Towards Real Time Robot 6D Localization in a Polygonal Indoor Map Based on 3D ToF Camera Data

Jan Wülfing [*] **Joachim Hertzberg** [*] **Kai Lingemann** [*]
Andreas Nüchter [**] **Thomas Wiemann** [*] **Stefan Stiene** [*]

[*] *University of Osnabrück, Institute of Computer Science*
{*jwuelfin/hertzberg/lingemann/twiemann/sstiene*}*@informatik.uni-osnabrueck.de*
[**] *Jacobs University Bremen gGmbH, School of Engin. and Science*
a.nuechter@jacobs-university.de

Abstract: This paper reports a method and results for solving the following problem: Given a 3D polygonal indoor map and a mobile robot equipped with a 3D time of flight (ToF) camera, localize at frame rate the 6D robot pose with respect to the map. To solve the problem, the polygonal map is represented for efficient usage as a solid-leaf BSP tree; at each control cycle, the 6D pose change is estimated a priori from odometry or IMU, the expected ToF camera view at the prior pose sampled from the BSP tree, and the pose change estimation corrected a posteriori by fast ICP matching of the expected and the measured ToF image. Our experiments indicate that, first, the method is in fact real-time capable; second, the 6D pose is tracked reliably in a correct map under regular sensor conditions; and third, the tracking can recover from some faults induced by local map inaccuracies and transient or local sensing errors.

Keywords: Navigation, localization, 6D robot pose, trajectory tracking

1. MOTIVATION AND BACKGROUND

Robot maps of indoor environments have often been 2D occupancy maps in the past (Moravec and Elfes (1985); Thrun et al. (2005); LaValle (2006)), allowing robots to be localized in 3D poses (position and orientation). However, as mobile robots enter applications, robot control has to face the fact that real world is no 2D place. Avoiding safely obstacles and steps needs to regard full 3D information; mapping anything beyond flatland needs to use 3D maps; and localization in a 3D map has to consider 6D robot poses in general. In consequence, a considerable part of recent literature has turned to tackling this increase in dimensionality (see Nüchter (2009) for an introduction). 6D poses are also required beyond robotics if hand-held sensor modules get applied for sensing the environment.

Localization in 2D indoor environments is regarded as solved. That changes when dealing with localization in 3D, with six pose dimensions. Typical SLAM-generated maps from that body of work are 3D point clouds as originating from registered 3D laser scans (Nüchter (2009)). This is a useful sensing, representation and processing framework in some cases, but it has two drawbacks for being used in many service robot applications:

(1) Acquiring a single 3D scan takes a significant amount of time, causing problems to handle ego motion and independent motion in the scene;
(2) large 3D maps in the form of registered 3D scan point clouds are inherently memory-intensive and computing time intensive.

Polygonal 3D maps are a possible format for solving the efficiency problems, which is essential for large scale applications. They could be generated from 3D point cloud maps, e.g., by region growing and meshing approaches used in Computer Graphics (Hoppe et al. (1992)); alternatively, they could come from importing facility management software CAD building data. For acquiring 3D environment information in short time cycles during robot operation, two types of sensors get used in the robotics literature as alternatives to 3D scanners: stereo vision and 3D time of flight (ToF) cameras. Both could or should be used for real-time localization in a polygonal indoor map, too.

However, only little work has been presented so far about actually using polygonal maps. Morisset et al. (2009) describes a legged robot that builds such an environment representation and employs it for path planning. Rusu et al. (2008) presents a technique to extract cuboid representations of kitchen furniture from 3D laser scans.

Currently, indoor applications are solved either by using 2D maps, or by reducing 3D maps to 2D for localization, thus applying the same 2D methodology. This method does not work, however, when dealing with sloping ground, or, more common in indoor scenarios, with a robot that displays uneven movement – or with no robot at all: Localizing a sensor, e.g., as part of a hand-held device, require to deal with 6D movements and full 3D maps, all the while respecting efficiency constraints. This paper describes a system towards achieving this goal: We describe a method and results for real-time 6D localization of a mobile robot in a polygonal 3D map, given data from a 3D ToF camera. "Real-time" here means the frame rate of the 3D ToF camera. In this study, we have used a pose tracking approach based on blending a sensor data expectation at the esti-

mated prior pose with the actual ToF camera data in every time cycle, resulting in a posterior pose estimation. In the discussion part below, we will get back to the possibility of applying state of the art probabilistic methods like Monte Carlo localization (Thrun et al. (2005)) in our scenario.

The rest of the paper is organized as follows. Next, we sketch two technical ingredients used here. Then, the main component is introduced: the localization procedure itself, including the processing of the sensor data, the generation of simulated data in the map, and the matching of both. Experiments and results are described then, and a discussion concludes the paper.

2. TECHNICAL INGREDIENTS

First, we describe briefly two technical components that are important for the method described next: 3D ToF cameras as the sensor, and BSP trees as a data structure.

2.1 Time of Flight Cameras

ToF cameras have attracted roboticists' interest recently. They yield 3D point clouds at good frame rates and are compact and energy efficient. On the other hand, their resolution, measurement range and aperture are currently limited and their point clouds suffer from noise and artefacts. Yet, 3D ToF cameras have proven to be of use in robotics, such as for obstacle avoidance (Weingarten et al. (2004)), localization (Ohno et al. (2006)), and mapping (May et al. (2009)). Smart filter techniques reduce noise and artifacts in the point clouds to get reasonable results (Fuchs and May (2008)). ToF cameras can be calibrated using photogrammetric methods to reduce distortions.

All experiments for this paper were carried out with a O3D100 camera by PMDTechnologies, here referred to as *PMD camera*. It yields a resolution of 50×64 pixels at a frame rate of up to 24 Hz, subject to the environment-dependent integration time.

2.2 BSP Trees

For efficient localization, our polygonal map is stored in an optimized way as a *Solid Leaf BSP Tree* (Ericson (2005)). Binary Space Partitioning (BSP) trees recursively subdivide space into convex sets by hyperplanes, as described by Fuchs et al. (1979). For a given point, the Solid Leaf BSP tree allows the closest polygon in the model to be determined in $O(\log n)$ time, where n is the number of stored polygons in the tree. Additionally, it can efficiently be determined whether a given point is accessible, i.e., whether it is inside a defined area of the model. This excludes space within walls, pillars and such.

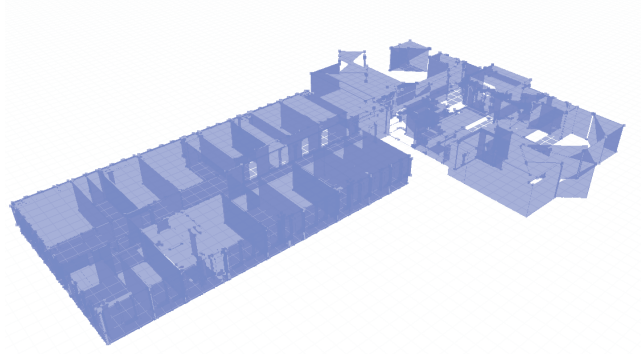


Fig. 1. A 3D polygonal model of our office floor at the University of Osnabrück. The model is hand-made based on laser distometer measurements.

3. THE LOCALIZATION PROCEDURE

The localization procedure follows the following scheme: Between two readings of the camera, the robot's pose is tracked by means of odometry or IMU information, resulting in the 6D prior pose estimate $\hat{\boldsymbol{P}}_t \in \mathbb{R}^6$ at time t. As soon as we receive new range data, a simulated point cloud based on $\hat{\boldsymbol{P}}_t$ is calculated, using the 3D polygonal map and a ToF camera sensor model. This produces the sensor data that is expected at $\hat{\boldsymbol{P}}_t$. Next, this expectation is matched with the actual sensor data, using an efficient version of the ICP algorithm. Finally, the quality of the matching result is evaluated and used to correct $\hat{\boldsymbol{P}}_t$, if found reliable, resulting in the posterior pose $\boldsymbol{P}_{C,t}$ at time t. This scheme is illustrated in Fig. 2. The following subsections unfold the details.

3.1 ToF Data Filtering

In a first step, the PMD data has to be preprocessed, since it suffers from significant noise. This has two main sources: Misreadings caused by reflections or external light, and artifacts that appear at depth discontinuities. In these situations, the sensor averages the distance measurements of two surfaces. To filter out these artifacts, we use a noise reduction algorithm by Huhle et al. (2008).

This algorithm evaluates each point by rating it according to the distance difference to its neighboring pixels. Points that are part of a homogenous area score higher than those with no or only few neighbors within a similar distance. If the score misses an empirically chosen threshold, the point is classified as an outlier and gets removed.

Fig. 3 shows the results of the filtering process. Data points in areas with varying point densities and without

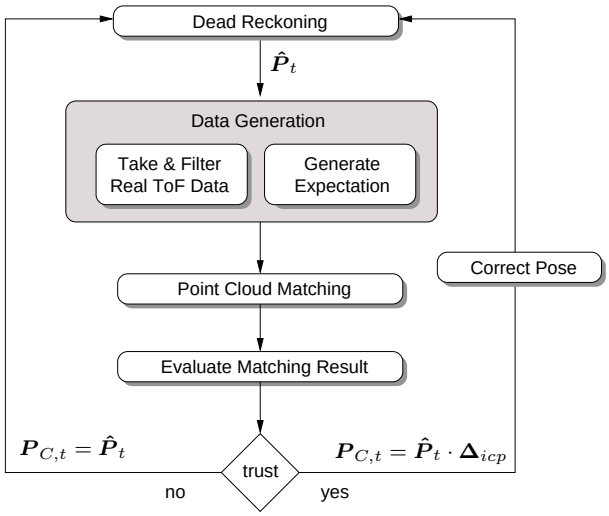


Fig. 2. The localization procedure's basic scheme.

neighbors are removed. The remaining points have a high probability of belonging to a physical surface.

3.2 ToF Camera Model for Generating Expected Data

Sensor Model. To create a correspondence between the ToF camera readings and the given 3D polygonal map, we create a virtual point cloud that samples the surfaces in the map, assuming the current prior pose estimate $\hat{\boldsymbol{P}}_t$. This process is based on a sensor model for the camera and a pose estimation. In our experiments, this pose estimation is delivered by the robot odometry. It could also be given by other sources, such as an IMU.

The ToF camera configuration is modeled as a pinhole camera with a mirrored image plane, cf. Fig 4. Accordingly, rays from the origin are cast $\boldsymbol{o}$ through a discrete grid on the image plane. Since the internal camera parameters (focal length, chip size, physical pixel size) are unknown, the model is parametrized by

- min and max vertical field-of-view angles (α, β),
- min and max horizontal field-of-view angles (ϕ, γ).

Using these, the coordinates of the corner vertices Z_1, Z_2, Y_1 and Y_2 of the mirrored image plane are calculated as

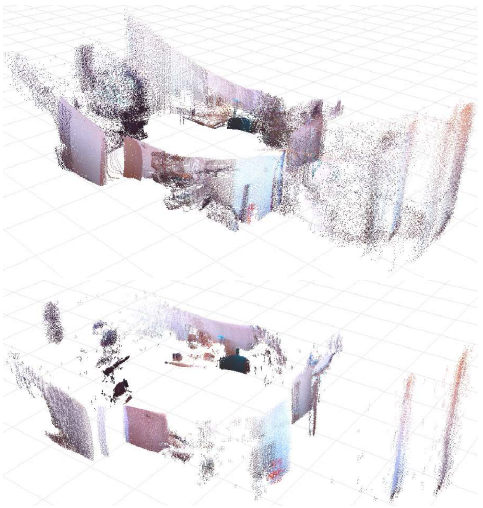


Fig. 3. Noise reduction in PMD camera data. Top: Perspective view of a single PMD image. Bottom: View of the same data with noise reduction applied as described.

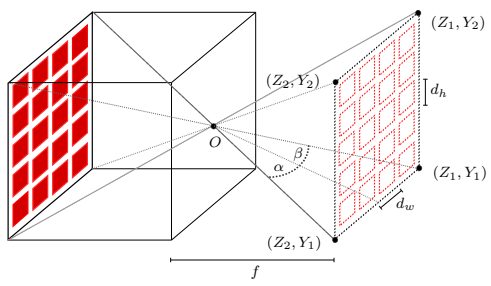


Fig. 4. Sketch of the PMD camera geometry with a mirrored image plane.

Table 1. PMD camera simulation parameters.

Resolution [px]		Range [m]		Field of view angles [°]			
r_V	r_H	t_{min}	t_{max}	α	β	ϕ	γ
64	50	0.0	7.5	24.9	19.7	22.8	11.6

$$Z_1 = \begin{pmatrix} f \\ -\tan\alpha \cdot f \end{pmatrix}, \qquad Z_2 = \begin{pmatrix} f \\ \tan\beta \cdot f \end{pmatrix},$$
$$Y_1 = \begin{pmatrix} f \\ -\tan\phi \cdot f \end{pmatrix}, \qquad Y_2 = \begin{pmatrix} f \\ \tan\gamma \cdot f \end{pmatrix}.$$

The focal length f has no influence on the result of the ray casting process, because if the field-of-view angles are known, it is unimportant where the image plane is positioned. For simplicity we therefore assume $f = 1$. Using the equations above, the specific dimensions (width w and height h) of our virtual PMD camera are:

$$w = d(\overline{Z_1 Z_2}) = \sqrt{1 + (-\tan\alpha - \tan\beta)^2},$$
$$h = d(\overline{Y_1 Y_2}) = \sqrt{1 + (-\tan\phi - \tan\gamma)^2}.$$

Based on the pixel resolution of the real camera (r_v, r_h), the virtual image plane is discretized with the pixel sizes

$$d_x = \frac{w}{r_h} \qquad d_y = \frac{h}{r_v}.$$

Ray Casting. Each discrete point $\boldsymbol{p}_i$ on the image plane produces a ray of light through $\boldsymbol{o}$ in direction $\boldsymbol{u}$. Thus, each direction vector $\boldsymbol{u}$ is transformed by a matrix $\boldsymbol{T}_{C,t} \in \mathbb{R}^{4\times4}$ that corresponds to the robot pose $\boldsymbol{P}_{C,t}$:

$$\begin{pmatrix} \boldsymbol{u}' \\ 1 \end{pmatrix} = \boldsymbol{T}_{C,t} \begin{pmatrix} \boldsymbol{u} \\ 1 \end{pmatrix}$$

where $\boldsymbol{u}'$ represents the transformed direction vector. For each direction vector $\boldsymbol{u}'$, a ray can be calculated,

$$\boldsymbol{r} = \boldsymbol{o} + t_{hit} \cdot \boldsymbol{u}', \tag{1}$$

where $\boldsymbol{r} \in \mathbb{R}^3$ denotes the ray, $t_{hit} \in \mathbb{R}$ is a scalar and $\boldsymbol{u} \in \mathbb{R}^3$ the direction vector. The point of origin $\boldsymbol{o} \in \mathbb{R}^3$ describes the camera position. Each ray is now intersected with the model, or, to be exact, with the polygon closest to $\boldsymbol{o}$ in the model, which is determined by traversing the BSP tree. This results in the parameter t_{hit} that can be used to calculate the intersection point $\boldsymbol{p} \in \mathbb{R}^3$,

$$\boldsymbol{p} = t_{hit} \cdot \boldsymbol{u}.$$

Note that in contrast to Eq. (1), $\boldsymbol{o}$ is omitted. Moreover, $\boldsymbol{u}$ rather than the rotated $\boldsymbol{u}'$ is used as the direction vector. This is due to the nature of the camera's raw data, which contains no information about the camera pose.

Fig. 5(a) contrasts the result of the ray casting procedure with the actual camera data. The real and the simulated cameras were both placed in front of a flat wall/polygon for calibration. Obviously, both point clouds share a common shape. Their resolution matches, too, but they cover different areas of the wall, due to (initially) incorrect field-of-view angles. The correct parameters are then determined by modification until the two datasets matched. The result of this calibration is shown in Fig. 5(b), Table 5 displays the empirically determined parameters for the PMD ToF camera used in our experiments.

Here both point clouds match almost perfectly. If the estimated pose is very accurate, these parameters provide

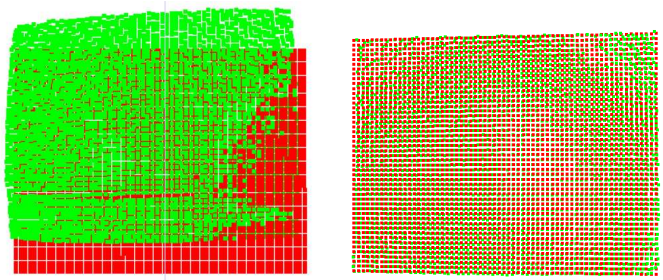
the most efficient way of simulating the point cloud. In practice, however, the odometry-based pose estimation $\hat{\boldsymbol{P}}_t$ is most likely erroneous. In the extreme, this could result in losing the pose if the simulated point cloud would not overlap the real one. To exclude this problem, we increase field of view and resolution of the simulated sensor.

3.3 Point Cloud Matching

The simulation result is a point cloud representing what the ToF camera is expected to sense at the estimated pose. To correct the pose estimation, a transformation $\boldsymbol{\Delta}_{icp} \in \mathbb{R}^{4\times4}$ (consisting of a rotation $\boldsymbol{R}$ and a translation $\boldsymbol{t}$) between the expected and the measured point clouds is computed. This transformation is determined using an optimized ICP implementation Nüchter et al. (2007).

3.4 Evaluation of the Matching Result

Optimally, i.e., if the map were correct, robot motion between two frames were small, and the environment were free of features spoiling the camera image, $\boldsymbol{\Delta}_{icp}$ would quantify the prior pose estimate error precisely. However, reality is suboptimal. For example, unmapped obstacles may cause differences between the expected and the measured point cloud, resulting in matching errors. Specular objects like mirrors or windows may cause noise; this can be reduced by the noise reduction procedure mentioned in Sec. 3.1, but at the cost of losing resolution.



(a) (b)

Fig. 5. Camera model calibration. Simulation (red), PMD camera (green). (a) Simulation with incorrect angles. (b) Simulation with correct angles after calibration.

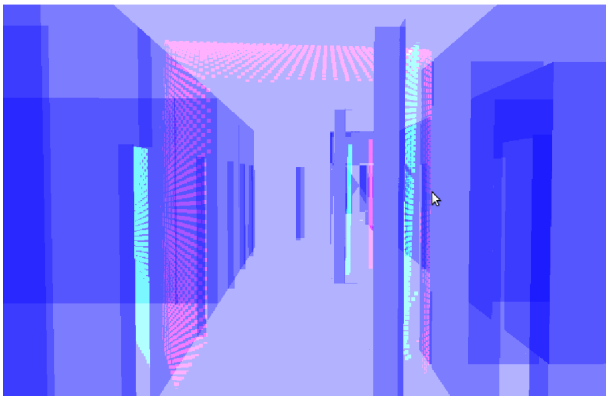


Fig. 6. 3D map of the environment, with simulated (pink) and real scan (turquoise), before matching.

In consequence, the ICP matching result is checked heuristically, before being applied to correct the prior pose estimate. If it is likely to err, it gets discarded and the prior pose $\hat{\boldsymbol{P}}_t$ estimate is used as the posterior, i.e.,

$$\boldsymbol{P}_{C,t} = \hat{\boldsymbol{P}}_t.$$

The heuristic evaluation is based on the following criteria:

- $i_{icp} \in \mathbb{N}$ denotes the number of iterations needed to determine the ICP result. If it equals 0, the ICP algorithm was unable to find *any* point pairs and obviously did not compute a transformation at all.
- $n_{cam} \in \mathbb{N}$ represents the number of points in the real camera's point cloud. If the data is noisy, many points are removed by the noise reduction and n_{cam} is small. The ICP result is rejected, if n_{cam} fails to exceed an empirically chosen threshold t_{cam}. In practice,
$$t_{cam} = \frac{r_V r_H}{2} \tag{2}$$
has been reasonable, where r_V and r_H represent the vertical and horizontal resolutions of the sensor.
- $n_{icp} \in \mathbb{N}$ represents the number of corresponding point pairs found by ICP in the last iteration. If n_{icp} is significantly smaller than n_{cam}, the ICP result is possibly erroneous. This phenomenon would most likely occur when the real world has an obstacle that is not in the map. The transformation is rejected if
$$n_{icp} < \frac{n_{cam}}{2} = 1600,$$
which again has been empirically determined.
- $e_{icp} \in \mathbb{R}$ denotes the mean distance between all corresponding point pairs in the last iteration. Our rejection criterion is chosen with respect to the accuracy of the map and the noise of the used sensor.
- Finally, the plausibility of the pose update is checked. This is achieved by applying $\boldsymbol{\Delta}_{icp}$ to the robot's pose and performing a "pose-in-solid-space" test, which makes sure that the updated pose is in free space rather than inside an object in the map. If the pose were in *solid* space, the update is rejected.

If the ICP transformation is trusted, it is applied to the pose estimation $\hat{\boldsymbol{P}}_t$ and generates the posterior pose:

$$\boldsymbol{P}_{C,t} = \hat{\boldsymbol{P}}_t \boldsymbol{\Delta}_{icp}.$$

4. EXPERIMENTS

We have tested the accuracy of our procedure in several experiments. In this section we will present two examples to demonstrate the ability of tracking the robot pose in 6D. The first example demonstrates planar pose tracking by driving a closed loop in a classroom. The second one extends pose tracking to a non-planar environment by driving over a ramp, thus introducing true 6D poses.

4.1 Pose Tracking in a classroom

The first experiment was done in a classroom of size 7.0 m × 4.7 m. Fig. 7 shows the setup and results. The robot was teleoperated to drive a loop, starting and ending in the lower right corner. The turning points were marked on the floor to make the experiments reproducible and have ground truth. The PMD camera was mounted with

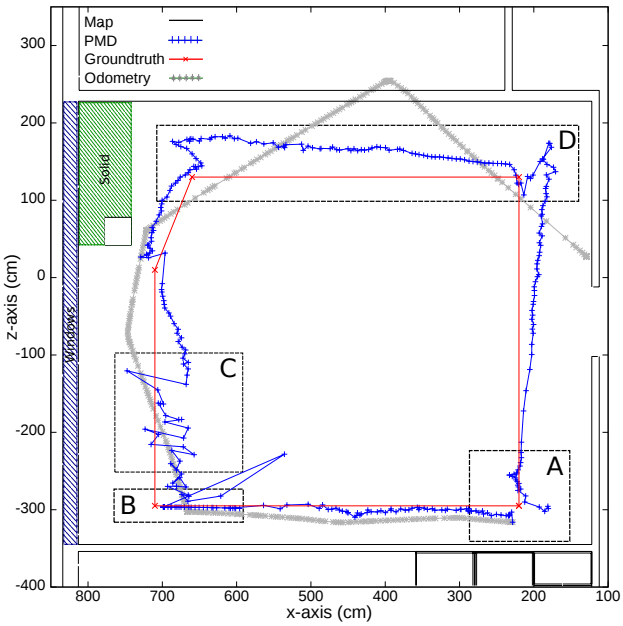


Fig. 7. Setup and results of the classroom experiment. The map and trajectories are shown in projection from top (x and z values).

an upwards tilt (about 30°). The 3D map was hand-modeled and is, therefore, subject to inaccuracies: In the area labeled green, the room was modeled as solid space, where there is really just a small pillar. Another map fault was the presence of a unmapped blackboard protruding into the room by about 35 cm (area D). The left wall contains several windows, indicated by the blue marking.

The results in this environment proved to be highly reproducible. The proposed algorithm outperforms pure odometry. The loop closure was almost successful (see A). Start and end position differ by about 50 cm, which is in the order of magnitude of one robot length. However, three sections in the trajectory deserve further discussion.

Section B shows a trajectory that is smooth compared to the others. This effect is due to the windows in the left wall. In this region the PMD suffers from significant noise. More and more points get filtered out, and the point clouds become too sparse to find a confident ICP match, so these pose corrections are rejected and the pose update is based on odometry alone. Another severe disturbance occurs while driving in parallel to the windows where reflections distort the results of the ICP matching (section C). However, as soon as a more benign section is reached, the camera based localization takes on the job again. Despite the inaccuracy of the map in the upper left corner, the algorithm is able to correct the pose until the estimation nearly matches ground truth.

In C the trajectory deviates significantly but stably from ground truth. This is due to the blackboard. It is perceived by the PMD camera, but not present in the map. In consequence the real blackboard and the mapped wall are matched, resulting in the obvious displacement. After the area with the faulty map is left, the localization recovers.

4.2 Pose Tracking involving a Ramp

In the second experiment, localization in true 6D is tested. The robot had to drive over a ramp raising by ca. 30 cm at a pitch angle of ca. 10°. The localization result is compared to a run on the flat floor. Fig 8 shows the results. Label A covers the part of the trajectory in front of the ramp. Here, both results show strong, but uniform deviations from ground truth. As soon as the robot reaches the ramp, the result becomes stable and almost exactly resembles ground truth. Without the ramp, the localization results climb constantly until they reach ground truth. In section C, the robot approaches a wall without recognizable features. Thus, the height information can no longer be determined and the estimation error accumulates resulting in an increasing height estimation.

The localization results on the ramp become stable as soon as ceiling points get into view. Even without prominent features, the matching between the simulated and the measured point cloud succeeds because of the camera tilt angle. Odometry gives no height information at this point. Therefore, the simulated results would expect a constant height of the ceiling, which results in a detectable offset between the perceived and the simulated point clouds that can be determined by ICP.

4.3 Performance

The desired real-time capability of the procedure means in practice a cycle time lower than the shutter speed of the PMD camera, which itself depends on the integration time. In the experiment, we have achieved this. In general, the cycle time depends mainly on two components: generating the simulated point cloud, and ICP matching. Based on BSP trees, the simulation time is logarithmic in the number of map polygons. For the small point clouds involved here (remember that the resolution of the O3D100 is 64×50), the time for ICP matching using the optimized ICP implementation (Nüchter et al. (2007)) is negligible.

5. DISCUSSION

The state of the art localization approach for mobile indoor robots is arguably Monte-Carlo localization in a 2D occupancy map based on 2D laser scans (Thrun et al. (2005)). We have done three related things differently here: Used a 3D ToF camera, used a polygonal 3D map, and used maximum-likelihood mono-modal pose tracking. We will address the three deviations from the state of the art in turn, and finally reflect on what we have achieved.

3D ToF cameras are around for a while now. They are an attractive type of sensor for robotics in principle, but current exemplars have a number of drawbacks compared to other sensors, as stated above (Sec. 2.1). For localization based on the PMD O3D100, the combination of low aperture and limited range (7.5 m) is bad: Frequently, all that the sensor sees is a flat wall, allowing precise localization regarding one horizontal dimension only. Ideally, concave wall/wall or wall/ceiling edges should be tracked, but the range limit makes this often impossible. In consequence, the pose estimation may drift away in some dimensions. It will typically recover, but if precise localization is needed

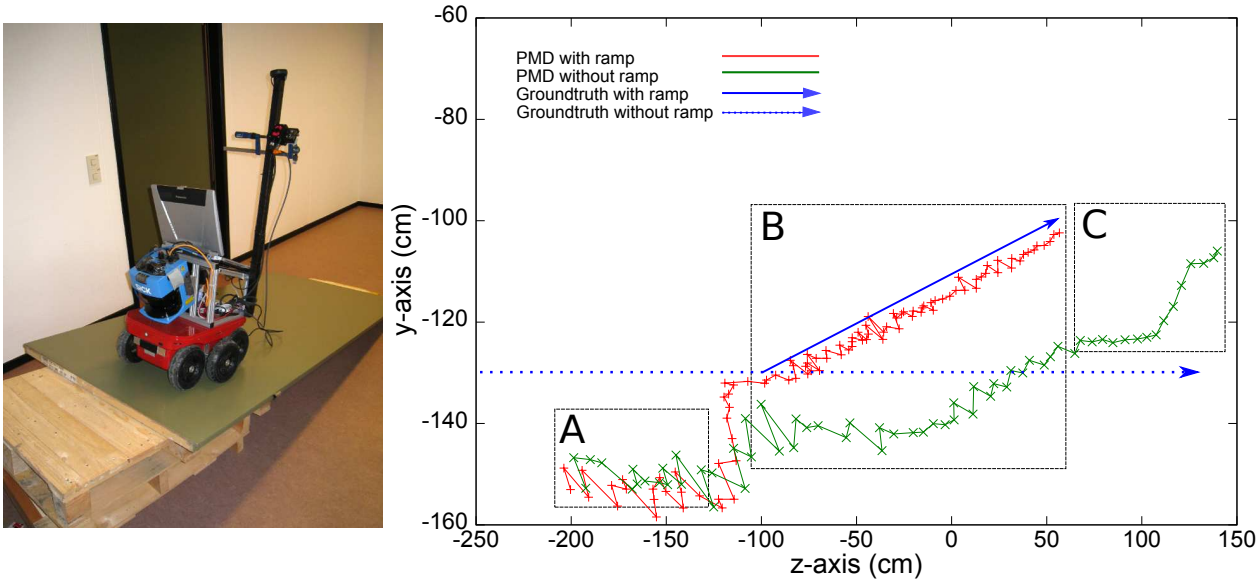


Fig. 8. Setup and results of the "ramp" experiment. The Trajectories are shown in projection from the left side (y and z values). In the beginning (section labeled A), the initial y pose estimation was wrong. When the robot enters the ramp, the ceiling comes into the view of the camera, and the pose is corrected successfully.

at any time, fusion with other sensor data is mandatory for today's 3D ToF cameras.

Using 3D polygonal maps is uncommon in robotics. Yet, as generating 3D point cloud maps is under intensive research, as point cloud maps are costly, and as polygonal maps are a natural choice for human-made environments, they are moving into focus. In our experiments, they have kept the promise of computation and storage efficiency. The approach of matching an expected data sample with the actual 3D camera data for generating a posterior pose estimate, exploits the real data well and has proven robust against map inaccuracies. Matching methods other than ICP matching point clouds are possible: For example, principal planes could be identified in the 3D ToF data and fitted with the polygonal map. This is future research.

Finally, our mono-modal pose tracking approach is obviously compatible with multi-modal particle filter localization. A particle filter version could be realized by sampling a prior pose distribution from odometry and/or IMU data, and basically applying our method locally for each particle. The evaluation described in Sec. 4 would have to be turned into estimating the posterior pose distributions per particle. So we are not arguing for localization by mono-modal pose-tracking, but our experiment rather serves for formulating the localization method for 3D ToF camera data and polygonal maps, independent of its mono- or multi-modal usage. This will be examined in future research.

We have presented a method for 6D localization using 3D ToF camera data. Current cameras suffer from some technical deficits, and so, in consequence, does the localization performance. We reckon these deficits will go away, and then 6D localization will be solved naturally with their data. Our experiments make plausible that the localization can recover from faults induced by unfavorable sensing conditions, moderate map faults and prior pose errors. Polygonal maps are the map format of choice here, owing to their storage and time efficiency. Our experiment shows that they are a handy format for real-time 6D localization.

REFERENCES

Ericson, C. (2005). *Real Time Collision Detection*. Elsevier.

Fuchs, Kedem, and Naylor (1979). Predetermining Visibility Priority in 3-D Scenes. *SIGGRAPH*, 175–181.

Fuchs, S. and May, S. (2008). Calibration and registration for precise surface reconstruction with time-of-flight cameras. *IJISTA*, 5(3/4), 274–284.

Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., and Stuetzle, W. (1992). Surface reconstruction from unorganized points. *Computer Graphics*, 26(2), 71–78.

Huhle, B., Jenke, P., and Strasser, W. (2008). On the fly scene acquisition with a handy multisensor system. *Int. J. Intell. Syst. Technol. Appl.*, 5(3/4), 255–263.

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, U.K.

May, S., Droeschel, D., Holz, D., Fuchs, S., Malis, E., Nüchter, A., and Hertzberg, J. (2009). Three-dimensional mapping with time-of-flight cameras. *J. Field Robotics*, 26(11-12), 934–965.

Moravec, H. and Elfes, A.E. (1985). High resolution maps from wide angle sonar. In *Proc. ICRA*, 116–121.

Morisset, B., Rusu, R., Sundaresan, A., Hauser, K., Agrawal, M., Latombe, J., and Beetz, M. (2009). Leaving flatland. toward real-time 3d navigation. In *Proc. ICRA*.

Nüchter, A., Lingemann, K., Hertzberg, J., and Surmann, H. (2007). 6D SLAM – 3D Mapping Outdoor Environments. *J. Field Robotics*, 24(8/9), 699–722.

Nüchter, A. (2009). *3D Robotic Mapping – The Simultaneous Localization and Mapping Problem with Six Degrees of Freedom*, volume 52 of *STAR*. Springer, Heidelberg.

Ohno, K., Nomura, T., and Tadokoro, S. (2006). Real-time robot trajectory estimation and 3d map construction using 3d camera. In *Proc. IROS*, 5279–5285.

Rusu, R.B., Marton, Z.C., Blodow, N., Dolha, M.E., and Beetz, M. (2008). Functional Object Mapping of Kitchen Environments. In *Proc. IROS*. Nice, France.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press.

Weingarten, J., Gruener, G., and Siegwart, R. (2004). A State-of-the-Art 3D Sensor for Robot Navigation. In *Proc. IROS*.