Depth Camera based Localization and Navigation for Indoor Mobile Robots

Joydeep Biswas
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
joydeepb@ri.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
mmv@cs.cmu.edu

***Abstract*—We present here the Fast Sampling Plane Filtering (FSPF) algorithm, which reduces the volume of the 3D point cloud by sampling points from the depth image, and classifying local grouped sets of points as belonging to planes in 3D (called the "plane filtered" points) or points that do not correspond to planes (the "outlier" points). We present a localization algorithm based on an observation model that down-projects the plane filtered points on to 2D, and assigns correspondences for each point to lines on the 2D map. The full sampled point cloud (consisting of both plane filtered as well as outlier points) is processed for obstacle avoidance for autonomous navigation. We provide experimental results demonstrating the effectiveness of our approach for indoor mobile robot autonomy. We further compare the accuracy in localization using 2D laser rangefinders *vs.* using 3D depth cameras.**

I. Introduction and Related Work

Given that indoor mobile robots have limited onboard computational power, there are two immediate hurdles to using depth cameras for mobile robot autonomy:

1) Depth cameras typically generate voluminous data that cannot be processed in its entirety in real time for localization
2) Techniques for localization and mapping (*e.g.* occupancy grids) in 2D do not scale well in terms of memory and computational complexity for use in 3D. Conversely, the problem of mapping 3D observations to existing 2D maps is not straightforward.

We tackle both these problems using the Fast Sampling Plane Filtering (FSPF) algorithm that samples the depth image to produce a set of points corresponding to planes, along with the plane parameters (normals and offsets). The filtered point cloud is then used to localize the robot on an existing 2D vector map. The sampled points are also used to perform obstacle avoidance for navigation of the robot. To illustrate the key processed results, Fig. 1 shows a snapshot using a single depth image after plane filtering, localization, and computing the obstacle avoidance margins.

Map building using raw point clouds [7, 4] in general scale poorly in terms of memory requirements with the density of point clouds and the size of maps. Approaches to map building using planar features extracted from 3D point clouds [8, 3] have been explored in the past. 3D Plane SLAM [5] is a 6D SLAM algorithm that uses observed 3D point clouds to construct maps with 3D planes. The plane detection in their work relies on region growing [6] for plane extraction, whereas our approach uses sampling of the depth image. In addition, our observation model projects the observed planes onto the existing 2D vector map used for 2D laser rangefinder sensors.

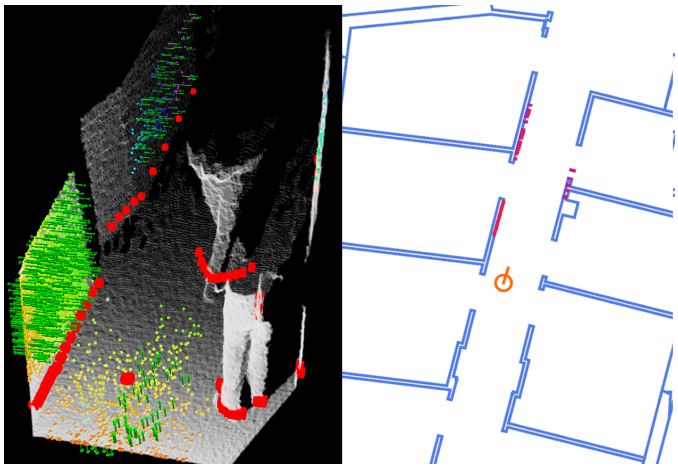


Fig. 1. Snapshot of depth image processing: On the left, the complete 3D point cloud is shown in white, the plane filtered 3D points in color along with plane normals, and the obstacle avoidance margins denoted by red boxes. On the right, the robot's pose is shown on the vector map (blue lines), with the 3D point correspondences shown as red points.

II. Fast Sampling Plane Filtering

The Fast Sampling Plane Filtering (FSPF) algorithm samples random neighborhoods in the depth image, and in each neighborhood, it performs a RANSAC based plane fitting on the 3D points. Thus, it reduces the volume of the 3D point cloud by extracting geometric features in the form of planes in 3D while being robust to outliers.

FSPF takes the depth image I as its input, and creates a list P of n 3D points, a list R of corresponding plane normals, and a list O of outlier points that do not correspond to any planes. FSPF proceeds by first sampling three locations d_0,d_1,d_2 from the depth image. The first location d_0 is selected randomly from anywhere in the image, and d_1 and d_2 are selected randomly within a neighborhood of size η around d_0. The 3D coordinates for the corresponding points p_0, p_1, p_2 are then computed. A search window of width w' and height h' is computed based on the mean depth (z-coordinate) of the points p_0, p_1, p_2, and the minimum expected size S of the planes in the world. Additional $l-3$ local samples d_j are then sampled from the search window to obtain a total of l

local samples. The plane fit error for the reconstructed 3D point p_j from the plane defined by the points p_1, p_2, p_3 is computed to determine if it as an "inlier." If more than $\alpha_{in} l$ points in the search window are classified as inliers, then all the inlier points are added to the list P, and the associated normals to the list R. This algorithm is run a maximum of m_{max} times to generate a list of at most n_{max} 3D points and their corresponding plane normals.

III. Localization and Navigation

A. Localization

Localization using the Plane filtered point cloud is performed using Monte Carlo Localization (MCL) [2] and Corrective Gradient Refinement (CGR) [1].

Since the map on which the robot is localizing is in 2D, the 3D filtered point cloud P and the corresponding plane normals R are first projected onto 2D to generate a 2D point cloud P' along with the corresponding normalized normals R'. Points that correspond to ground plane detections are rejected at this step. Let the pose of the robot x be given by $x = \{x_1, x_2\}$ where x_1 is the 2D location of the robot, and x_2 its orientation angle. The observable scene lines list L is computed using an analytic ray cast. The observation likelihood $p(y|x)$ (where the observation y is the 2D projected point cloud P') is computed as follows:

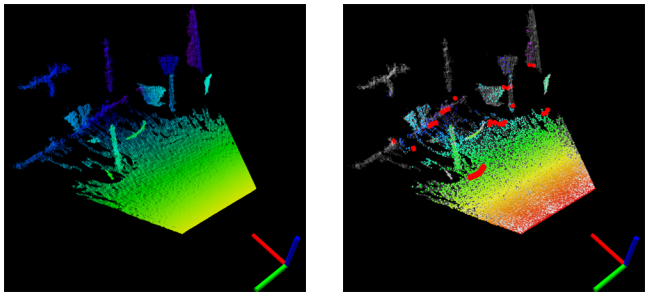
1) For every point p_i in P', line l_i ($l_i \in L$) is found such that the ray in the direction of $p_i - x_1$ and originating from x_1 intersects l_i.
2) Points for which no such line l_i can be found are discarded.
3) Points p_i for which the corresponding normal estimates r_i differ from the normal to the line l_i by a value greater than a threshold θ_{max} are discarded.
4) The perpendicular distance d_i of p_i from the (extended) line l_i is computed.
5) The total (non-normalized) observation likelihood $p(y|x)$ is then given by:

$$p(y|x) = \prod_{i=1}^{n} \exp\left[-\frac{d_i^2}{2f\sigma^2}\right] \tag{1}$$

Here, σ is the standard deviation of a single distance measurement, and $f : f > 1$ is a discounting factor to discount for the correlation between observed points. The observation likelihoods and their gradients thus computed are used to update the localization using CGR.

B. Navigation

For the robot to navigate autonomously, it needs to be able to successfully avoid obstacles in its environment. This is done by computing open path lengths available to the robot for different angular directions. Obstacle checks are performed using the 3D points from the sets P and O. Given the robot radius r and the desired direction of travel θ_d, the open path length $d(\theta)$ as a function of the direction of travel θ, and hence the chosen obstacle avoidance direction θ^* are calculated as:



(a) **(b)**

Fig. 2. Obstacle avoidance: The raw 3D point cloud (a) and (b) the sampled points (shown in color), along with the open path limits (red boxes). The robot location is marked by the axes.

$$d(\theta) = \min_{p \in P \cup O} \left(\max(0, ||p \cdot \hat{\theta}|| - r)\right) \tag{2}$$

$$\theta^* = \arg\max_{\theta} \left(d(\theta)\cos(\theta - \theta_d)\right) \tag{3}$$

Here, $\hat{\theta}$ is a unit vector in the direction of the angle θ, and the origin of the coordinate system is coincident with the robot's center. Fig. 2 shows an example scene with two tables and four chairs that are detected by the depth camera. Despite randomly sampling (with a maximum of 2000 points) from the depth image, all the obstacles are correctly detected, including the table edges. The computed open path lengths from the robot location are shown by red boxes.

IV. Experimental Results

Our experiments were performed on our custom built omni-directional indoor mobile robot, equipped with the Microsoft Kinect sensor. To compare the accuracy in localization using the Kinect, we also used a Hokuyo URG-04LX 2D laser rangefinder scanner. Localization using the laser rangefinder was performed using MCL [2] and CGR with a 2D point cloud sensor model [1].

Fig. 3 shows the mean errors in localization for the Kinect and laser rangefinder sensors while traversing a path 374m long. Although localization using the Kinect is not as accurate as when using the laser rangefinder, it is still consistently accurate enough for the purposes of indoor mobile robot navigation, with a mean offset error of less than 20cm and a mean angular error of about $1.5°$. It is worth noting, however, that the Kinect sensor is an order of magnitude cheaper than the laser rangefinder, and hence would still be the more cost-effective solution for robots on a lower budget.

To test the robustness of the depth-camera based localization and navigation solution, we set a series of random waypoints for the robot to navigate to, spread across the map. The total length of the path was just over 4.1km. Over the duration of the experiment, only the Kinect sensor was used for localization and obstacle avoidance. The robot successfully navigated to all waypoints, but localization had to be reset at three locations, which were in open areas of the map where Kinect sensor

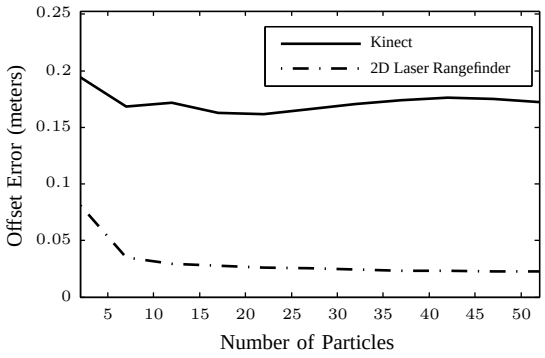


Fig. 3. The offset errors from the true true robot locations while localizing using the Kinect *vs.* Laser Rangefinder

Fig. 4. Trace of robot location for the long run trial. The locations where localization had to be reset are marked with crosses.

could not observe any walls for a while. Fig. 4 shows the trace of the robot's location over the course of the experiment.

REFERENCES

[1] Corrective Gradient Refinement for Mobile Robot Localization. *To Appear in IROS 2011*.

[2] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the National Conference on Artificial Intelligence*, pages 343–349. JOHN WILEY & SONS LTD, 1999.

[3] P. Kohlhepp, P. Pozzo, M. Walther, and R. Dillmann. Sequential 3D-SLAM for mobile action planning. In *IROS 2004*.

[4] A. Nuchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6d SLAM with approximate data association. In *ICAR 2005*.

[5] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga. Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics 2010*.

[6] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak. Fast plane detection and polygonalization in noisy 3D range images. In *IROS 2008*.

[7] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *ICRA 2000*.

[8] J. Weingarten and R. Siegwart. 3D SLAM using planar segments. In *IROS 2006*.