We decided against using the Microsoft Kinect which gained increasing popularity in indoor robotics applications since its release in late 2010 as this sensor is based on infrared structured light and is therefore not suited for environments with strong sunlight.
The Hokuyo laser scanner is mounted on a Robotis RX-28 servo motor. The laser has a range of 4 meters and a field of view of 240°. Mounting the Hokuyo on a servo motor enables us to tilt the laser which adds a third dimension to its readings. Tilting the laser allows the robot to cover a larger field of view and, especially due to the slow driving speed of our Pioneer robot, we are able to create a more accurate 3D model of the environment immediately in front of the robot.

The sonars and bumpers that come with the base platform are only used as supplementary sensors and emergency stop. Sonars are too imprecise to use them for accurate obstacle detection, but they still provide some rough estimate which e.g. can be used when passing other robots. The bumpers are used as emergency stop to protect the robot and the colliding object or person when all other sensors have failed.

III. Software Architecture

Our software architecture uses the Robot Operating System (ROS) [4] framework and is therefore mainly written in C++. ROS gained increasing popularity in the robotics community throughout the past years and today provides a large amount of robotics libraries and tools. Using ROS we are able to focus on the key problems of the competition such as path detection and reliable navigation as ROS already provides drivers for using all our sensors, a fairly mature communication interface and a lot of tools such as logging which are already integrated in the system.

Our system, illustrated in figure 4, is divided into three main layers:

- **Sensor Layer**: The sensor layer is the first layer in our processing pipeline. The processes in this layer simply read the measurements of all our sensors and publish them using the ROS communication framework. All the sensors used on our robot are supported by ROS and therefore we did not have to write a lot of code to integrate them in our system.

- **Perception Layer**: This layer receives the stereo camera, LIDAR (Light Detection And Ranging) and sonar data and fuses them into an evidence grid [5] using the sensor locations as described in the global robot model. Furthermore it tries to detect a drivable path and calculates the visual odometry.
- **Planning & Control Layer**: The last layer in our system is responsible for calculating actions based on the information provided by the sensor and perception layers. It calculates the current global position of the robot and tries to find the optimal path to the designated goal using an OpenStreetMap (OSM) map. After calculating the required trajectory to follow the path it sends the corresponding steering commands to the robot.

ROS' flexible communication system enables us to easily log, print and visualize information sent between all the processes which allows us to debug the system more easily. Another big advantage is that we are able to use the same system for both the real robot and a simulated model in the Gazebo simulator.

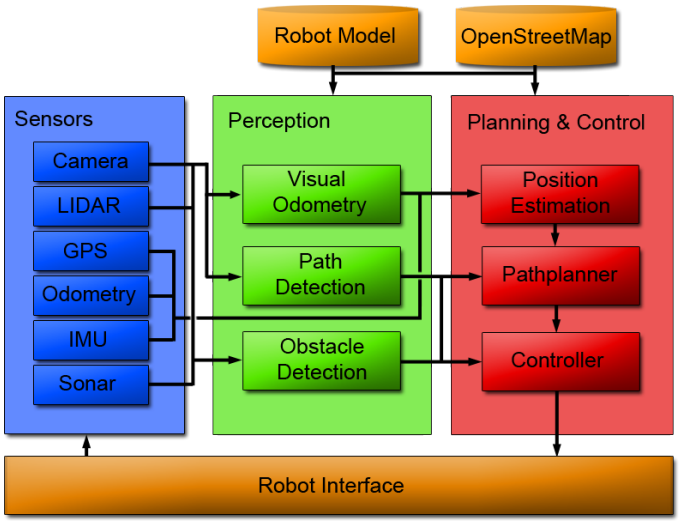

Fig. 4. Software Architecture

A. Global Information

There are two kinds of (mostly) static information which are of use to the whole system: the robot model and the OSM map used for navigation.

1) Robot Model: ROS uses the Unified Robot Description Format (URDF) to describe the structure of robots. A URDF file is an XML file describing a tree structure of the robot and different attributes such as size and weight of its parts. It can also be expanded with information about simulated sensors to use the robot model in a simulator such as Gazebo.
The robot model is modified and published using ROS' TF library which provides an easy way of dealing with the manifold of coordinate frames in a robotics system.
Besides for simulation purposes we use the URDF model to locate the origin of each sensor to fuse their data in a global reference frame. In the planning & control layer the model is used to avoid collisions with objects and plan a feasible path.

2) OpenStreetMap: The OSM map is mainly used for global path planning and the only map permitted for use in the Robotour'11 competition. We've developed a custom OSM parser and service layer to provide useful information such as the distance to the street crossings connecting to the current street or the width and type of the current street.
At the beginning of each run the globally optimal path to the goal is calculated using Dijkstra's algorithm with a Fibonacci heap [6]. If a path should turn out to be blocked or we should somehow miss a path we are still able to calculate a new path and proceed towards our goal.

B. Perception Layer

The perception layer is responsible to feed the planning & control layer with sufficient information to safely navigate the robot through its environment. The perception layer consists of three main modules: visual odometry (VO), sensor fusion and obstacle detection, and path detection.

1) Visual Odometry: Visual Odometry (VO) [2] is the process of determining the position and orientation of a robot by calculating the motion between each pair of images. VO is usually much more precise than wheel odometry and when combining the measurements of all position estimation sensors the best inertial localization results can be achieved, as demonstrated in [3]. This also helps to accomodate for blocked and distorted GPS signals as it is possible to reliably travel several hundreds of meters using only VO, IMU data and wheel odometry.
ROS provides a package called vslam to calculate the visual odometry based on stereo image pairs, therefore we did not have to implement this from scratch and save a lot of valuable development time. If the vslam package should not prove to be reliable enough for our purposes we will write a ROS wrapper for the VO library recently released[1] by [7].

2) Sensor Fusion and Obstacle Detection: Probabilistic fusion of multiple sensor measurements and obstacle detection is achieved using an evidence grid [8]. The advantage of using evidence grids is that different sensor models can be used to accommodate for differently accurate sensors. For example the uncertainty of an obstacle being present at a given location is much higher when using sonar measurements compared to the measurements obtained from a Hokuyo laser scanner.
After fusing all the sensor data into one common evidence grid, we simply need to threshold the values to detect obstacles and drivable terrain. Due to the probabilistic nature

[1] http://www.rainsoft.de/software/libviso2.html

of the grid and the short amount of time between each update, moving obstacles are updated accordingly and do not require any additional processing.

ROS already provides 2D and 3D occupancy grids, but we were not satisfied with the current implementation as it does not use a probabilistic model. Therefore we use the ROS wrapper of OctoMap [9], which implements an octree storing probabilistic sensor readings, for fusing our 3D sensor readings and afterwards convert those values into a 2D occupancy grid used by ROS' navigation system.

3) Path Detection: Using solely the fused evidence grid it would already be possible to follow a save path and avoid obstacles. The Robotour'11 rules specify that the robot must not leave the labeled pathways. This would not be possible using solely 3D information as the paths are not always bound by high objects such as fences or bushes. Therefore we also use the 2D color images published by our Bumblebee camera in combination with the previously calculated 3D environment to detect the drivable path.

Our approach is to find drivable terrain in a region immediately in front of the robot using the information stored in our evidence grid. After locating the drivable surface within that area we project the region into the image plane of one Bumblebee camera and use the selected pixel information to learn a visual model of the current street. Using that model we are then able to detect the path in the image and further extract an approximate shape of the path to avoid driving off the permitted path.
This approach has already been described in [10] and [11] and has successfully been applied to real-world navigation problems.

C. Planning & Control Layer

The Planning & Control layer is the final step in our processing pipeline. It is responsible for calculating the current position, calculating and following a global path, and finally sending the appropriate steering commands to the robot. To fullfill those tasks the layer's processes incorporate all the previously calculated information such as obstacle maps and drivable surfaces and global information such as the OSM map.

1) Position Estimation: The robot needs to have good knowledge about its current global position on a given map to reach the goal expressed in gps coordinates. In the robot's main domain, a city park, it is expected that the gps signal is blocked off or distorted by trees and buildings most of the time. Therefore it is not safe to rely solely on gps navigation. We use an approach as described by [3] to improve our global position estimate by fusing GPS, visual odometry, wheel odometry and IMU measurements. This allows us to travel reliably even with long GPS dropouts and large errors in received signals.
To do this we use a modified version of ROS' extended kalman filter (EKF) pose estimator as the standard filter only uses the estimates of three sensors while we are able to use the estimates of four different sensors.

2) Pathplanner: Robotour'11 rules demand the use of an OpenStreetMap (OSM) map as the only source for global navigation. Teams are not allowed to use private data such as custom built maps or pre-recorded trajectories. To fullfill this requirement the pathplanner uses the information provided by our global OSM service to calculate a global path to the desired goal position. In the best-case scenario this calculation usually only needs to occurr once at the beginning of each run. If a path should turn out to be blocked or we miss a crossing we are able to calculate a new route and will still be able to proceed towards the goal.
While maps usually provide a fairly good representation of street roads, unstructured paths such as paths in a park are usually only provided with simplified representations. To accommodate for those approximations we do not entirely rely on the path shape defined in the map but rather use the path detected by the vision layer to plan our local path. It is the responsibility of the controller module to follow this local path.

3) Controller: After the pathplanner has finished calculating a local path, it is the controller's responsibility to calculate the required steering commands to follow this path while avoiding all obstacles. Many robots, such as [12], use a simple PID controller to keep the vehicle on track. This method has the disadvantage of ignoring vehicle kinematics and dynamics. We chose to use a technique known as *trajectory rollout*, e.g. used in [13] and [14]. This method calculates multiple trajectories by simulating different feasible steering commands over a short amount of time. It then iterates over all calculated trajectories and selects the trajectory that does not collide with any obstacles and is closest in distance and orientation to the desired path.
We do not take speed control into account as the maximum speed of our robot is limited to 2.8km/h so we simply differ between stop and full-speed which simplifies our calculations when iterating through different possible steering commands. ROS already provides an interface for different controllers which we were able to use. In theory it would also provide rajectory rollout but, like many ROS libraries, this planner is tailored to the PR2's base and therefore was not suited for our robot.

IV. Conclusion

In this paper we have described our approach in conquering the goal of the Robotour competition which we will compete at in September 2011. We explained the hardware used in this system and the advantages of each sensor and how we accommodate for their disadvantages. Further we described the software system and the use of the Robot Operating

System in this system.

Currently our system uses just dead reckoning for navigation, which is the calculation of one's current position based on the mostion estimate since the last calculated position and hence prone to increasing errors with traveled distance. This can lead to wrong decisions when trying to follow a route on a street network and the robot might end up far from the intended destination.
On roads with lane markings localization can be improved up to a few centimeters in precision by aligning detected lane markings with previously recorded maps or aerial images of the current street, as shown in [15] and [16]. This approach is not possible with roads in our vehicle's domain, such as park roads and other unstructured paths, as those do not have any defined lane markings. Therefore a new approach needs to be designed to improve localization in such road networks e.g. by reliably detecting road crossing or road shape features to compare them against a human map and thereby increase knowledge about the current position.
Another interesting application would be the use of an unmanned aerial vehicle equipped with a downward facing camera to support the autonomous ground vehicle in determining its current position by providing aerial imagery of the current road segment and the robot's position on it.

We hope this paper provides good insight for both current and future teams and helps them in developing their systems. Our software system will be released after the competition and teams are welcome to take a look at it.

ACKNOWLEDGMENT

The authors gratefully acknowledge the contributions of all persons involved in the project. Special thanks to Mohamed Auf, Patrick Schwab and Marvin Vesely for their work in developing the system and Ewald Schmudermayer for his mechanical support. The project is funded by the University of Applied Sciences Technikum Wien.

REFERENCES

[1] J. Ia and M. Dlouh, "Robotour - robotika.cz outdoor delivery challenge," in *Proceedings of the 1st international conference on Robotics in Education, RiE2010*. FEI STU, Slovakia, 2010, pp. 89–93.

[2] D. Nister, O. Naroditsky, and J. Bergen, "Visual Odometry," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 1, pp. 652–659, 2004.

[3] M. Agrawal and K. Konolige, "Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS," in *ICPR*, August 2006.

[4] 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.

[5] H. P. Moravec, "Sensor Fusion in Certainty Grids for Mobile Robots." *AI Magazine*, pp. 61–74, 1988.

[6] M. Fredman and R. Tarjan, "Fibonacci Heaps And Their Uses In Improved Network Optimization Algorithms," *Foundations of Computer Science, Annual IEEE Symposium on*, vol. 0, pp. 338–346, 1984.

[7] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3d Reconstruction in Real-time," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011.

[8] M. Martin and H. Moravec, "Robot Evidence Grids," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-96-06, March 1996.

[9] 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 *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, Anchorage, AK, USA, May 2010. [Online]. Available: http://octomap.sf.net/

[10] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski, "Self-supervised Monocular Road Detection in Desert Terrain," in *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.

[11] I. Katramados, S. Crumpler, and T. P. Breckon, "Real-Time Traversable Surface Detection by Colour Space Fusion and Temporal Analysis," in *Computer Vision Systems*, ser. Lecture Notes in Computer Science, vol. 5815. Springer Berlin / Heidelberg, 2009, pp. 265–274.

[12] 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, "Stanley, the robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, 2006, accepted for publication.

[13] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. D. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge," *J. Field Robot.*, vol. 25, no. 9, p. 598–614, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?id=1405647.1405653&coll=GUIDE&dl=GUIDE&CFID=30320363&CFTOKEN=48228491

[14] B. P. Gerkey and K. Konolige, "Planning and Control in Unstructured Terrain," in *ICRA 2008 Workshop on Path Planning on Costmaps*, May 2008.

[15] J. Levinson and S. Thrun, "Robust vehicle localization in urban environments using probabilistic maps," in *ICRA*, 2010, pp. 4372–4378.

[16] O. Pink, F. Moosmann, and A. Bachmann, "Visual features for vehicle localization and ego-motion estimation," in *Intelligent Vehicles Symposium, 2009 IEEE*, june 2009, pp. 254 –260.