Efficient Scene Simulation for Robust Monte Carlo Localization using an RGB-D Camera

Maurice F. Fallon, Hordur Johannsson and John J. Leonard

***Abstract*— This paper presents Kinect Monte Carlo Localization (KMCL), a new method for localization in three dimensional indoor environments using RGB-D cameras, such as the Microsoft Kinect. The approach makes use of a low fidelity *a priori* 3-D model of the area of operation composed of large planar segments, such as walls and ceilings, which are assumed to remain static. Using this map as input, the KMCL algorithm employs feature-based visual odometry as the particle propagation mechanism and utilizes the 3-D map and the underlying sensor image formation model to efficiently generate simulated RGB-D camera views at the location of particle poses, using a graphical processing unit (GPU). The generated 3D views of the scene are then used to evaluate the likelihood of the particle poses. This GPU implementation provides a factor of ten speedup over a pure distance-based method, yet provides comparable accuracy. Experimental results are presented for five different configurations, including: (1) a robotic wheelchair, (2) a sensor mounted on a person, (3) an Ascending Technologies quadrotor, (4) a Willow Garage PR2, and (5) an RWI B21. The results demonstrate that the system can perform robust localization with 3D information for motions as fast as 1.5 meters per second. The approach is designed to be applicable not just for robotics but other applications such as wearable computing.**

I. Introduction

Localization in a previously mapped environment is a key skill to enable lifelong robotic operation. Monte Carlo Localization (MCL) [1], [2] is a well-established technique for mobile robot localization using 2-D laser range scanners. While MCL has been hugely successful, previous work has relied on sensors that cost several thousand dollars, and has largely been limited to motions in the plane in which the prior map was generated. The extension of MCL to three dimensional environments has not been straightforward due to several challenges, including: (1) creation of a suitable 3-D environment representation for the prior map; (2) operation in situations where wheeled odometry measurements are unavailable; and (3) achieving computational efficiency to enable operation with a sufficient number of particles to maintain robust performance.

In this paper, we present Kinect Monte Carlo Localization (KMCL), a new approach to robust and accurate localization in three dimensional environments using inexpensive RGB-D cameras such as the Microsoft Kinect[1] The approach makes use of an *a priori* 3-D model of the area of operation which is composed of large planar segments, such as walls and ceilings, which are assumed to remain static in typical indoor environments. Visual odometry is used for particle propagation, and a likelihood function is computed for each particle by comparing the current sensed data with a prediction of the scene that is efficiently computed using a GPU.

This paper is structured as follows: 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, Section IV presents a series of experimental demonstrations using several platforms — robotic and man-portable – that demonstrate the performance of the approach.

II. Creating a 3-D Building Model

A. Pose Estimation

In this section we outline a procedure for the extraction of a 3-D building model using a robot trajectory estimated using Simultaneous Localization and Mapping. SLAM is a fundamental subfield of robotics and has progressed from the initial work of Smith, Self and Cheeseman [3] to the current

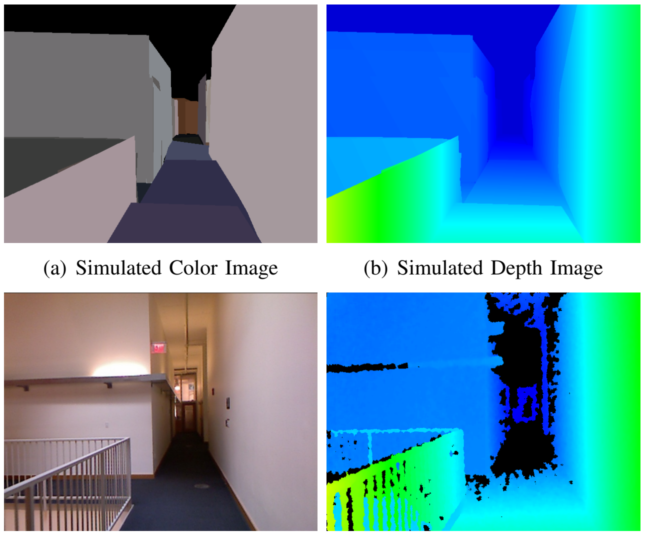

(a) Simulated Color Image (b) Simulated Depth Image
(c) Measured Color Image (d) Measured Depth Image

Fig. 1. *Using a prior 3-D building model we efficiently generate simulated depth and color views (top) which are then compared to RGB-D camera data (bottom). Using a particle filter, hundreds of similar views are evaluated and used to estimate the sensor pose in 3-D.*

This work was partially supported by ONR grants N00014-05-10244, N00014-07-11102, N00014-06-1-0043 and N00014-10-1-0936. The authors are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA, USA `mfallon,hordurj,jleonard@mit.edu`

[1]In this work we will typically refer to the Kinect, but this work is relevant to other sensors such as the SwissRanger and stereo cameras.

Algorithm 1: Kinect Plane Extraction

```
Given a RGB-D point cloud;
First discard all points beyond 5m range (and hence in
the inaccurate non-linear range of the sensor);
Downsample cloud using voxel tree with 30cm leaves;
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 and pose;
        The plane color is determined to be the median
        color of the original points;
```

state of the art of non-linear least squares optimization of the entire robot trajectory and observation set. Implementations such as iSAM [4] and HogMan [5] 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.

At this initial stage of our project we have utilized this type of LIDAR-estimated trajectory to infer the motion of the RGB-D sensor rigidly connected to the Hokuyo UTM-30LX. Doing so allows us to simplify the 3-D map building procedure and to focus on RGB-D-only localization. Note that ongoing visual SLAM research, such as [6], [7], would allow us to drop the requirement for a LIDAR sensor in the map building module in the future.

B. Map Extraction

Using this accurate estimate of the Kinect sensor trajectory, we wish to extract the dominant planar 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 3-D model of an area. Our thesis is that this type of model is sufficient for accurate and robust robot localization in three dimensions.

In what follows we present the algorithm we used to generate the coarse building model in Figure 2 which was sufficient to localize the RGB-D camera. We refer the reader to more advanced research on the perception, segmentation and understanding of 3-D point clouds such as [8], [9] which result in more complete and error-free models of the environment.

We present in Algorithm 1 a series of steps which extract these planes from successive scans of RGB-D data taken by the sensor *continuously* 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

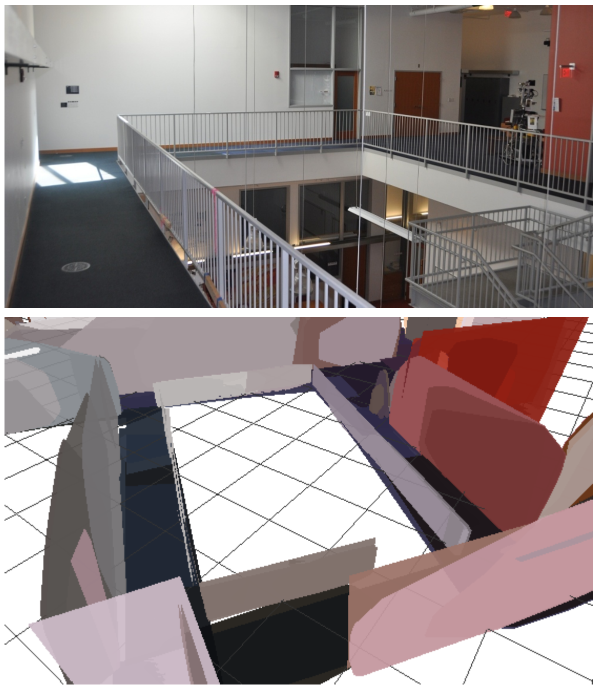


Fig. 2. *The first step of this algorithm is to generate a 3-D building model which indicates the broad geometric information (but not the precise textural information). Plane colors are for illustration purposes only at this stage. This model is later used to allow an RGB-D sensor to localize.*

for each successive scan. We then combine each observation of a particular plane into a final representative polygon. The set of all such planes then represents an accurate map for the area, as seen in Figure 3.

C. 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 3-D map segmentation and storage systems have been proposed and implemented — often building on the octree voxel data structure. For example OctoMap [10] 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 40,000m^2 required 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. Subsequently, should a loop closure be detected by an exploring robot, it is difficult to adjust the structure of the tree. Nonetheless, the octree approach has significant application for path planning and free space detection in cluttered environments.

In comparison the planar polygon map is 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 then easily be adjusted — updating and improving the 3-

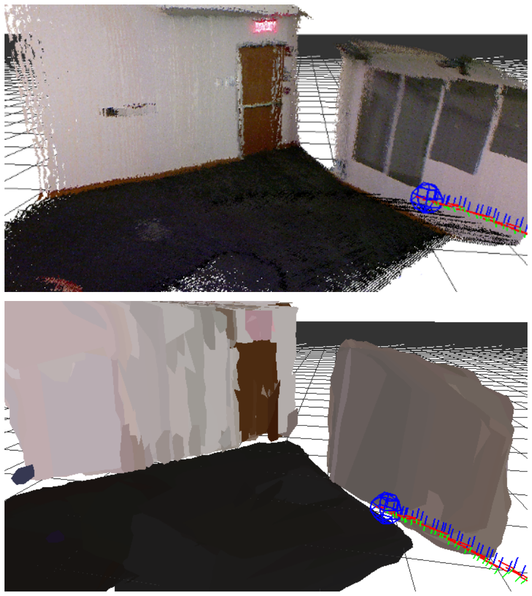


Fig. 3. *Plane extraction procedure: A frame of raw Kinect data is first projected into a 3-D point cloud (top). Then large planar objects are successively extracted (bottom, for several consecutive frames).*

D model. Our representation is also efficient — we project that it will require only 2 MB of disk space to represent the entire 70,000m^2 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.

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. [11]. Initial adaptations of SMC to robot localization were later reported [1], [2] using laser range finders and optionally wheel odometry to localize in 2-D. Using 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 increasingly noisy beyond 5m but do provide 3-D estimation within its 60 degree field of view. Nonetheless we wish to use the RGB-D data to estimate the sensor pose in 3-D at each time-frame k

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

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.

a) 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 constrained degrees of freedom — namely pitch, roll and height — can be accurately estimated independent of the particle filter. This dimensions can be removed either using an assumption of horizontal motion (in the case of a ground robot) or 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 6 element state vector will become

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

In future work we aim to use an IMU and a 8 element dimensional state vector to allow for estimation of the height of an quadrotor or a person ascending/descending a staircase so as to fully support motion in 3-D — in addition to sensing in 3-D.

A. Particle Propagation

The 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 complex probability distributions can be represented 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 [6]. For each dimension the propagation equations are of the form (in this case for the $\mathcal{X}$-dimension)

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

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

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

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

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

$$\dot{\phi}_{k,\text{vo}} = \frac{\phi_{k,\text{vo}}}{\triangle T} \tag{7}$$

Typically the VO frequency used was $\triangle T = 0.1$ 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. The interested reader is directed to [6] for extensive testing of the visual odometry.

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 problems with the rolling shutter of the Kinect camera.

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

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

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 entirely[2]. For longer duration failures, we envisage abandoning the current particle set and reinitializing the system anew using visual bag of words. This will be undertaken as future work (see Section V).

B. *Likelihood Function*

Having proposed the particle set 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 consideration to its design.

Firstly we propose to down-sample the incoming RGB-D image by a factor of 16–32. Our experimentation has shown that from the 640x480 pixel image/cloud, a sufficiently informative likelihood function is possible using only a 20x15 image. Using this smaller image we wish to determine which of the particle poses is most justified.

In this section we describe two methods to do this. In the following section we describe an approach based on how well artificially generated model views match the measured data, while in Section III-B.2 we describe a method similar to the ICP scoring function. A short comparison of the two methods is given in Section III-B.3.

1) Generated Range-Image Ray-to-Plane Function: We propose to compute the likelihood of a particle pose by directly generating the range image that would have been detected from that location using the prior 3-D model. This range image is generated by finding the closest intersection between a ray through the optical center and a point on the image plane with the map for each point in the simulated range image. However instead of brute-force computing the intersection of each ray with each plane, it is possible to render each plane directly to the image plane instead. The distance to the closest plane can then be queried from the associated Z-buffer. This rendering approach is supported in all modern Graphical Processing Units (GPU) and the image for each particle can be rendered very efficiently in this manner.

Our implementation uses the OpenGL library to render the simulated image for each particle. This is similar to the approach in [12] which used gradients in the color image while here we use the depth information. When OpenGL renders an image, such as our 3-D model, it natively uses the Z-buffer to determine hidden surface removal. After rendering, the Z-buffer values can be read and we will use them to compute the likelihood of the current sensor measurement conditioned on the pose of the particle.

[2]By comparison, momentary failure of visual odometry as part of a Vision SLAM system can be result in significant error

Before comparing the sensor depth with the Z-buffer values there are a few technical issues in the rendering pipeline that need be taken into account. First all plane polygon vertices are transformed using the model transform into the camera coordinate frame. (Note that OpenGL defines the camera look axis along the negative Z-axis.)

After the model transformation, the projection transform is applied. This projection creates the clip coordinates, which are in homogeneous coordinates, with each rendering primitive clipped to the box $(-w, -w, -w), (w, w, w)$. The z values are projected in such a way that everything outside the range $[-z_n, -z_f]$ is clipped, where z_n and z_f are the near and far range of the z-axis [13]. For the Kinect RGB-D sensor z_n is 0.7m and z_f is 20m.

Finally the projected inverse depth on the range $[1/z_n, 1/z_f]$ is mapped $[0, 1]$ before it is written to the Z-buffer which we can then access.

For a camera calibration matrix K

$$K = \begin{bmatrix} f_x & 0 & -c_x & 0 \\ 0 & f_y & -c_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

the projection matrix can be written as KP

$$P = \begin{bmatrix} 2/W & 0 & 1 & 0 \\ 0 & 2/H & 1 & 0 \\ 0 & -\frac{z_f + z_n}{z_f - z_n} & -\frac{2 z_f z_n}{z_f - z_n} & \\ 0 & 0 & -1 & 0 \end{bmatrix} \tag{10}$$

Where W and H are the image width and height respectively. An example depth image is illustrated in Figure 1. Next we wish to compare it to the measured image (also illustrated) to produce a likelihood for the particle pose.

a) Likelihood Formulation: To compute the particle likelihood either the inverse depth Z-buffer values can be converted to depth or the RGB-D depth data can be converted to the inverse depth. As the accuracy of camera depth measurements is a linear function of inverse depth we propose to formulate the likelihood function in inverse depth space. This is similar to the approach taken in monocular camera SLAM, for example [14].

For a given particle $\mathcal{A}_k^{(p)}$ at time step k, the inverse depth image, $Z_{(p)}^G = (z_{(p)}^0, \ldots, z_{(p)}^{N_i})$, containing N_i points is generated as described above. For each inverse depth pixel, z_k^i, in the measured image $Z_k^M = (z_k^0, \ldots, z_k^{N_i})$, the likelihood is evaluated as follows

$$p(\mathbf{z}_k^i | \mathcal{A}_k^{(p)}) = \beta c_r \mathcal{N}(z_k^i; z_{(p)}^i, \sigma_d^2) + (1 - \beta)\mathcal{U}(0, 1) \tag{11}$$

where the inverse depth varies in the range $(0, 1)$. An appropriate normalization constant, c_r, had been added for the truncated normal distribution and the inverse depth variance σ_d^2 was chosen to be $0.1m^{-1}$. The addition of the uniform distribution supports heavy tailed behavior and in doing so each point in the cloud has only a small effect on the overall

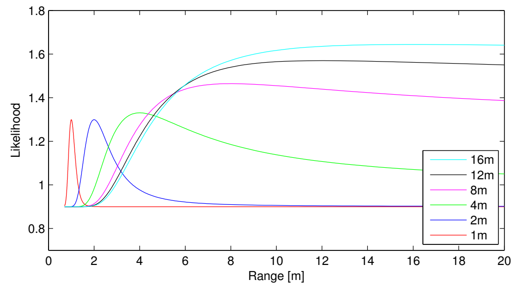


Fig. 4. *Inverse Depth-parameterized likelihood function evaluated for a variety of nominal model depths. This parameterization evaluates high-range depth points with higher variance than shorter ranges — accurately representing the underlying disparity measurement uncertainty.*

likelihood function. The parameter $\beta = 0.01$ was found to give good experimental performance.

The overall likelihood of the particle is then the product of the point likelihoods across the entire cloud

$$p(\mathbf{Z}_k|\mathcal{A}_k^{(p)}) = \prod_{i=1}^{N_i} p(\mathbf{z}_k^i|\mathcal{A}_k^{(p)}) \tag{12}$$

The 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

$$\tilde{w}_k^{(p)} \propto \tilde{w}_{k-1}^{(p)} p(\mathbf{Z}_k|\mathcal{A}_k^{(p)}) \tag{13}$$

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

Figure 4 illustrates the inverse depth parameterization of the likelihood function described in Equation 11. It has been projected onto the depth axis, evaluated for a series of nominal model depths. For points at shorter range the function is much more discriminative than for points at large ranges — directly matching the measurement error distribution. This allows us to take advantage of Kinect depth measurements all the away up to 20 meters away.

While this approach is a principled way of utilizing the noisy long-range Kinect RGB-D data, discretization of the measurement depth values (ranging $(0, 2047)$) as well as texture dependent biases are evident at long ranges (>15m).

In addition to depth information, this method can be straightforwardly supplemented with basic color information by reading the color buffer to produce colored model views similar to that illustrated in Figure 1. This could be useful in long shapeless corridors for example.

2) Pure Euclidean Point-to-Plane Function: For comparison to the method proposed above, we also developed a more traditional likelihood function initially. It operates in a manner somewhat similar to the *cost function* of the Iterative Closest Point (ICP) by evaluating the Euclidean distance between the RGB-D point cloud (transformed by the proposed particle pose) and the planar submap.

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 found by comparing each point, $z_{(p)}^i$, from the cloud to each plane, s_j, in the submap mentioned above

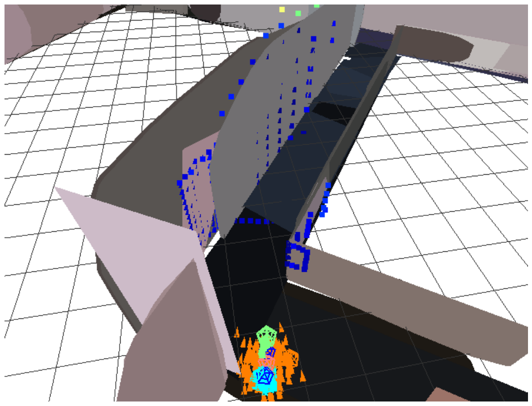


Fig. 5. *Illustration the particle pose (green marker) which the best fits the 3-D map given the current RGB-D data — that is the particle with the largest likelihood. The blue/green/yellows dots indicate the distance from each RGB-D data point to the nearest plane (blue is a low distance). In addition, the orange triangles indicate the poses of the entire particle set.*

$$d_{i,\min}^{(p)} = \arg\min_j \|z_{(p)}^i - s_j\| \tag{14}$$

where $\| * \|$ represents the distance from point to plane.

Given this distance, the individual point likelihood is then evaluated using a form similar to Equation 11 with the per particle likelihood again being the product of the individual point likelihoods.

3) Comparison Between Functions: For many sensor views these two methods have demonstrated accurate localization as a good match of all points to the accurate 3-D building model inevitably results in a low distance score (See Section IV-B). While the latter method has been demonstrated to be significantly more computationally intensive, additionally it demonstrates a failure mode that our proposed method does not suffer from.

ICP search is well known to suffer from poor convergence when incorrectly initialized and the Euclidean-based particle filter has been observed to demonstrate a similar behavior. An example situation is that presented in Figure 1. The method is prone to incorrectly associating depth points from the wall to the model's railing — resulting in an incorrect local minimum and a possible failure of the particle filter.

Instead the likelihood surface for the proposed generative method would not suffer from this issue as such a pose could not have sensed that measurement. This correct likelihood surface is illustrated in Figure 6.

IV. Localization Examples

In this section we will present a number of examples demonstrating accurate localization in the 3-D map environment. Five platforms were used: a robotic wheelchair, a Willow Garage PR2 (supplemented with a Kinect), an Ascending Technologies quadrotor and a man-portable mapping device as shown in Figure 7 as well as an RWI B21.

Each platform had a Microsoft Kinect facing forward. For all the platforms except the quadrotor an accurate estimate of

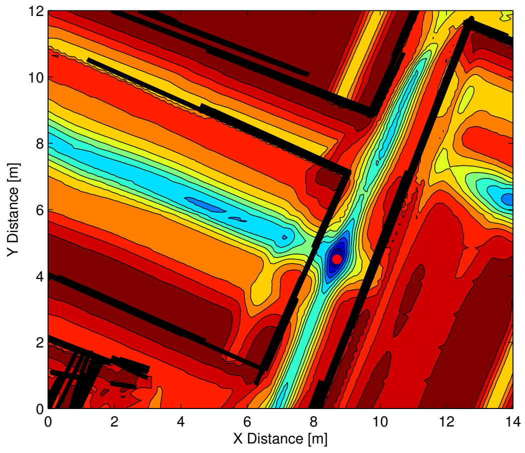


Fig. 6. *The generative likelihood function evaluated as a contour surface with 10 cm spacing (in 2D) around the location illustrated in Figure 1 (denoted by the red circle). The walls and railings are in black. The multi-modal surface has a broad peak at the correct location ensuring stable MCL.*

the ground truth pose was estimated using posegraph SLAM and a Hokuyo UTM-30LX mounted in the primary plane of motion and was used to estimate the localization error of the KMCL system. The height of the sensor varied between 1–2 meters, which demonstrates the flexibility of this approach.

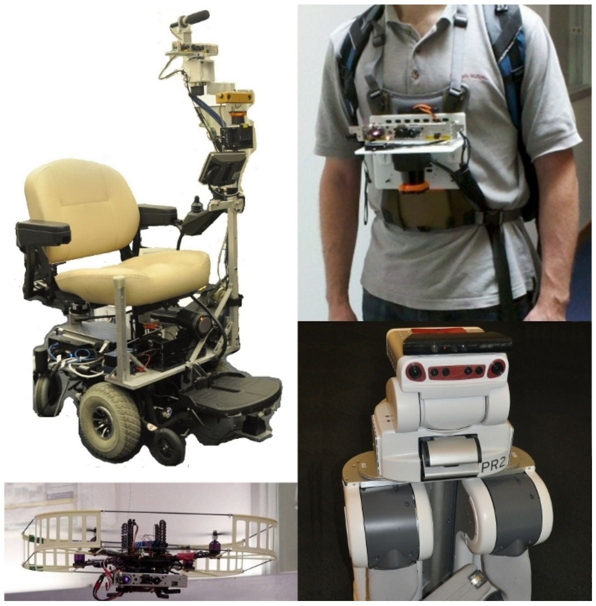


Fig. 7. *Platforms used in testing, from top-right clockwise: man-portable mapping unit, Willow Garage PR2, quad-rotor and a robotic wheelchair. In addition an RWI B21 was used but is not shown.*

A. Illustrative Results

For each platform a dataset was collected and post-processed using the KMCL algorithm to generate the trajectories illustrated in Figure 8 which also indicates the ground truth estimated using the LIDAR on each platform. Table I presents some numerical results for 350 particles.

Platform	Duration	Distance	Speed	Median Error	3σ %
Units	seconds	m	m/s	m	%
Man-carried	94	99	1.05	0.66	52
Wheelchair	180	215	1.20	0.48	75
Quadrotor	45	$\sim$30	$\sim$0.66	n/a	n/a
PR2	266	126	0.47	0.30	0.90
B21	349	152	0.43	0.33	0.84

TABLE I

Performance of the KMCL algorithm (using 350 particles) for the trajectories in Figure 8.

No major failures 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 - such as long corridors or blank walls. In these locations the particle set naturally disperses somewhat until some conclusive geometric information is observed, at which point the particle distribution coalesces to a single mode and continues accurate tracking. These situations, despite being representative of the underlying position uncertainty, result in an increased error result in Table I and Section IV-B. For small particle sets these situations can also result in particle filter divergence (as discussed in Section IV-B).

The error metrics we chose were median absolute error and the percentage of SLAM poses lying within a 3σ interval of the weighted particle mean. As indicated in the table, typical median error is of the order of 40 cm. This value is inflated due to poor performance in the aforementioned locations.

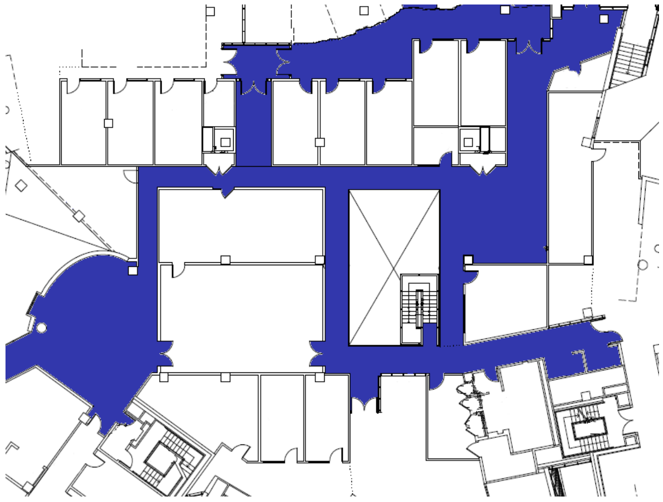
In particular the data for the map-portable exhibits significant motion blur and frequent visual odometry failure; thus the results with this system indicate the robust nature of our approach. Additionally the quality of the 3-D map and the choice of image downsample factor are other parameters that could be studied so as to improve performance further.

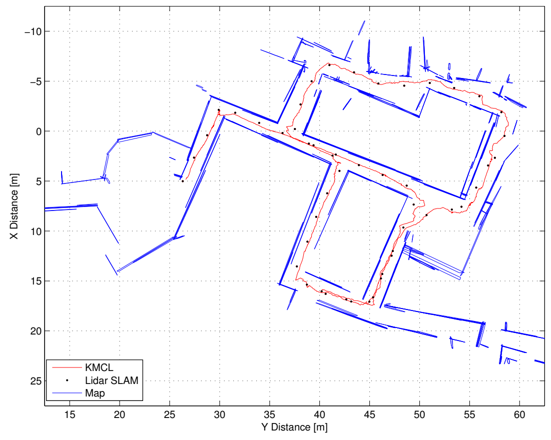
B. Performance Evaluation

To fully quantify the performance of our algorithm we carried out a series of Monte Carlo runs using the dataset collected with the robotic wheelchair. Varying the number of particles from 12 to 400, the algorithm was run on a 3 minute, 215 meter dataset for 20 independent runs. This represents a total of 15 hours of computation time.

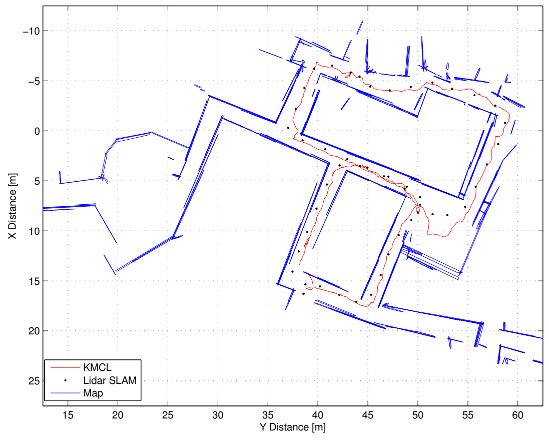
The results of each run were compared to the LIDAR pose of the robot aligned with the particle filter pose trajectory. In Figure 9 we present our results. The results give an indication of the accuracy of the proposed algorithm as well as its stability for different particle numbers.

In summary we observed roughly equivalent localization accuracy with equal numbers of particles — with the Euclidean likelihood function being slightly more accurate. The major difference between the two methods was in the computation requirements. For 100 particles and a framerate of 10 Hz, the generative method is real-time while the Euclidean method is 5 times slower than real-time. The slow pace of the Euclidean likelihood precluded us from testing with 200 and 400 particles (where the gap was even wider).

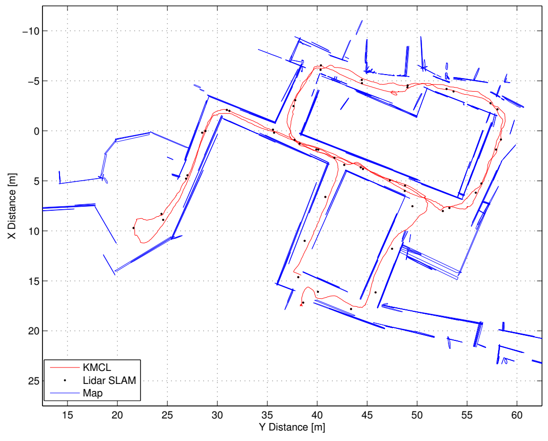

(a) Floor Plan



(b) Willow Garage PR2



(c) Map Portable Device



(d) Robotic Wheelchair

Fig. 8. *Top-down view of mapping performance for three of the robots localizing using the same map (non-simultaneously). The main walls in the map are shown in blue while each robot's Kinect MCL output is shown with a red line. Black dots indicate the posegraph SLAM estimated position during the runs — which indicates accurate localization.*

Stable real-time operation with 350 particles has been realized on a 4-core 2.53GHz Pentium Core2 powered laptop with an Nvidia Quadro 1700M with 32 Cores — utilizing one CPU core each for data capture and visual odometry and Monte Carlo localization split between each architecture and processing at an effective rate of 7–8Hz. In regions of low uncertainty as few as 10 particles are required for operation. Implementation of an adaptively resizing particle cloud would be useful in such circumstances [15]. We envisage some straightforward optimizations (such as submapping) will allow for 1000 hypotheses in realtime for challenging locations.

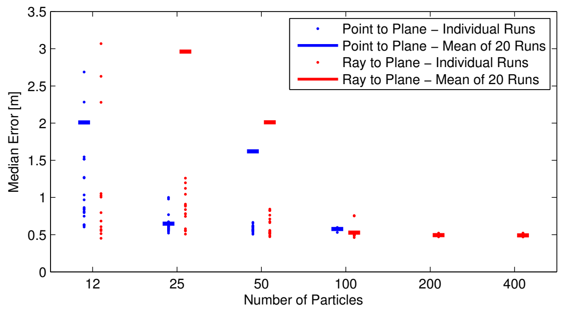
Finally we would like to reemphasize that only the RGB-D Kinect sensor was used in all of these experiments so as to demonstrate the robustness of Monte Carlo localization with such a low cost sensor. Additional sources of odometry such as wheel odometry or an IMU would of course have been used to improve the prediction model and to address locations in which the visual odometry fails, as mentioned above. We have avoided doing so for simplicity and generality.

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 was then used to localize a series of robotic and mapping platforms moving within the mapped environment using a particle filter and *only the data from the RGB-D sensor*.

Our results illustrate that Kinect-based localization is accurate and robust to the failure of its sub-systems, as well as operating in realtime. The application area of this approach is wide: including not just traditional robotic localization but many wearable and virtual reality applications.

One of the primary novelties of this work is the use of a generative measurement model to simulate range images which are then directly compared to the measured RGB-D image. This GPU-based approach is significantly quicker than a more traditional ICP-like method. While the map-portable system and the quadrotor datasets present some variation in roll, pitch and height; we have not as yet



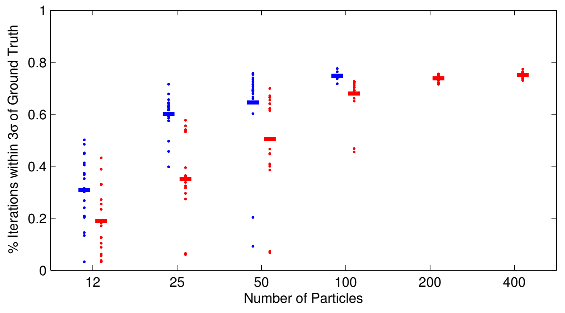


Fig. 9. *Performance metrics for both likelihood functions (averaged for 20 separate runs). Typical performance for 100 particles is of the order of 0.5 m median error and 78% of estimates within 3σ of the true location. Note that for some failed runs with low particle numbers the median error is greater than 3.5m.*

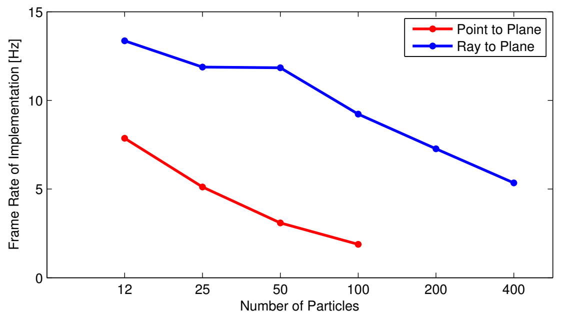


Fig. 10. *Timing statistics for both likelihood functions. In the case of the later, real time operation (at 10Hz) has be achieved with 100 particles and the frequency reduces linearly with the number of particles.*

extended this system to completely free 3-D motion. This extension is the focus of our future research.

Additionally we would like to support *kidnapped robot* localization using appearance-based bag-of-words (BOW) [16]. We envisage that when a location is suggested by the BOW algorithm, it can be used to propose new particle locations and to resolve localization ambiguity. Finally, open loop operation of the algorithm is also being investigated on some of the platforms mentioned above.

VI. Acknowledgments

This project makes significant use of the Point Cloud Library [17] and the Sequential Monte Carlo Template Class [18]. 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 and Matt Walter, Sachithra Hemachandra, Jon Brookshire and Seth Teller for the robotic wheelchair dataset and development of the man-portable mapping device.

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] 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.

[5] 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.

[6] 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.

[7] 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*, (Örbero, Sweden), 2011.

[8] J. Yao, P. Taddei, M. R. Ruggeri, and V. Sequeira, “Complex and photo-realistic scene representation based on range planar segmentation and model fusion,” *Intl. J. of Robotics Research*, pp. 1263–1283, 2001.

[9] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroter, L. Murphy, W. Churchill, D. Cole, and I. Reid, “Navigating, recognising and describing urban spaces with vision and laser,” *The International Journal of Robotics Research*, vol. 28, October 2009.

[10] 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.

[11] 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.

[12] S. Nuske, J. Roberts, D. Prasser, and G. Wyeth, “Experiments in visual localisation around underwater structures,” in *Field and Service Robotics*, Springer Tracts in Advanced Robotics, pp. 295–304, Springer, 2010.

[13] R. Wright, B. Lipchak, and N. Haemel, *OpenGL SuperBible: Comprehensive Tutorial and Reference*. Addison-Wesley Professional, fourth ed., 2007.

[14] L. A. Clemente, A. J. Davison, I. Reid, J. Neira, and J. D. Tards, “Mapping large loops with a single hand-held camera,” in *Robotics: Science and Systems (RSS)*, Jun 2007.

[15] D. Fox, “Adapting the sample size in particle filters through KLD-sampling,” *Intl. J. of Robotics Research*, vol. 22, pp. 985–1003, Dec. 2003.

[16] M. Cummins and P. Newman, “Highly scalable appearance-only SLAM - FAB-MAP 2.0,” in *Robotics: Science and Systems (RSS)*, (Seattle, USA), Jun 2009.

[17] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, (Shanghai, China), May 2011.

[18] A. M. Johansen, “SMCTC: Sequential Monte Carlo in C++,” *Journal of Statistical Software*, vol. 30, pp. 1–41, 4 2009.