

Robust 3D Monte Carlo Localization using an RGB-D Camera and a Semantic Building Model

Maurice Fallon and John Leonard

Abstract—This paper presents a system for the robust localization of an RGB-D Camera, such as a Microsoft Kinect, in 3D. Our approach first focuses on the extraction of a low fidelity 3D model of the area of operation made up of semantic planar segments. In particular this emphasizes the extraction of major planes which are most likely to remain static (such as walls, floors, ceilings) while excluding non-planar point clouds (such as furniture) which we deem to be clutter and more likely to move in subsequent operation. Utilizing planar primitives, rather than voxel grids, is an important decision because (1) it allows for efficient 3D mapping relative to robot poses during exploration and (2) directly represents the real world structure. Using this map as input, an efficient Monte Carlo Localization algorithm is proposed which utilizes visual odometry (as the particle propagation mechanism) and coarse shape and color (to form the likelihood function). Demonstration of this approach is provided using an array of robots and human-mounted platforms. The application area of this approach is wide: including not just traditional robotic localization but many wearable and virtual reality applications.

I. INTRODUCTION

Localization in a previously mapped environment is a key skill to enable lifelong robotic operation. Previous research has studied this problem with 2D LIDAR range finders: demonstrating localization in a 2D occupancy gridmaps [1], [2]. However these sensors typically cost several thousand dollars — a figure which is not suitable, for example, for the low-cost domestic robotic market. Instead this paper describes ongoing work to develop a robust and accurate localization algorithm for the significantly cheaper RGB-D Camera such as a Microsoft Kinect¹ in 3 dimensions.

This paper is split into two parts. In Section II we discuss the attributes of a (minimal) 3-D planar map of an indoor environment intended for localization and then demonstrate how the map can be generated using RGB-D data from a Kinect. In Section III a Monte Carlo Localization algorithm is outlined which utilizes only the RGB-D data to reliably position the sensor within this map. Finally a series of experimental demonstrations are presented in Section IV using several platforms — robotic and man-portable.

II. 3D MAP BUILDING

a) Pose Estimation: In this section we outline a procedure for the extraction of a 3D building model using a robot pose estimated using Simultaneous Localization and Mapping. SLAM is a fundamental subfield of robotics and

The authors are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA
mfallon, jleonard@mit.edu

¹In this work we will typically refer to the Kinect, but other sensors based on infrared projection are in development

Algorithm 1: Kinect Plane Extraction

Given a Kinect RGB-D point cloud;
First discard all points beyond 5m range (and hence in the inaccurate non-linear range of the sensor);
Downsample point cloud using voxel tree with leaf of size 30cm;
while 30% of points remain or the last extracted plane was greater than $1m^2$ **do**
 Extract the largest plane (with points within 0.03m) from the remaining cloud using RANSAC;
 Remove any disconnected points from this plane;
 if the plane area exceeds $1m^2$ **then**
 Retain the plane, its coefficients, centroid and pose;
 The plane color is determined to be the median color of the original points;

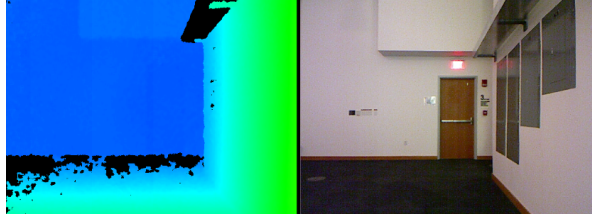
has progressed from the initial work of Smith, Self and Cheeseman [3] through EKF and later particle filter SLAM, such as [4], to non-linear least squares optimization of the entire robot trajectory and observation set — so called full SLAM. The current state of art, such as iSAM [5] and HogMan [6], provide efficient and incremental smoothing via non-linear optimization of the underlying robot trajectory when given a series of measurement constraints — such as those provided by LIDAR scan-matching. Very accurate, large scale 2D maps can be created in this way.

At this initial stage of our project we will utilize a LIDAR-estimated trajectory created in this way to infer the motion of the RGB-D sensor (rigidly connected to the Hokuyo UTM-30LX on the push cart illustrated in Figure 5). Doing so allows us to simplify the 3D map building procedure and to focus on RGB-D-only localization in the later part of this paper². The LIDAR-estimated pose is indicated in Figure 2.

b) Map Extraction: Using this accurate estimate of the Kinect sensor trajectory, we now wish to extract the planar semantic information from the RGB-D data. We assert that the largest planar polygons in any indoor environment are those that are (a) typically stationary from day to day (b) permanent structural features and (c) sufficient to form a recognizable 3D model of an area. Our thesis is that this type of model is sufficient for accurate robot localization in 3 dimensions.

²Ongoing visual SLAM research, such as [7], [8], would allow us to drop the requirement for a LIDAR sensor in the map building module in future.

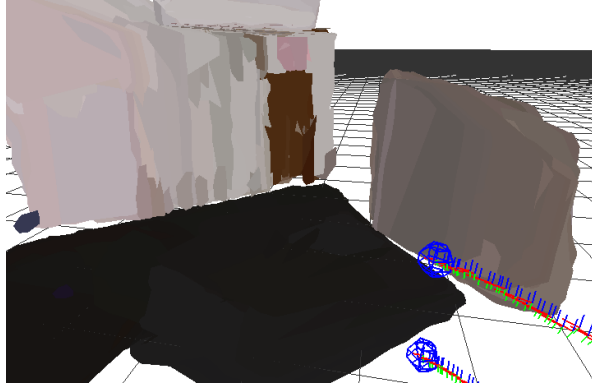
We present in Algorithm 1 a series of steps which extracts these planes from successive scans of RGB-D data taken by the sensor moving through an indoor building. Typically this procedure results in the extraction of 3–8 large planes representing the major planar objects within the sensor field of view. The location of the plane polygons can then be converted into the global reference frame using the aforementioned sensor pose. This procedure is then repeated for each successive scan within 5m of a central pose. Typically this results in multiple observations of the same plane, as large planes are typically observed many times as the sensor passes.



(a)



(b)



(c)

Fig. 1. Plane extraction procedure: A frame of raw kinect data (top) is first projected into a 3D point cloud (middle). Then large planar objects are successively extracted (bottom, for several consecutive frames). Also shown are the sensor pose triads.

A second algorithm (Algorithm 2) is then required which combines each planar observation into a final representative planar polygon representing a wall, floor, door or ceiling. The set of all such planes within this region then represents an

Algorithm 2: Planar Submap Determination

Given a set of overlapping planes;

Choose a center point for the submap (eg the pose of the sensor at some time);

Find the set of planes whose centroids are within a 5m of this point;

while *planes remain in the original set* **do**

 Choose a plane, incrementally find the set of planes which intersect it, using these conditions;

- Both plane normals are within 4 degrees;
- 2nd plane polygon can be projected onto the first with less than 5cm of translation;
- The percentage overlap of the projected points is greater than 80% ;

for all *accepted planes* **do**

 Merge into a single new plane, re-estimating the centroid and the plane coefficients and Convex Hull polygon — weighted by the number of points of the originating cloud;

 Remove all merged planes from the original stack;

accurate submap for this area, again illustrated in Figure 1. Submaps are overlapped slightly so that approximately 30 submaps each containing about 30 planes can represent an entire building floor of MIT’s Stata Center.

A. Model Utility and Storage Requirements

In this section we will overview some design properties of this map and explain why we believe it to be a useful representation.

Other efficient 3D mapping and storage systems have been proposed and implemented — often building on the octree data structure. For example OctoMap [9] demonstrated efficient octree-based occupancy grid mapping allowing for compact storage of large scale maps with minimal memory and disk storage requirements. Their representations of the New College dataset of 40,000m² requires 50 MB of memory or 1 MB of disk space using OctoMap.

However, our thesis is that by utilizing a fixed grid of points the resultant octree is in a sense ‘baked in’ and disconnected from the underlying robot pose used to construct it. It is difficult to adjust the structure of the tree — should a loop closure be detected by an exploring robot. In comparison the planar polygons are connected to poses in the pose graph optimization mentioned previously by a relative transformation. Should loop closures or adjustments be detected the location of any of the planes can easily be adjusted — updating and improving the 3D model.

Our representation is also efficient — we project that it will require only 2 MB of disk space to represent the entire 70,000m² 9 floor MIT Stata Center. It intentionally does not represent small furniture, chairs, tables etc. This type of information is typically not required for long-range path planning and is often transient in any particular location. Semantic labeling of the representation is also feasible - at-

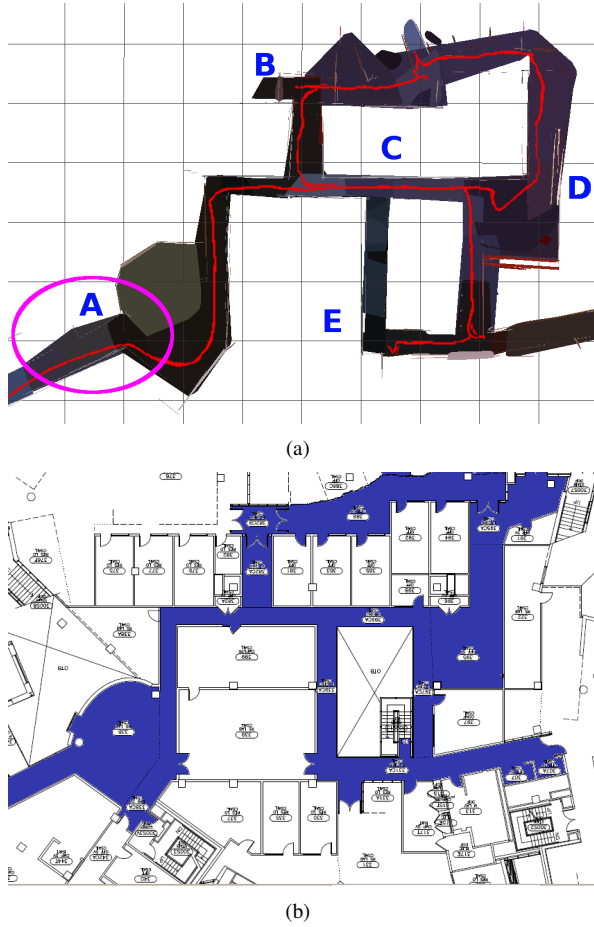


Fig. 2. Using the sensor pose estimated from the LIDAR (red), a global estimate of the plane locations can be inferred and an accurate 3-D model of the entire floor can be constructed as shown in this topdown view (top). Note the accuracy of the top-down model relative to the building floor-plan (bottom). The size of a typical submap is indicated by the purple loop.

taching the building connectivity graph or the room function to room-sized submaps.

Nonetheless, octree occupancy maps have many uses in path planning and free space detection. We believe that algorithms can be developed to transform between the two representations to allow for complementary use.

Finally, we do not claim that the procedure outlined here is in anyway optimal and we can ourselves envisage a number of optimizations to improve the appearance of these 3D plane models. However using this semantic map representation we aim to demonstrate accurate localization using only the Kinect sensor.

III. RGB-D MONTE CARLO LOCALIZATION

Having created the planar map, we propose to utilize particle filter localization to estimate the pose of a robot moving in this environment. Particle Filtering, more generally known as Sequential Monte Carlo (SMC), was initially proposed by Gordon et al. [10] and is well reviewed in [11]. Initial adaptations of SMC to robot localization were later reported [1], [2] using laser range finders and optionally

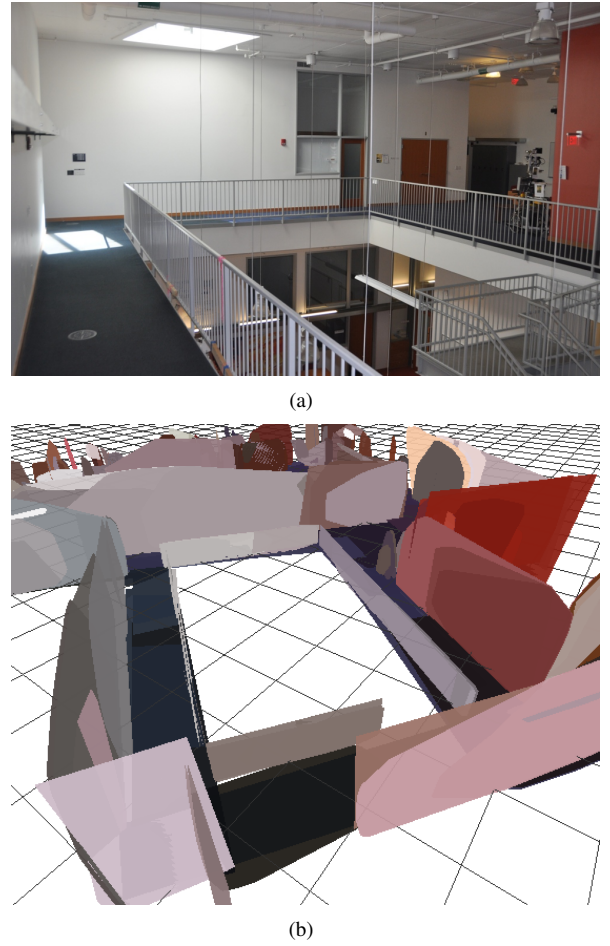


Fig. 3. Illustration of a section of the 3-D building model. Note that the model is intended to indicate the geometric information - not the precise textual information. Plane colors are for illustration purposes only at this stage.

wheel odometry to localize in 2D. Through the development of the ROS AMCL software package this approach is widely used by many researchers in the community.

While laser range finders (FOV 180–270 degrees, range 30–80m), are very different in quality to RGB-D sensors which typically are highly non-linear beyond 5m and provide no returns at all beyond 9m but do provide 3D estimation within its 60 degree field of view. Nonetheless we wish to use the RGB-D data to estimate the sensor pose in 3D at each time-frame k

$$\mathcal{A}_k = (x_k, y_k, z_k, \phi_k, \theta_k, \psi_k) \quad (1)$$

as well as the associated velocities in each filter dimension. However the addition of a state dimension to a particle filter typically requires an exponential increase in the number of particles.

c) Height, Pitch and Roll Estimation: While the proposed likelihood function can estimate in the full 6-DOF, it is prudent to reduce the dimensionality where possible. For this reason we will assume that the three constrained degrees of freedom — namely pitch, roll and height — can

be accurately estimated independent of the particle filter. This reduces the state dimension to 6 using one of the following:

- Use of an IMU
- Assumption of horizontal motion (in the case of a ground robot)
- Direct estimation of the floor plane from the RGB-D depth data.

In the experiments in Section IV we typically used the later method. The state vector will become

$$\mathcal{A}_k = (x_k, y_k, \phi_k, \dot{x}_k, \dot{y}_k, \dot{\phi}_k) \quad (2)$$

A. Particle Propagation

Our goal is to estimate the posterior distribution of the sensor state recursively using the standard two step Bayesian update rule. We will use sequential Monte Carlo methods to approximate the recursion of the non-linear and non-Gaussian system. In this way we can represent complex probability distributions by a set of weighted Monte Carlo importance samples. We will assume that the initial state distribution, $p(\mathcal{A}_0)$, is known or can be estimated as suggested in Section V.

For each subsequent frame we will propagate the previous state estimate according to the state transition distribution, $p(\mathcal{A}_k|\mathcal{A}_{k-1})$ using the estimate produced by the FOVIS visual odometry algorithm [7]. For each dimension the propagation equations are of the form (in this case for the $\mathcal{X}$ -dimension)

$$x_k = x_{k-1} + \Delta T \dot{x}_k + \mathcal{N}(0, \sigma_x^2) \quad (3)$$

$$\dot{x}_k = \dot{x}_{k,\text{vo}} + \mathcal{N}(0, \sigma_{\dot{x}}^2) \quad (4)$$

where the final term in each equation adds a small amount of normally distributed noise so as to support unexpected target motion using $\sigma_x^2 = 0.004$ and $\sigma_{\dot{x}}^2 = 0.0004$. The term $\dot{x}_{k,\text{vo}}$, is the relative (2-D) visual odometry translation estimate, $[v_{k,\text{vo}}, v_{k,\text{vo}}, \phi_{k,\text{vo}}]$, transformed into the particle's global coordinate frame

$$\dot{x}_{k,\text{vo}} = \frac{v_{k,\text{vo}} \cos(\phi_{k-1}) - w_{k,\text{vo}} \sin(\phi_{k-1})}{\Delta T} \quad (5)$$

$$\dot{y}_{k,\text{vo}} = \frac{v_{k,\text{vo}} \sin(\phi_{k-1}) + w_{k,\text{vo}} \cos(\phi_{k-1})}{\Delta T} \quad (6)$$

$$\dot{\phi}_{k,\text{vo}} = \frac{\phi_{k,\text{vo}}}{\Delta T} \quad (7)$$

Typically the period of time between frames is $\Delta T = 0.1$ seconds.

For smooth and continuous motion, the FOVIS algorithm demonstrates relative odometry estimates with a mean velocity error of 0.08m/s in typical indoor environments. However, during abrupt accelerations and sharp turning motions the feature-based visual odometry algorithm will suffer from periods of total failure. These failures are typically due to motion blur and other problems with the rolling shutter of the Kinect camera. (Given space restrictions, the interested reader is directed to [7] for extensive testing of the visual odometry.)

Fortunately, these failures are indicated by low levels of feature matching, when this is detected we will instead propagate the particle set using a noise-driven dynamical model replacing Eq 4 with

$$\dot{x}_k = \dot{x}_{k-1} + \mathcal{N}(0, \sigma_{\dot{x}}^2) \quad (8)$$

and $\sigma_{\dot{x}}^2 = 0.001$. If the failure is relatively short in duration (less than 3 seconds), it is possible for the MCL algorithm to overcome this failure entirely³. For longer duration failures, we envisage abandoning the current particle set and reinitializing the system anew using visual bag of words. This is discussed in Section V.

B. Likelihood Function

Having proposed the particles for the current instance, we now wish to evaluate a likelihood for each particle using the current sensor depth data and to use it to update the particle weights from the previous iteration. Typically, the majority of computing time is spent evaluating the particle filter likelihood function and we have given careful thought to its design.

Firstly we propose to down-sample the incoming data. Our experimentation has shown that from the 640x480 pixel image/cloud, a sufficiently informative likelihood function is possible using only 160–200 of the pixels. Using these points we wish to determine which of the particle poses is most justified.

d) *Euclidean-based Likelihood Function*: Our current likelihood function probabilistically measures the fit between the RGB-D point cloud (translated by the proposed particle pose) and the planar submap using Euclidean distance. This is carried out in a manner somewhat like the *cost function* of the Iterative Closest Point (ICP) algorithm - although we carry out no iterative procedure.

For a given particle $\mathcal{A}_k^{(p)}$, the RGB-D cloud is first transformed onto the particle pose and the minimum point-to-plane distance is then found by comparing each point, $z_i^{(p)}$, from the cloud to each plane, s_j , in the submap mentioned above

$$d_{i,\min}^{(p)} = \arg \min_j \|z_i^{(p)} - s_j\| \quad (9)$$

where $\|*\|$ represents the distance from point to plane. Given this distance, the likelihood of this point is evaluated as follows

$$p(\mathbf{z}_k^i | \mathcal{A}_k^{(p)}) = \beta c_r \mathcal{N}(d_{i,\min}^{(p)}; 0, \sigma_r^2) + (1 - \beta) r_{\max}^{-1} \mathcal{U}(0, r_{\max}) \quad (10)$$

where the maximum range is $r_{\max} = 3m$ and all ranges beyond this are set to $r_{\max}$. An appropriate normalization constant, c_r , had been added for the truncated normal distribution and the measurement variance σ_r^2 is estimated to be 0.5 meters. The uniform distribution supports heavy tailed behavior and in doing so each point in the cloud has only a

³By comparison, momentary failure of visual odometry as part of a SLAM system — for even a small number of frames — can result in problems for Vision-only SLAM

small effect on the overall likelihood function. The parameter $\beta = 0.01$ was found to give good experimental performance.

Finally, the overall likelihood of this particle is the product of the point likelihoods across the entire cloud

$$p(\mathbf{Z}_k | \mathcal{A}_k^{(p)}) = \prod_{i=1}^{N_i} p(\mathbf{z}_k^i | \mathcal{A}_k^{(p)}) \quad (11)$$

where N_i is the number of RGB-D points. An example of the evaluation of this likelihood function is demonstrated in Figure 4. This procedure is repeated for each of the particles in the cloud and the weights are then updated to produce an estimate of the posterior distribution at the current time

$$\tilde{w}_k^{(p)} \propto \tilde{w}_{k-1}^{(p)} p(\mathbf{Z}_k | \mathcal{A}_k^{(p)}) \quad (12)$$

Residual resampling is carried out whenever the effective sample size of the particle set falls below 0.5.

Note that ongoing work is considering replacing this likelihood function by a ray-based projective likelihood function as well as considering the contribution of color information.

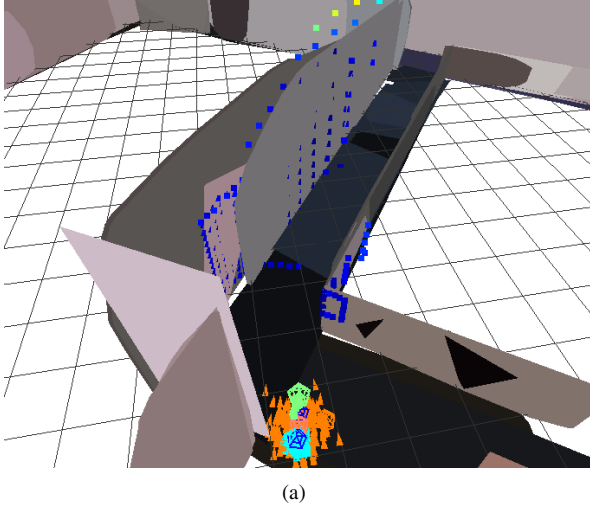


Fig. 4. Example illustration showing the particle pose (green pentagon) which the best fits the 3D map given the current RGB-D data — that is the particle with the largest likelihood. The blue/green/yellow dots indicate the Euclidean distance from each RGB-D data point to the nearest plane (blue is a low distance, yellow is high). In addition, the orange triangles indicate the pose of the entire particle set. This view point corresponds to the Kinect sensor located along the left hand side of Fig. 3

IV. LOCALIZATION EXAMPLES

In this section we will present a number of examples demonstrating accurate localization in the 3D map environment illustrated in Figure 2. In each case data was collected and post-processed to generate the numerical results in Table I which also indicates the path taken in each case.

Each platform had either a Kinect or a Primesense RGB-D sensor. For the man-portable system and the push cart an accurate estimate of motion was estimated using a Hokuyo UTM-30LX mounted in the primary plane of motion and was used to estimate the localization error of the MCL

system. Note that the PR2 and iRobot Create ROS toolchain is currently in development.

No major failure of the localization algorithm occurred in these experiments (i.e. the entire particle set diverging) although troublesome locations containing little or no visual or geometric features do exist within the building. As indicated in the table, 2D RMS error of the order of 50cm was seen for these three platforms, which is encouraging at this early stage. This value is inflated due to poor performance in the aforementioned locations, which perhaps is to be expected. The data for the map-portable exhibits significant motion blur and occasional visual odometry failure; thus the results for this system indicate the robust nature of our approach.

Stable real-time operation with 150 particles has been realized on a 4-core 2.53GHz Pentium Core2 powered laptop — utilizing one core each for data capture, visual odometry and Monte Carlo localization and typically processing at 10Hz. In regions of reduced uncertainty as few as 10 particles are required with operation at several times real-time. For that reason, implementation of an adaptively resizing particle set could be useful in such circumstances [12].

Finally we would like to reemphasize that only the RGB-D Kinect sensor was used in these experiments so as to demonstrate the robustness of Monte Carlo localization with such a low cost sensor. Localization of a nodding or rotating LIDAR within this planar map is straightforward and is likely to be more accurate — given the improved accuracy and range of such a sensor. Also, additional sources of odometry such as wheel odometry or an IMU could have been used to improve the prediction model and to address some of failure modes of the visual odometry mentioned above. We have avoided doing so for simplicity and generality.

Platform	Path	Duration	Distance	RMS Error
Man Portable	ECDBCE	101	88.07	0.885
Quad Rotor	ECDC	48	35	n/a
Push Cart	ACDBC	153	95.33	0.4306
Willow Garage PR2		Toolchain in development		
iRobot Create		Toolchain in development		

TABLE I

Error of RGB-D MCL (using 150 particles) compared to LIDAR-based SLAM estimate. Units: duration seconds; distance and 2D RMS error meters. Path letters correspond to those in Figure 2

V. CONCLUSIONS AND FUTURE WORK

This paper presented an algorithm for the extraction of a geometrically-accurate building model built using planar segments extracted from RGB-D data from the low cost Microsoft Kinect. The model can be efficiently stored and is amenable to improvement and updating.

The model was then used to localize a series of robotic and mapping platforms moving within the mapped environment using a particle filter. Our results illustrate that this Kinect-based localization algorithm is accurate, is robust to the failure of its sub-systems, as well as operating in realtime. The application area of this approach is wide: including not



(a)

Fig. 5. Platforms used in testing, from top-right clockwise: man-portable mapping unit, Willow Garage PR2, iRobot Create, quad-rotor and a push cart.

just traditional robotic localization but many wearable and virtual reality applications.

The work presented herein is under active development and there are many avenues for improvement of the algorithm. As mentioned above we make no use of color information. A projective error module as the ability to improve our performance in sparse environments while a GPU-based implementation has the capacity to vastly increase the speed of operation.

In addition, we would like to explore the possibility of so-called *kidnapped robot* localization using an appearance-based bag-of-words (BOW) such as [13], [14]. We envisage that when a location is proposed by the BOW algorithm, it can be used to propose a portion of the particle set which can then be used to resolve any localization ambiguity.

Finally, open loop operation of the algorithm (for point-to-point navigation) currently being investigated on a number of the platforms mentioned above.

VI. ACKNOWLEDGMENTS

This project makes significant use of the Point Cloud Library [15] (particularly in the map building modules in Section II) and the Sequential Monte Carlo Template Class [16]. Thanks to Abraham Bachrach, Albert Huang and Daniel Maturana and Nick Roy for collection of the quadrotor dataset and for use of the FOVIS library.

REFERENCES

- [1] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 1999.
- [2] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, May 2001.
- [3] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Autonomous Robot Vehicles*, pp. 167–193, Springer Verlag, 1990.
- [4] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "FastSLAM: An efficient solution to the simultaneous localization and mapping problem with unknown data association," *Journal of Machine Learning Research*, vol. 4, no. 3, pp. 380–407, 2004.
- [5] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Shanghai, China), May 2011.
- [6] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Anchorage, Alaska), May 2010.
- [7] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an RGB-D camera," in *Proc. of the Intl. Symp. of Robotics Research (ISRR)*, (Flagstaff, USA), August 2011.
- [8] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J. J. Leonard, "6-DOF multi-session visual SLAM using anchor nodes," in *European Conference on Mobile Robotics*, (Örö, Sweden), 2011.
- [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.
- [10] N. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," vol. 140, pp. 107–113, 1993.
- [11] A. Doucet, N. de Freitas, and N. Gordon, eds., *Sequential Monte Carlo methods in practice*. New York: Springer-Verlag, 2000.
- [12] D. Fox, "Adapting the sample size in particle filters through KLD-sampling," *Intl. J. of Robotics Research*, vol. 22, pp. 985–1003, Dec. 2003.
- [13] C. Cadena, D. Gálvez, F. Ramos, J. Tardós, and J. Neira, "Robust place recognition with stereo cameras," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, (Taipei, Taiwan), October 2010.
- [14] M. Cummins and P. Newman, "Highly scalable appearance-only SLAM - FAB-MAP 2.0," in *Robotics: Science and Systems (RSS)*, (Seattle, USA), Jun 2009.
- [15] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *icra*, (Shanghai, China), May 2011.
- [16] A. M. Johansen, "SMCTC: Sequential Monte Carlo in C++," *Journal of Statistical Software*, vol. 30, pp. 1–41, 4 2009.