Mixed 2D/3D Perception for Autonomous Robots in Unstructured Environments

Johannes Pellenz[1], Frank Neuhaus[2],
Denis Dillenberger[2], David Gossow[2], and Dietrich Paulus[2]

[1] Bundeswehr Technical Center for Engineer and General Field Equipment,
Universtitätsstr. 5, 56070 Koblenz, Germany
JohannesPellenz@bwb.org
[2] Active Vision Group, University of Koblenz-Landau,
Universitätsstr. 1, 56070 Koblenz, Germany
{fneuhaus,dillenberger,dgossow,paulus}@uni-koblenz.de,
http://robots.uni-koblenz.de

Abstract. Autonomous robots in real world applications have to deal with a complex 3D environment, but are often equipped with standard 2D laser range finders (LRF) only. By using the 2D LRF for both, the 2D localization and mapping (which can be done efficiently and precisely) *and* for the 3D obstacle detection (which makes the robot move safely), a completely autonomous robot can be built with affordable 2D LRFs. We use the 2D LRF to perform particle filter based SLAM to generate a 2D occupancy grid, and the same LRF (moved by two servo motors) to acquire 3D scans to detect obstacles not visible in the 2D scans. The 3D data is analyzed with a recursive principal component analysis (PCA) based method, and the detected obstacles are recorded in a separate obstacle map. This obstacle map and the occupancy map are merged for the path planning. Our solution was tested on our mobile system Robbie during the RoboCup Rescue competitions in 2008 and 2009, winning the mapping challenge at the world championship 2008 and the German Open in 2009.

This shows that the benefit of a sensor can dramatically be increased by actively controlling it, and that mixed 2D/3D perception can efficiently be achieved with a standard 2D sensor by controlling it actively.

1 Introduction

Autonomous mobile systems are still a current research topic; the testing ground for the systems is often a obstacle free laboratory or office like environment or structured, well defined areas such as the RoboCup Soccer fields. However, the ongoing development also enables using robots in rough terrain as well. This was demonstrated in the DARPA Grand Challenge in 2004 and 2005, as well as in the DARPA Urban Challenge in 2007 [Thrun, 2006]. Most of the successful robots in the DARPA competitions were equipped with either multiple 2D laser range finders (LRFs), or with a 3D LRF that scans in 64 planes at the same time (Velodyne HDL-64E[1]). Using multiple LRFs or a Velodyne HDL-64E is not an option for most small robots due to the high cost,

[1] See: http://www.velodyne.com/lidar

J. Ruiz-del-Solar, E. Chown, and P.G. Ploeger (Eds.): RoboCup 2010, LNAI 6556, pp. 303–313, 2011.

the weight and the power consumption. An alternative to multiple 2D LRFs is to rotate a single 2D LRF (for an application in rescue robotics, see [Ohno et al., 2008] or [Nüchter, 2006]). Using such a rotating 2D LRF while driving results in a lack of accuracy, which finally yields to non precise maps. Sheh et al. [Sheh et al., 2007] avoid this problem by using a SwissRanger SR-3100 time of flight (TOF) camera. They use the Hough transform to generate a model of the environment, which is assumed to have the form of a NIST random step field (a well defined structure with a significant direction). Therefore, their approach does not work in unstructured environments. Kadous et al. [Kadous et al., 2006] used extracted features in the surrounding of the robot to select behaviors to control the robot over the rough terrain. The focus of their work is to find a suitable representation of the environment for machine learning.

Our goal is to combine – in one device – the complete perception in 3D using a 3D LRF *and* the speed and precision of a static 2D LRF for accurate 2D position estimation and precise 2D maps. Our approach works as follows: The localization and mapping is done in 2D while the robot is driving. The 2D laser scans come from short range 2D LRF which is mounted on two servo motors. The global map is stored in an single occupancy grid. The grid is composed of 50 mm $\times$ 50 mm cells that store the information if the cell is occupied or not [Elfes, 1989]. The information is stored in two planes: One plane stores the number of times the cell was "touched" by a laser beam, and the other plane stores the information if the cell has be seen as occupied. The ratio of the values on both cells yields the probability value for an occupation. A frontier based algorithm (Exploration Transform, see [Wirth and Pellenz, 2007]) selects the next frontier that the robot is going to explore and also calculates the safest path to this target. Whenever such a frontier is reached, a 3D scan is acquired. To capture the 3D scan, the same LRF that is used for the 2D mapping is utilized by tilting the LRF with one of the servo motors. To detect the regions where obstacles occur, the 3D data is analyzed using a hierarchical principal component analysis (PCA): First, a larger area is analyzed. If the whole area is "flat" (by checking the properties of the eigenvalues), then no 3D obstacles were detected. If there are obstacles in the area, the area is split in two parts, and the PCA is applied for both sub-regions. This is done until the remaining areas are flat or are smaller than 10 cm $\times$ 10 cm. After the analysis, the location of 3D obstacles is known in the local robot coordinate system. Using the localization information from the 2D particle filter based localization step, these spots are stored in a separate map, the obstacle map. Keeping the information separate helps the localization algorithm to match the following 2D laser scans with the occupancy map. However, for the navigation algorithm it is important to known about both, the obstacles detected by the 2D and the 3D LRF. Therefore, both maps are merged into one navigation map, which is then used by the Exploration Transform to plan the path.

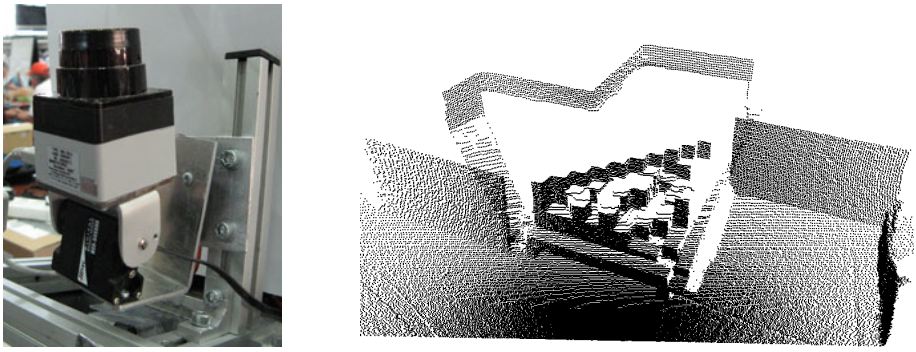
The mixed 2D/3D perception was implemented on our mobile robot Robbie and was tested during the RoboCup in 2008 and 2009. Using this technique, our wheel-based robot was able to detect obstacles in different heights such as step fields and ramps, but also objects hanging from the arena elements (stalactites).

The rest of the paper is organized as follows: Sec. 2 describes the self made 3D sensor that is used for most of the experiments. Sec. 3 explains the algorithms that are used to analyze the 3D data and to store the gained information in two maps. Sec. 4

describes the experiments that were performed using two different types of data sets, an outdoor dataset (captured using a commercial 3D LRF) and an indoor RoboCup Rescue data set (captured with the self made 3D LRF). Sec. 5 describes the results, and Sec. 6 finally lists the open issues and the future work.

2 Sensor Description

The 2D laser scans for the indoor datasets come from a Hokuyo URG-04LX LRF; this sensor can measure distances up to 5.6 meters. The LRF is mounted on two servo motors, which – in 2D mode – keep the sensor always adjusted [Pellenz, 2007]. This way, the laser beams still measure the surrounding walls, and not the ground or obstacles behind the next barrier. The setup is depicted in Fig. 1a. The servo motors are controlled by a microcontroller and a laptop; the declination data comes from a cheap two-axis gravity sensor (ADXL202). In the 3D mode, the roll servo is used to keep the servo straight, and the tilt servo is used to change the scanning plane. 55 scans are taken in about 1° steps in front of the robot. An example of a 3D laser scan is depicted in Fig. 1b.



(a) Laser scanner mounted on two servo motors.

(b) Resulting 3D laser scan.

Fig. 1. Laser range finder Hokuyo URG-04LX gimbal mounted on two servo motors (the roll servo motor is occluded) and the resulting 3D point cloud of a random step field

3 Algorithms

The analysis of the 3D point cloud is based on the principal component analysis (PCA). In the following subsections, first the PCA and the grid-based PCA is introduced, followed by our extension, the Hierarchical PCA. Finally, an approach to analyze the roughness of the ground using the distribution of 3D laser distances is presented.

3.1 PCA

The PCA of the covariance matrix of a three-dimensional point cloud (which is in this specific case in fact equivalent to the eigendecomposition) yields three eigenvectors

with corresponding eigenvalues. The latter indicate the variance of the point cloud along the corresponding axes, and can be used to determine the shape of the point cloud. Fig. 2 shows the three different cases that can occur: If all eigenvalues are roughly equal, the point cloud is more or less unstructured without any specific principal direction. If one of them is significantly higher than the remaining two, the shape of the point cloud is cylindrical and the eigenvector corresponding to the highest eigenvalue defines the direction of the cylinder. If one of the eigenvalues is very low in comparison to the other two, the point cloud is more or less a plane, and the eigenvector corresponding to the lowest eigenvalue defines the normal of the plane.

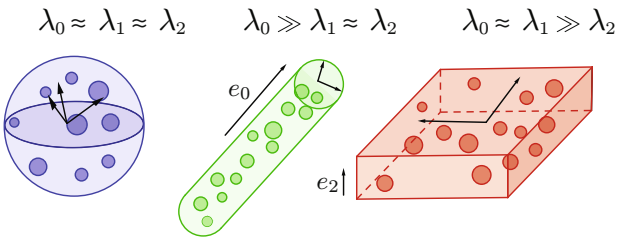


Fig. 2. Three cases of point cloud analysis

The above mentioned methodology only yields good results for reasonably small point clouds. Those containing multiple planes/cylinders or other complex structures simply end up in the case where all three eigenvalues are large. In order to gain local information about a large point cloud (such as the one delivered by the 3D laser scanner), the data has to be subdivided into smaller chunks, which can then be analyzed with the help of the PCA. There is a multitude of possibilities to subdivide the point cloud. We use a simple 2D grid, which is centered around the origin of the sensor. Other researchers such as [Lalonde et al., 2006] used 3D grids. The problem with these uniformly sized cloud-fragments is that the ever increasing spread of sensor's field of view makes information about far-away ground surfaces sparse. Selecting a small cell-size for the grid results in many distant cells being completely empty, simply because not a single laser beam endpoint happens to be inside this cell. Selecting a large cell size allows larger regions to be considered in the PCA step. However, this causes problems in regions close to the sensor: If a small obstacle is inside a cell, the whole cell is classified as occupied. Subsequent path planning algorithms operating exclusively on this grid would have to keep significantly more distance to obstacles than really needed. In some cases, such as narrow passages, where precise navigation is required, this methodology can even impede path-planning altogether, simply because almost everything is marked as impassable. What is really needed is an algorithm that combines the advantages of both, small *and* large cells.

3.2 Hierarchical PCA

The Hierarchical PCA is an algorithm we developed to recursively subdivide the point cloud. We use it on a 2D grid, however the extension to 3D is straightforward. The cell size is chosen to be relatively small.

The initial input of the algorithm is the whole grid. In each step, it performs a PCA analysis on the points in the considered region. If the point cloud contained in the region is not flat enough, the algorithm splits the region in two pieces and recursively calls itself on the two resulting fragments. The splitting simply occurs at the middle of the longest edge of the region. If the input region can no longer be subdivided, and it is still not flat enough, it is considered an obstacle. To quantify flatness, we first define the local height disturbance h as:

$$h = \lambda_k \quad \text{with} \quad k = \underset{i \in \{0,1,2\}}{\operatorname{argmax}} |\mathbf{e}_i^{\mathrm{T}} \mathbf{z}| \tag{1}$$

where $\mathbf{z}$ is a vector pointing up, $\mathbf{e}_i$ are the eigenvectors, and λ_i the eigenvalues of the point cloud in the current region. A region is considered flat, if h is below some threshold. This definition makes sure, that even sloped but flat areas get low values of h.

Algorithm 1. Recursive PCA Lterrain analysis

```
function RECURSIVEPCA(area a)
    h ← local height disturbance (a)          ▷ Using the eigenvalues calculated by the PCA
    if (h ≤ t) then                                                          ▷ t is a threshold
        mark all cells in a as drivable
    else
        if (size(a) ≤ (10 cm × 10 cm)) then
            mark cell in a as obstacles
        else
            (a_0, a_1) ← split (a)                                  ▷ Split the area in two parts
            recursivePCA(a_0)
            recursivePCA(a_1)
        end if
    end if
end function
```

3.3 Roughness Analysis

Having distinguished between passable and non-passable areas, one issue remains: Even though a region may technically be drivable, it may still be desirable to prefer one region over another. An example of this is when the robot is driving on a track and next to this track a still passable rubble pile is located. Obviously, one expects the robot to continue its way on the track instead of driving through the debris. One may be tempted to use simply the local height disturbance as a measure for the roughness of the terrain at that spot. Unfortunately, it turns out that due to the large sensor noise relative to the height of the bumps that are to be detected, it is infeasible to distinguish rough from flat areas properly.

As [Montemerlo et al., 2008] have already noticed, a much better indicator of roughness is the local *distance* disturbance, since small bumps and dents in the ground cause large changes in distance of adjacent laser measurements because of the grazing angles with which the laser rays hit the ground. This can be seen in Fig. 3: The box represents

the laser range finder, mounted at a height h_1. The robot is standing in front of a bumpy area. The bumps are assumed to have a fixed height h_2 in the considered region. Now each ray will either hit a bump or barely graze above one, hitting the ground behind. The distance difference e between a measurement that has hit the ground, and one that has hit the bump is relatively large compared to the height of the bump. In addition to that, it linearly increases as the distance to the bump grows. This can be seen in the following equation for e, which can be derived using the intersecting lines theorem:

$$e = d \cdot \frac{h_2}{h_1 - h_2} \tag{2}$$

Note how d scales up the remaining term, which in fact describes the intensity of the bump.

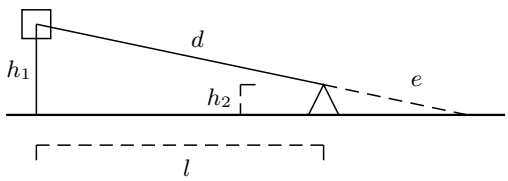


Fig. 3. Geometry of distance disturbances

The 3D laser scanner based on the Hokuyo URG-04LX provides 55 2D laser scans per 3D scan: one for each orientation the tilt servo motor was set to. We consider each 2D laser scan separately. If one laser is not able to measure a distance at a certain spot, a very low distance value will be reported (values below 20 indicate an error). In a first fast preprocessing step, we eliminate these erroneous measurements. This is done by looping over each scan, overwriting every faulty measurement with the last valid value. In the next step, a heavily smoothed copy of this data is made. We use a simple mean filter (of size nine)–mostly because of speed and because superior low-pass filters such as Gaussian or even edge-preserving smoothing filters did not show any significant improvement. Finally, the unfiltered version is subtracted from the low-pass filtered version, forming a high-pass filtered version of the data. Now, the distance deviation δ_i from the smoothed version is known for each laser measurement. For each grid-cell containing points, we compute the variance of distance deviations $\hat{\sigma}_\delta^2$ of all n points in that cell, using

$$\hat{\sigma}_\delta^2 = \frac{1}{n}\sum_{i=1}^{n}\delta_i^2 - \left(\frac{1}{n}\sum_{i=1}^{n}\delta_i\right)^2 \tag{3}$$

However, $\hat{\sigma}_\delta^2$ is really a sum of the inherent sensor noise σ_L^2 and the actual distance variance σ_δ^2. Since we are only interested in the latter, and since we assume the sensor noise to be additive, we eliminate the sensor noise by computing σ_δ^2 as $\max\{0, \hat{\sigma}_\delta^2 - \sigma_L^2\}$.

This measure is not yet independent to the distance, as we have motivated in equation (2): It will be too low in areas near the robot, or too high for areas which are far away. Therefore, we obtain the local terrain roughness r as we eliminate the previously computed measure from the influence of the distance:

$$r = \frac{\sigma_\delta^2}{d_{\text{Cell}}^2} \tag{4}$$

where d_{Cell} is the distance of the laser to the respective grid-cell. The roughness r can now be used to determine the drivability of the terrain: High values of r correspond to high terrain roughness, low values to low roughness. Now a robot could, for example, try to find a path, where the sum of all r's in the crossed cells is minimal.

3.4 Obstacle Map m^{obst}

After the analysis of the 3D scan using the Hierarchical PCA, the location of 3D obstacles is known in the local robot coordinate system. For performance reasons, we do not want to perform a complete 3D mapping which would involve a time consuming and sometimes unstable 3D point cloud registration as described in [Nüchter, 2006]. In our case it is sufficient to take the 2D pose information (x, y, θ) from the 2D SLAM and to calculate the positions of the obstacles using this pose information.

It is important that the 3D obstacles are stored in a separate map and that the occupancy map (with the obstacles seen by the LRF in 2D mode) is not modified, because the occupancy map is used as a reference for the 2D SLAM algorithm. If the 3D obstacles would be marked here, mismatches between these obstacles and other objects in the 2D scans could occur.

Using the localization information from the 2D particle filter based localization step $(x_{robot}^{\mathsf{W}}, y_{robot}^{\mathsf{W}}, \theta)^{\mathsf{T}}$, the world coordinates of the midpoint $\boldsymbol{p}_{obst}^{\mathsf{W}} = (x_{obst}^{\mathsf{W}}, y_{obst}^{\mathsf{W}})^{\mathsf{T}}$ of a occupied cell is calculated:

$$\boldsymbol{p}_{obst}^{\mathsf{W}} = \begin{pmatrix} \cos(\theta) x_{obst}^{\mathsf{R}} - \sin(\theta) y_{obst}^{\mathsf{R}} + x_{robot}^{\mathsf{W}} \\ \sin(\theta) x_{obst}^{\mathsf{R}} + \cos(\theta) y_{obst}^{\mathsf{R}} + y_{robot}^{\mathsf{W}} \end{pmatrix} \tag{5}$$

Finally, the obstacle is stored in a separate map $\boldsymbol{m}^{obst}$, if it is not already marked in the occupancy grid $\boldsymbol{m}^{occ}$:

$$\boldsymbol{m}^{obst}_{\text{worldToMap}(\boldsymbol{p}_{obst}^{\mathsf{W}})} = occupied, \text{ if class}(c_{i,j}) = occupied \wedge \boldsymbol{m}^{occ}_{\text{worldToMap}(\boldsymbol{p}_{obst}^{\mathsf{W}})} = free \tag{6}$$

By checking for the obstacle in $\boldsymbol{m}^{occ}$ the algorithm makes sure that obstacles are stored only once if the obstacle was detected by the 2D LRF already.

4 Experiments

We tested the terrain classification algorithms on two different types of data sets: One dataset was acquired using the commercial 3D LRF Velodyne HDL-64E. These datasets represent larger scale outdoor scenes of the campus of the University Koblenz-Landau

and the surrounding. The other dataset was captured using the self made 3D LRF that was presented in Sec. 2. These datasets were captured on different RoboCup events. The result of the terrain classification is visualized: Every grid-cell is rendered as a small quadriliteral. They are oriented and located to approximate optimally the points contained in the respective grid-cells. Impassable cells are visualized by red boxes. The color of the quadriliterals is the result of the roughness analysis algorithm which was presented in section Sec. 3.3. The colors range from green (low roughness) to orange (high roughness).

(a) Velodyne HDL-64E mounted on a (human driven) street car.

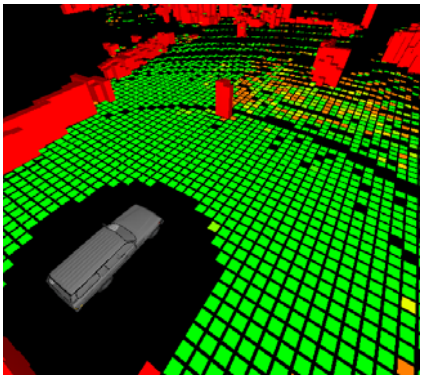
(b) 3D LRF mounted on our autonomous rescue robot Robbie.

Fig. 4. Vehicles for capturing the test data

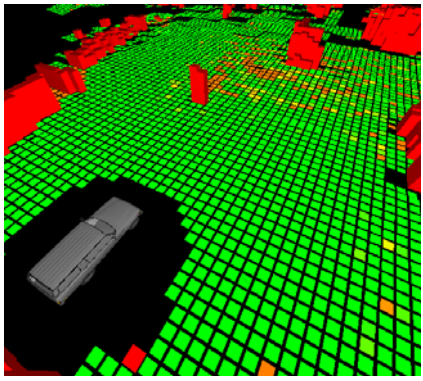
The sensors mounted on the test vehicles are shown in Fig. 4. Since only for the RoboCup dataset also a 2D map was generated, the mixed 2D/3D perception was tested with the RoboCup datasets only. For the visualization, the obstacle map $\boldsymbol{m}^{obst}$ was overlaid the occupancy grid $\boldsymbol{m}^{occ}$ so that the position of the obstacles can be verified visually. Finally, the robot equipped with the mixed 2D/3D perception was used during the RoboCups in 2008 and 2009.

5 Results

The result of the hierarchical PCA that was introduced in subsection 3.2 can be seen in Fig. 5a and 5b for a typical outdoor scene. Fig. 5a shows the result of the classical PCA with fixed size cells: Most cells are classified, but several cells–where not enough points were available–remain not classified. It is clearly visible that distant cells often do not contain any laser-endpoints. Fig. 5b shows how the Hierarchical PCA handles the same data: Multiple equally flat cells are merged for the analysis and small ”holes” in the data coverage are no longer a problem for the path-planner. However, since the algorithm performs the maximum subdivision in areas close to obstacles, the issue remains near these objects. This can be seen on the right in Fig. 5b (there is a small hole next to the obstacle).

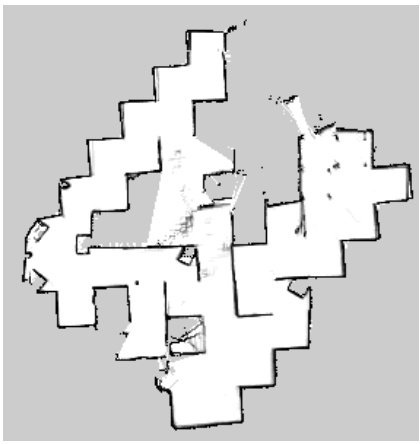


(a) PCA with fixed grid size: Some grid cells are unknown.

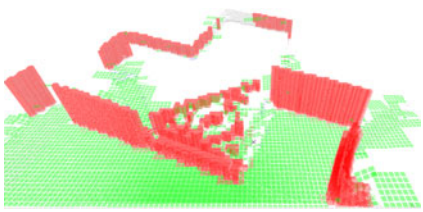


(b) Hierarchical PCA with variable grid size: Continuous surfaces.

Fig. 5. Comparison of the PCA with a fixed grid size (a) and the Hierarchical PCA (b). With the standard PCA gaps in the grid occur, whereas with the Hierarchical PCA larger regions are classified consistently as passable. (Note that the smallest possible tiles are shown in both images. The larger "logical tiles" used in the Hierarchical PCA are not visualized.)



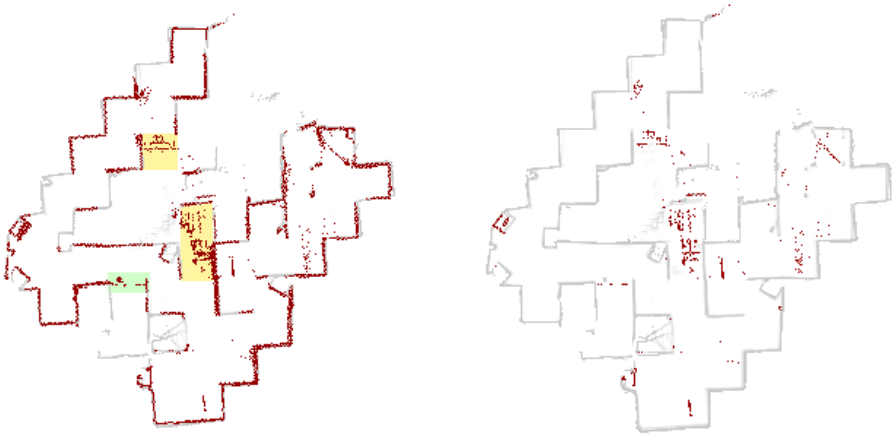
(a) 2D occupancy grid m^{occ} of a RoboCup Rescue arena (size is about 12 m $\times$ 12 m). For the robot not passable areas are marked as free, because the 2D LRF did not see the obstacles.)



(b) 2D grid with the result of the Hierarchical PCA: Red regions indicate obstacles, green regions indicate passable terrain. The position of the obstacles is stored in the map $\boldsymbol{m}^{obst}$.

Fig. 6. Same arena, different views: Complete 2D map of the arena and a classified 3D LRF scan

The result for the RoboCup dataset is shown in Fig. 6 and Fig. 7. Fig. 6a shows the 2D occupancy map, as it is generated by our 2D particle filter based SLAM approach. Notice that only the walls are visible as obstacles (in black); neither the ramps nor the step fields are detected by the 2D laser range finder. The 3D laser scan and the



(a) Visual representation of the 2D map of the detected 3D obstacles $\boldsymbol{m}^{obst}$. Obstacles are marked as dark dots.

(b) Like Fig. 7a, but obstacles that are already recorded in the occupancy grid $\boldsymbol{m}^{occ}$ are filtered out.

Fig. 7. Two versions of $\boldsymbol{m}^{obst}$: With all detected obstacles and only with obstacles that were not seen by the 2D LRF. (For better orientation, the occupancy grid is also shown in light gray.)

classification result (occupied or not) is shown in Fig. 6b. The walls are clearly visible, but also the stepfield (in the middle of the scan) is detected. Using the pose estimation from the 2D SLAM, the position of all occupied cells (in red) is stored in $\boldsymbol{m}^{obst}$. This map is shown in Fig. 7a. Since some obstacles (such as the walls) would be stored in both maps, obstacles that are already stored in the occupancy map are filtered out from $\boldsymbol{m}^{obst}$, because cells here have a larger size and this redundant information would reduce the resolution of the merged map for the path planning. The map that is finally used by the path planner is depicted in Fig. 7b.

Our solution was tested on our mobile system Robbie during the RoboCup Rescue competitions in 2008 and 2009. The robot won the mapping challenge at the world championship in 2008 and was the only robot that was able to map the whole arena. In 2009, the robot won the RoboCup German Open, both the challenge for autonomous robots and the overall competition.

6 Future Work

So far, the roughness value are calculated and visualized, but not used in the following frontier selection and path planning step. The use of this value in the path planning is easy, because the Path Transform requires a cost function, which could easily be extended by the roughness value. Another subject of research is the utilization of neighborhood information during the classification of grid-cells.

References

[Elfes, 1989] Elfes, A.: Occupancy grids: A probabilistic framework for robot perception and navigation. PhD thesis, Carnegie Mellon University (1989)

[Kadous et al., 2006] Kadous, M.W., Sammut, C., Sheh, R.: Autonomous traversal of rough terrain using behavioural cloning. In: The 3rd International Conference on Autonomous Robots and Agents (2006)

[Lalonde et al., 2006] Lalonde, J.-F., Vandapel, N., Huber, D., Hebert, M.: Natural terrain classification using three-dimensional ladar data for ground robot mobility 23(1), 839–861 (2006)

[Montemerlo et al., 2008] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Hähnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., Thrun, S.: Junior: The stanford entry in the urban challenge. J. Field Robotics 25(9), 569–597 (2008)

[Nüchter, 2006] Nüchter, A.: Semantische dreidimensionale Karten für autonome mobile Roboter. PhD thesis, Rheinische Friedrich-Wilhelms-Universität Bonn (2006)

[Ohno et al., 2008] Ohno, K., Kawahara, T., Tadokoro, S.: Development of 3d laser scanner for measuring uniform and dense 3d shapes of static objects in dynamic environment. In: ROBIO 2008: IEEE International Conference on Robotics and Biomimetics, February 22-25, pp. 2161–2167 (2009)

[Pellenz, 2007] Pellenz, J.: Rescue robot sensor design: An active sensing approach. In: SRMED 2007: Fourth International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster, Atlanta, USA, pp. 33–37 (2007)

[Sheh et al., 2007] Sheh, R., Kadous, M., Sammut, C., Hengst, B.: Extracting terrain features from range images for autonomous random stepfield traversal. In: IEEE International Workshop on Safety, Security and Rescue Robotics, SSRR 2007, pp. 1–6 (2007)

[Thrun, 2006] Thrun, S.: Winning the darpa grand challenge: A robot race through the mojave desert. In: ASE 2006: Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering, p. 11. IEEE Computer Society, Washington (2006)

[Wirth and Pellenz, 2007] Wirth, S., Pellenz, J.: Exploration transform: A stable exploring algorithm for robots in rescue environments. In: Workshop on Safety, Security, and Rescue Robotics, pp. 1–5 (2007), `http://sied.dis.uniroma1.it/ssrr07/`