Fourier Volume Registration based Dense 3D Mapping

Luke Lincoln¹*, Ruben Gonzalez¹

¹ Institute for Integrated and Intelligent Systems (IIS), Griffith University, Gold Coast, Australia

Abstract

In image processing phase correlation has been shown to outperform feature matching in several contexts. In this paper, a novel volume registration technique is proposed for solving the simultaneous localization and mapping (SLAM) problem. Unlike existing methods which rely on iterative feature matching, the proposed method utilises 3D phase correlation. This method provides high noise robustness, even in the presence of moving objects within the scene which are problematic for SLAM systems. Furthermore, a novel projection method is proposed which performs Fourier based volume registration 3 times faster. Quantitative and qualitative experimental results are presented, evaluating the proposed method’s the noise sensitivity, performance, reconstruction quality and robustness in the context of moving objects.

Received on 16 May 2016; accepted on 27 June 2016; published on 12 September 2016

Keywords: SLAM, volume registration, phase correlation, Fourier transform

Copyright © 2016 Luke Lincoln and Ruben Gonzalez, licensed to EAI. This is an open access article distributed under the terms of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/), which permits unlimited use, distribution and reproduction in any medium so long as the original work is properly cited.
doi:10.4108/eai.12-9-2016.151675

1. Introduction

Simultaneous localization and mapping (SLAM) has many applications in robotics, architecture and engineering, business and science. Its objective is to produce a map (2D birds-eye-view, or 3D) of an environment using image and other sensory data. This is typically performed by computing local features, iteratively matching them across frames and solving for the camera pose and location. This feature matching approach is dependent on finding a sufficient number of matches. When this is true the approach is able to cope with local matching disparities by treating them as outliers. This technique is not robust when features are not stable or when feature confusion occurs.

To alleviate these shortcomings we propose using 3D phase correlation based volume registration to solve the SLAM problem. Given two volumes, volume registration algorithms find a geometrical transformation in which to align the data within the volumes. Fourier based registration is known to be fast, robust to noisy data and scales naturally to parallel processing [8]. It has also been shown to be able to outperform local feature based methods in 2D image processing [10]. This approach is capable of real-time SLAM if used in a parallel programming context. However, this approach is still computationally complex for most practical volume sizes on multiprogramming systems.

In the context of SLAM, scale is often not required in registering volumes. Therefore we propose a novel technique to speed up Fourier based volume registration for estimating translation and rotation parameters. This is achieved by applying a novel projection transform called a spherical-map transform. Along with orthogonal projection, this method allows 2D phase correlation to be used in place of the 3D counterpart. The result is a 3D phase correlation method which is 3 times faster than the original. Applied to SLAM this performance increase allows larger volumes to be processed for the same complexity or correspondingly greater accuracy for a given amount of computation time.

2. Previous Work
2.1. Monocular Camera Feature Based Systems
Monocular Feature based SLAM systems use feature matches to estimate camera pose and location changes across frames [4]. Variations of this method use different features including: corners and lines [14], image patches [25] and exemplar feature matching [3]. SIFT features are used most often in SLAM [1, 7, 13, 22], in addition FAST features have been explored [16–19]. Beall et al [1] made use of both SIFT and SURF features in their underwater SLAM system. Real-time monocular SLAM systems based on this approach have also been proposed [3, 22]. RANSAC is often used in monocular SLAM [7, 16–18, 24] to remove outliers which cause incorrect camera parameter estimates. Bundle adjustment is also used as an additional step to refine camera parameter estimation [7].

2.2. Stereo Camera Feature Based Systems
Stereo based SLAM systems also use features to estimate camera parameters. However, stereo based systems are capable of generating dense depth data more easily using stereo algorithms. Miro et al [20] proposed a stereo based method which uses SIFT and the extended Kalman filter. The method by Van Gool et al [23] works with un-calibrated stereo pairs. It uses Harris corner features and a multi-view stereo algorithm. Sim et al [26] and Gil et al [9] both presented stereo based SLAM systems which use SIFT.

2.3. RGB-D Sensor Feature Based Systems
RGB-D SLAM systems use both depth and image data and are capable of generating dense 3D reconstructions. Many of these methods rely on feature matching techniques [5, 6, 11]. RANSAC is often used to filter outliers for the estimation of camera parameters [5, 6, 11]. Another method which has also been used extensively in the area is Iterative Closest Point (ICP) [2, 6, 11, 12, 21, 27]. ICP iteratively registers point cloud data, and is used to refine camera parameter estimates. A method named KinectFusion was proposed by Newcombe et al [21] which uses RANSAC and a GPU implementation of ICP. Whelan et al [29] extended this method allowing it to map larger areas using Fast Odometry From Vision (FOVIS) over ICP. Bylow et al [2] improved the ICP approach by registering data using a signed distance function.

2.4. Non-Feature Based Methods
Several RGB-D SLAM systems are also non-feature based [12, 15, 28]. Weiksdorfer et al [28] presented a novel sensor system named D-eDVS along with an event based SLAM algorithm. The D-eDVS sensor combines depth and event driven contrast detection. Rather than using features, it uses all detected data for registration. Kerl et al [15] proposed a dense RGB-D SLAM system which uses a probabilistic camera parameter estimation procedure. It uses the entire image rather than features to perform SLAM.

2.5. Summary
As is evident from the current literature, SLAM typically relies on feature matching and RANSAC. However, these approaches fail when there are too few features, when feature confusion occurs or, when features are non-stationary due to object motion. As the extent of random feature displacement becomes more global the effectiveness of these approaches diminishes. Feature matching also dominates in image registration. However, Fourier based methods have been shown to work well under larger rotations and scales [10] whilst being closed form, insensitive to object motion and scaling naturally to GPU implementations. Accordingly, we propose a novel, closed form Fourier based SLAM method.

3. Method
The proposed SLAM method consists of various steps. First each frame $f_i$ that is captured, consisting of a colour and depth image pair is projected into 3D space, forming colour point cloud $points_i$ and re-sampled into a volume $V_i$. Then, the transform parameters between pairs of volumes $V_i$ and $V_{i+1}$ are estimated using $VolumeRegister_{points_i, points_{i+1}}$ shortened to $VR_{points_i, points_{i+1}}$. These parameters are used to update transformation matrix $M$. The points corresponding to $f_i$ ($points_i$) are then transformed using the updated $M$ matrix and added to the cumulative $PointCloud$ database. Two lists, $Cameras$ and $Poses$, are also updated to track camera pose and location per frame. This basic procedure is given in listings 1 and elaborated upon in subsequent subsections.

3.1. Sensor Input
The input to our method is a color and depth image pair, $f(u, v)$ and $g(u, v)$ obtained using an Asus Xtion PRO LIVE sensor at a resolution of $640 \times 480$. Each pixel is projected into 3D space using 

$X_{u, v} = \frac{(u-c_x)Z_{u, v}}{f}$, $Y_{u, v} = \frac{(v-c_y)Z_{u, v}}{f}$ and $Z_{u, v} = g(u, v)$. Here, $[c_x c_y]^T$ represent the center of the image whilst $f$ represents the focal length, defined as 525.0. The point clouds generated by projecting these images are then quantized into image volumes. Results reported in this paper were obtained using volumes of $384^3$ voxels in size.

3.2. Volume Registration
Figure 1 shows a functional block diagram of our method. The input data are two 3D volumes.
Listing 1. Phase Correlation Based SLAM Algorithm

```cpp
f_1 = ReadFrame();
PointCloud = project(f_1);
M = IdentityMatrix();
Camera = [0, 0, 0]^T;
Pose = [0, 0, 1]^T;
Cameras = [Camera], Poses = [Pose];
while (more frames) {
    f_2 = ReadFrame();
    points_1 = project(f_2);
    points_2 = project(f_1);
    V_1 = ResampleVolume(points_1);
    V_2 = ResampleVolume(points_2);
    (\theta, \phi, t_x, t_y, t_z) = VRopt_t_z(t_z(V_1, V_2);
    M = M \times Transform((\theta, \phi, t_x, t_y, t_z);
    points_1 = Transform(points_1, M);
    PointCloud = PointCloud \cup points_1;
    Camera = M^{-1} \times Camera;
    Pose = M^{-1} \times Pose;
    Cameras.add(Camera);
    Poses.add (Camera | Pose);
    f_1 = f_2;
}
```

(Volume_1 and Volume_2) and the output is the transformation matrix required to register the two volumes. The volumes are first Hanning windowed. Next, a translation independent representation is obtained for each by taking the magnitude of their 3D FFTs. Then a log function is applied to the resulting magnitude values, improving scale and rotation estimation [10]. Following a log-spherical transformation, 3D phase correlation is performed to find the global rotation and scale relationship between Volume_1 and Volume_2. Volume_1 is then inversely transformed by the rotation and scale parameters, leaving only the translation to be resolved. This is found by applying phase correlation again between the transformed Volume_1 and Volume_2.

### 3.3. Phase Correlation

Given a volume V_1 and a spatially shifted version of it V_2, the offset can be recovered using PhaseCorrelation (Eq. 1). This function takes two volumes as input and returns the translation between them.

\[
(x, y, z) = \text{PhaseCorrelation}(V_m, V_n) \tag{1}
\]

The PhaseCorrelation function first applies 3D FFTs to volumes, V_1 and V_2, converting them into the frequency domain, i.e. \( F_{1_{x,y,z}} = \text{FFT}(V_1) \) and \( F_{2_{x,y,z}} = \text{FFT}(V_2) \). Taking the normalised cross power spectrum using Eq. 2 completes the Phase correlation function.

\[
F_{3_{x,y,z}} = F_{1_{x,y,z}} \circ F_{2_{x,y,z}}^* \tag{2}
\]

Here, \( \circ \) is an element-wise multiplication and \(|x|\) is the magnitude function. Taking the inverse FFT of \( F_{3} \), gives the phase correlation volume \( V_3 \). The location of the peak value in \( V_3 \), \((x_1, y_1, z_1)\) gives the shift between the \( V_1 \) and \( V_2 \). The phase correlation volume is typically noisy making the peak difficult to locate.

### 3.4. Recovering Scale, Rotation and Translation Parameters

If \( V_1 \) and \( V_2 \) are instead rotated and scaled versions of the same volume, such that they are related by some translation \((t_x, t_y, t_z)\), y-axis rotation \( \theta \), and scale \( \varphi \). Further action is required to recover translation parameters. The first step, given two volumes \( V_1 \) and \( V_2 \) of size \( N^3 \) is to apply a Hanning windowing function (Eq. 3).

\[
H_{W_{x,y,z}} = \frac{1 - \cos(2\pi \sqrt{\left(\frac{x}{N}\right)^2 + \left(\frac{y}{N}\right)^2 + \left(\frac{z}{N}\right)^2})}{2 \sqrt{\left(\frac{x}{N}\right)^2}} \tag{3}
\]

The rotation and scale factors are recovered first using a translation independent representation of the volumes using the Fourier shift theory. For this, the magnitude of the FFT of the volumes is taken, \( M_1 = |\text{FFT}(V_1)| \), \( M_2 = |\text{FFT}(V_2)| \). The zero-frequency of both \( M_1 \) and \( M_2 \) is shifted to the center of the volume and the log of the result is taken \( M_1' = \log(M_1) \), \( M_2' = \log(M_2) \) which reduces noise on the phase correlation volume. A log-spherical transform is then used to turn rotation and scaling into translation for both \( M_1' \) and \( M_2' \). Eq. 4 shows the corresponding log-spherical space coordinate \((X_{\log-spherical}, Y_{\log-spherical}, Z_{\log-spherical})\) for a given \((x, y, z)\) euclidean space coordinate.

\[
\begin{align*}
X_{\log-spherical} &= \frac{\arctan \left( \frac{y - \frac{N}{2}}{\sqrt{x^2 + y^2 + z^2}} \right)^{-1} N}{360} \\
Y_{\log-spherical} &= \frac{\arccos \left( \frac{y}{\sqrt{x^2 + y^2 + z^2}} \right) N}{180} \\
Z_{\log-spherical} &= \log \left( \frac{N}{2N} \right) \end{align*} \tag{4}
\]

The log-spherical transforms of \( M_1' \) and \( M_2' \) are then phase correlated to find the shift between them, \((x_{M_1'}, y_{M_1'}, z_{M_1'}) = \text{PhaseCorrelation}(M_1', M_2')\). The rotation \( \theta \) and scale \( \varphi \) factors between \( V_1 \) and \( V_2 \) can
then be found from the shift parameters using Eq. 5.

\[
\theta = \frac{-360 \times M'}{N} \quad \varphi = e^{-\left(2.56^{-1}N\right)z_{M'N^{-1}}}
\]  
(5)

Using \(\theta\) and \(\varphi\), \(V_1\) can now be inverse transformed (using \((\frac{N}{2}, \frac{N}{2}, \frac{N}{2})\) as the origin). This aligns \(V_1\) and \(V_2\) with respect to scale and \(y\)-axis rotation. The translation parameters \((t_x, t_y, t_z)\) can then be found using phase correlation as given in Eq. 6.

\[
(t_x, t_y, t_z) = \text{PhaseCorrelation}(\text{scale}(\text{rotate}(V_1, \theta), \varphi), V_2)
\]  
(6)

The complete function to recover translation, rotation and scaling, combining equations 2-6 as is denoted in 1 is 7.

\[
(\theta, \varphi, t_x, t_y, t_z) = \text{PhaseCorrelation}_{\theta, \varphi, t_x, t_y, t_z}(V_m, V_n)
\]  
(7)

4. Improving Performance

To reduce complexity we focus on areas which require the most computation time. In the earlier defined Fourier based reconstruction technique, this occurs in the two 3D phase correlations which need to be computed. We describe the method of reduction here; a block diagram for this technique is given in figure 2. We refer to this method as fast volume registration (FVR) in reference general volume registration (VR). The speedup begins by computing the 3D DFT of both input volumes and taking the magnitude of these. Rather than directly performing a 3D log-spherical transform and a 3D phase correlation operation on these volumes, we use a novel transform we call a spherical-map transform (details in 4.1).

This transform converts rotation into translation whilst simultaneously unfolding the 3D space down to 2-dimensions. After this, a 2D phase correlation that requires significantly less processing compared with the 3D counterpart is used to compute the rotation. Next, having obtained the rotation parameter, the rotation is eliminated from the transformation by rotating the first volume by this parameter. The two volumes are then passed through two orthogonal projection mapping functions. This also converts the volumes to 2D space. We use two transforms for both volumes, one projection along the \(x\)-axis, another along the \(z\)-axis. Once the \(x\)-axis projections of both volumes are complete, we can do another 2D phase correlation to give us the \(z\)-translation. The 2D phase correlation of the \(z\)-axis projections gives us the \(x\) and \(y\) axis translations separating the original volumes. The final output of this method gives the rotation and translational shifts between the input volumes. The projections add little complexity to the overall algorithm and since 2D phase correlation operations are used in place of 3D ones, much computation time is reduced.

4.1. Spherical–map transform

The spherical map transform both reduces the 3D volume to a 2D image, and any rotation about the \(y\)-axis becomes \(x\)-axis translation in the output image.
An example of the bunny model and the spherical-map transform of this model is given in figure 3, the mathematics are defined in equations 8 and 9. Given a coordinate in 2D Cartesian space x,y, we compute the ray \([Ray_x, Ray_y, Ray_z]\) from the volume center and sum up the voxel values along the ray (equation 9).

\[
\begin{align*}
Ray_x(x, y) &= \cos\left(\frac{360x}{N}\right) \sin\left(\frac{180y}{N}\right) + \frac{N}{2} \\
Ray_y(y) &= \cos\left(\frac{180y}{N}\right) + \frac{N}{2} \\
Ray_z(x, y) &= \sin\left(\frac{360x}{N}\right) \sin\left(\frac{180y}{N}\right) + \frac{N}{2}
\end{align*}
\] (8)

\[
Im_{x,y} = \sum_{r=1}^{(2^-1N)^3} Vol(Ray_x(x, y)r, Ray_y(y)r, Ray_z(x, y)r)
\] (9)

This process essentially sums up the values along a given ray defined by scaling spherical coordinates and adding up the values intersecting the ray. The resulting image, maps 3D y-axis rotation to 2D x-axis translation.

### 4.2. Projection-map transform

The projection map transform is similar to an orthogonal projection of the volume along some given axis. For the projection map transform, given an output image \(Im_a\) and an input volume \(Vol_a\), each pixel in \(Im_a\) is defined mathematically as the summation of values along a particular axis given the image coordinates. The x-axis transform and the z-axis transform are defined in equations 10 and 11 respectively.

\[
Im(z, y) = \sum_{x=0}^{N} Vol_a(x, y, z)
\] (10)

\[
Im(x, y) = \sum_{z=0}^{N} Vol_a(x, y, z)
\] (11)

The process defined by equation 10 maps 3D z-axis translation to 2D x-axis translation, whilst equation 11 maps 3D x-axis and y-axis translation into 2D x-axis and y-axis translation.

### 5. Performance Analysis

#### 5.1. 3D Phase Correlation Performance

To assess the performance of our method, the size of the volumes being registered is defined as \(N^3\) whilst each frame is sampled at a resolution of \(W \times H\). The projection process requires \(12WH\) operations whilst re-sampling the point cloud requires \(2WH\) operations. The Volume Registration process, \(VolumeRegister_{\theta\phi t}(V_1, V_2)\) consists of \(2 \times Hanning\) windowing processes, \(2 \times 3D\ FFTs, 2 \times \\log\)-spherical transforms, \(2 \times \ \log\)-spherical transforms, \(2 \times \ \log\)-spherical transforms, \(2 \times \ \log\)-spherical transforms, and \(1 \times \text{linear transformation and peak finding.}\)
The Hanning windowing function requires 26 operations. The 3D FFT has complexity of $3N^3 \log N$, the log and log-spherical transform functions require 3 and 58 operations per voxel respectively. Multiplying two frequency spectra together and transforming a volume requires 15 and 30 operations per voxel respectively. Finding the peak value requires $2N^2 \log N$ operations. The complexity in terms of number of operations for the phase correlation process is given in Eq. 12. This process requires $2 \times 3$D FFTs, $1 \times$ frequency spectra multiplication, and $1 \times$ peak finding operation.

$$6N^3 \log N + 2N^3 + 15 \quad \text{(12)}$$

The total complexity can then be found by taking into account the projection and re-sampling totals as well as the total for $VolumeRegister\theta\phi(t_1,t_2)\{V_1,V_2\}$. Tallying the number of operations for each process and multiplying them by number of times the process is performed gives us the number of operations as a function of $W$, $H$ and $N$ in Eq. 13.

$$6N^3 + 28WH + 18(N^3 \log N) + 230 \quad \text{(13)}$$

5.2. Analysis of Speed Improvement

To compare performance of the generic volume registration method with the speed up, we use the complexity defined in equation 13. Here, we ignore the cost of projecting the depth map. The 3D DFT has complexity $3N^3 \log(N)$. This is the first step (see figure 2), the next is the spherical-map transform which is complexity $45N^3$. If processed on the GPU the performance becomes 45 operations per voxel assuming that one voxel is assigned to one unit of processing. A 3D transform is 30 operations per voxel, 2D phase correlation requires 15 operations to multiply the frequency spectra and $2N^2 \log(N)$ operations to do the 2D FFT. Finally a projection map transform requires 1 operation per voxel.

In total, the proposed method requires $2 \times 3$D FFTs, $2 \times$ spherical-map transforms, $1 \times$ 3D geometrical transformation, $3 \times$ 2D phase correlations and $4 \times$ projection map transforms. The total complexity is added up for all of these functions and given in equation 14.

$$6 \log(N) \times (N^3 + N^2) + 169 \quad \text{(14)}$$

Figure 5 provides a visualization of the performance improvement which the proposed method achieves over the original Fourier volume registration approach. It is clear that the proposed method is around 3 times faster than the original Fourier based volume registration approach. This is due to the reduction in the amount of data to process afforded by the novel spherical-map transform and orthogonal projection methods.

6. Experiments

A number of experiments were undertaken to assess the reconstruction accuracy, noise sensitivity and robustness to object motion. Experiments were performed using an ASUS Zenbook UX303LN with an Intel i7 5500u Dual Core 2.4GHz processor, 8GB of RAM and an NVIDIA GE-FORCE 840 M GPU.
Figure 5. Comparison of performance between volume registration and the proposed speed up for different volume sizes.

Table 1. Translation Tracking

<table>
<thead>
<tr>
<th>translation (cm)</th>
<th>noise range (%)</th>
<th>SNR</th>
<th>error (cm)</th>
<th>error (voxel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5cm</td>
<td>0</td>
<td>∞</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5cm</td>
<td>10</td>
<td>20db</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5cm</td>
<td>25</td>
<td>12db</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5cm</td>
<td>50</td>
<td>6db</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5cm</td>
<td>75</td>
<td>2.5db</td>
<td>112.28</td>
<td>89.83</td>
</tr>
<tr>
<td>10cm</td>
<td>0</td>
<td>∞</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10cm</td>
<td>10</td>
<td>20db</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10cm</td>
<td>25</td>
<td>12db</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10cm</td>
<td>50</td>
<td>6db</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>15cm</td>
<td>0</td>
<td>∞</td>
<td>2.8</td>
<td>2.24</td>
</tr>
<tr>
<td>15cm</td>
<td>10</td>
<td>20db</td>
<td>2.8</td>
<td>2.24</td>
</tr>
<tr>
<td>15cm</td>
<td>25</td>
<td>12db</td>
<td>2.8</td>
<td>2.24</td>
</tr>
<tr>
<td>15cm</td>
<td>50</td>
<td>6db</td>
<td>198.55</td>
<td>158.84</td>
</tr>
</tbody>
</table>

For volumes of $384^3$, $1 \times$ registration per second was possible. To achieve real-time performance, 1 out of every 30th frames was processed.

[ht]

6.1. Reconstruction Quality

To assess reconstruction accuracy, two indoor environments (Apartment and Office) as well as one outdoor environment (Garden) were used, these can be seen in figures 4a, 4b and 4c respectively. The Apartment reconstruction was recorded by moving through a room whilst rotating the camera. Some frames contained nothing but featureless walls, others had contrast shifts due to the camera’s automatic contrast feature, yet, accurate reconstruction was achieved. The Office reconstruction was generated by rotating the camera about the y-axis while moving backwards. Whilst our method is a closed form solution, its accuracy is still comparable to existing feature based SLAM methods. Typical feature based methods work well with indoor environments where local features are readily distinguishable and easy to match. They do not tend to work as well with complex outdoor scenes where feature confusion is likely. To assess performance in such outdoor scenes, a garden scene containing bushes, plants and a ground covering of bark and rocks was used. In the case of a feature matching approach this scene would likely result in feature confusion, making camera tracking difficult. The proposed method was able to produce a good quality reconstruction. Hence, our approach readily overcomes difficulties common to feature matching methods.

6.2. Noise Sensitivity

To assess robustness to noise, the estimated camera parameters are compared to ground truth data under different noise conditions. In each experiment, varying amounts of random noise were added per voxel prior to registration. This is expressed in decibels using the Signal to Noise Ratio (SNR). Each voxel value lies in the range $[0-1]$. Here, a noise value of 10% means random noise was added in the range $[-0.05, 0.05]$. Tracking error is measured in centimetres and voxel error (the error in the phase correlation volume). The first experiment evaluated noise robustness whilst the camera was translated by varying amounts (5cm, 10cm and 15cm). Results in Table 1 show that, for camera translations up to 15cm and SNR values above 6.0 our method is robust to noise. At video rates, a
displacement of 10 cm per frame equates to a camera velocity of 3 m/s (about twice the normal walking speed).

Table 2 shows the results for tracking camera rotations of 10, 20 and 30 degrees per frame. At video rates, 12 degrees per frame is almost a full rotation per second. In rotations of 10 degrees, the error was less than a degree for all but a noise level of 30% and above. This base line error is due to the sampling resolution of the volume, as voxel error was in fact zero. As with pure translation, the effect of noise increases with camera translation, the effect of noise increases with camera translation.

6.3. Robustness to Object Motion

To assess robustness to object motion, experiments were conducted by moving the camera backwards along the z-axis by 5 cm per frame whilst moving objects in and out of the scene so that they only appear in one of the volumes being registered. Various sized objects including stacks of CDs, large boxes, people and several pieces of furniture were used and are measured by the percentage of the frame they occupy. Results from Table 3 show the proposed method was accurate up to an object size of 31%, but failed for objects taking up over 48.23%.

7. Conclusions

In this paper, we proposed a novel non-feature based approach to SLAM, which can generate accurate 3D color reconstructions of both indoor and outdoor environments. This method is a closed form solution, scales naturally to the GPU, and is shown to be robust to global noise and object motion. We also proposed a method to speed up this method by up to 3 times.

Table 2. Rotation Tracking

<table>
<thead>
<tr>
<th>rotation</th>
<th>noise (%)</th>
<th>SNR</th>
<th>error (°)</th>
<th>error (voxel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10°</td>
<td>0</td>
<td>∞</td>
<td>0.31</td>
<td>0</td>
</tr>
<tr>
<td>10°</td>
<td>10</td>
<td>20db</td>
<td>0.31</td>
<td>0</td>
</tr>
<tr>
<td>10°</td>
<td>25</td>
<td>12db</td>
<td>0.63</td>
<td>1</td>
</tr>
<tr>
<td>10°</td>
<td>30</td>
<td>10.5db</td>
<td>90.62</td>
<td>96</td>
</tr>
<tr>
<td>20°</td>
<td>0</td>
<td>∞</td>
<td>0.31</td>
<td>0</td>
</tr>
<tr>
<td>20°</td>
<td>10</td>
<td>20db</td>
<td>0.63</td>
<td>1</td>
</tr>
<tr>
<td>20°</td>
<td>15</td>
<td>16.5db</td>
<td>38.13</td>
<td>40</td>
</tr>
<tr>
<td>30°</td>
<td>0</td>
<td>∞</td>
<td>3.75</td>
<td>4</td>
</tr>
<tr>
<td>30°</td>
<td>10</td>
<td>20db</td>
<td>3.28</td>
<td>3</td>
</tr>
<tr>
<td>30°</td>
<td>15</td>
<td>16.5db</td>
<td>30</td>
<td>32</td>
</tr>
</tbody>
</table>

Table 3. Object Motion Test

<table>
<thead>
<tr>
<th>Object Size</th>
<th>error (cm)</th>
<th>error (voxel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.35</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2.95</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6.22</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12.28</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19.82</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22.39</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26.09</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>31.00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>48.23</td>
<td>38.42</td>
<td>15</td>
</tr>
<tr>
<td>74.32</td>
<td>113.57</td>
<td>44</td>
</tr>
</tbody>
</table>

8. Future Work

Future work in this area includes investigating a system to recover from mis-registered frames and to continue to improve performance.

Acknowledgement. Luke Lincoln would like to thank Thao Lincoln.

References


