A lightweight approach to 6-DOF plane-based egomotion estimation using inverse depth

Titus Jia Jie Tang, Wen Lik Dennis Lui and Wai Ho Li
Monash University, Australia
titus.tang@monash.edu, dennis.lui@monash.edu, wai.ho.li@monash.edu

Abstract

Egomotion estimation is a fundamental building block for self-localisation and SLAM, especially when an accurate map is not available to the robot. This paper presents a real time plane-based approach to estimate the 6 degrees of freedom (DOF) pose of a moving Microsoft Kinect. An inverse depth formulation is used to robustly fit multiple planes in the environment, which are then used to provide an egomotion estimate. Experiments presented demonstrate the algorithm's accuracy, repeatability and ability to handle different motions and scene compositions. Despite using full resolution Kinect depth images, our approach runs at 10-13 frames per second.

1 Introduction

Egomotion estimation is an important building block for self-localising autonomous systems in situations where a map is unavailable to the robot. This paper presents a computationally lightweight approach to egomotion estimation in 6 degrees of freedom using planes detected with a depth camera. An overview of our approach is shown in Figure 1.

Planes are detected using a Kinect which provides 2D depth images of a scene. Planes are modeled using inverse depth instead of Euclidean space, which is similar to the disparity-based approach of [Chumerin and Hulle, 2008]. Our inverse depth formulation allows a single threshold to be used for plane fitting while still naturally accounting for an increase in measurement error for points far away from the sensor. Egomotion is performed using 3 or more inverse depth planes, which provides a 6 degrees of freedom pose estimate in Euclidean space (3 parameters for translation and 3 parameters for orientation).

The proposed egomotion estimation approach differs from existing work on a number of fronts as detailed below

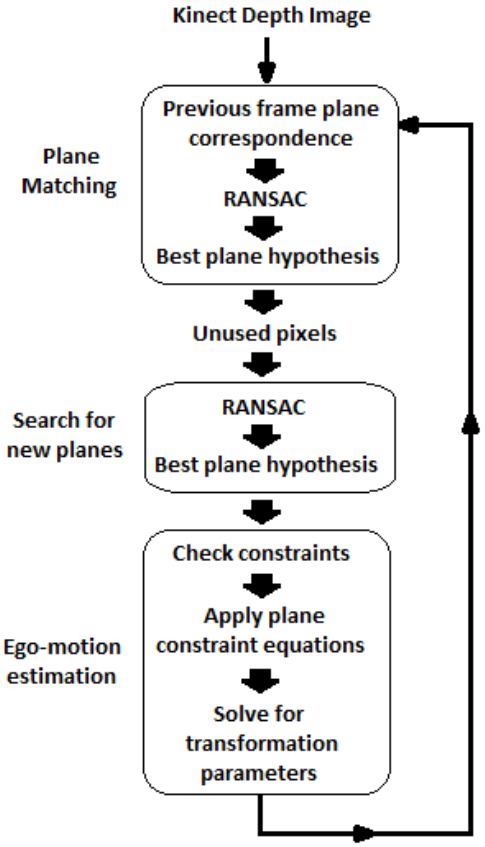


Figure 1: Flowchart for plane-based egomotion estimation algorithm.

- By contrast with plane-based egomotion estimation approaches [Nguyen *et al.*, 2007], our approach does not require orthogonality of planes but applies a more practicable constraint as detailed in Section 2.4.
- By contrast with vision-based plane detection and tracking approaches such as [Simon and Berger, 2002; Martinez-Carranza and Calway, 2009], our ap-

proach does not rely on visual features and surface texture but rather on spatial structure. A blank room is not a challenge to our system.

- By contrast with laser-based plane detection [Weingarten and Siegwart, 2006; Nguyen *et al.*, 2007; Martinez-Carranza and Calway, 2009], our approach operates on a depth camera as opposed to a mechanically swept high accuracy laser ranger. The lower accuracy and higher data density of the depth camera provides a different set of challenges to be solved, as detailed below.
- By contrast to existing plane-based approaches such as [Cobzas and Sturm, 2005], our approach does not require fiducial markers or human assistance in labeling of data or initialisation.

These are important incremental improvements that allow egomotion estimation to occur in real time within indoor environments using only depth imagery. However, in order to achieve real time egomotion estimation, we had to overcome two challenges:

- Depth data from the Kinect is roughly an order of magnitude less accurate than state of the art laser range finders. This poses a challenge when attempting to fit plane models as well as the subsequent step of egomotion estimation.
- The dense data provided by a depth camera as seen in Figure 2b can be difficult to deal with in real time, especially given that multiple planes must be detected per frame.

[Holz *et al.*, 2011] also uses a Kinect and plane fitting but for the purpose of object detection. Their algorithm runs at ∼7FPS on 640×480 images. Through a rapid RANSAC [Fischler and Bolles, 1981] based approach to plane fitting, our implemented system of plane matching and egomotion estimation is able to operate in real time at ∼10-13 FPS on images of the same resolution. Our inverse depth formulation of fitted planes also allows for the use of a single threshold during plane estimation, which provides computational savings as well as better management of measurement errors as the threshold *grows* at greater sensing distances.

The work presented here was carried out as a part of the Monash Vision Group (MVG), an ARC research centre dedicated to technologies that assist the visually impaired. Figure 3 shows a head mounted display (HMD) currently being used by MVG in psychophysics experiments where human subjects are artificially limited to low resolution similar to bionic vision. The HMD has been augmented with a Kinect sensor, which has already been used to develop novel sensing algorithms to generate low resolution bionic vision [mvg, 2011]. We plan to apply plane-based egomotion estimation and SLAM to further improve the quality of bionic vision as well as investigate non-invasive visual aids for those with low vision.

(a) Colour Image (b) Depth Image

Figure 2: An example of a colour image and depth image obtained from the Microsoft Kinect.

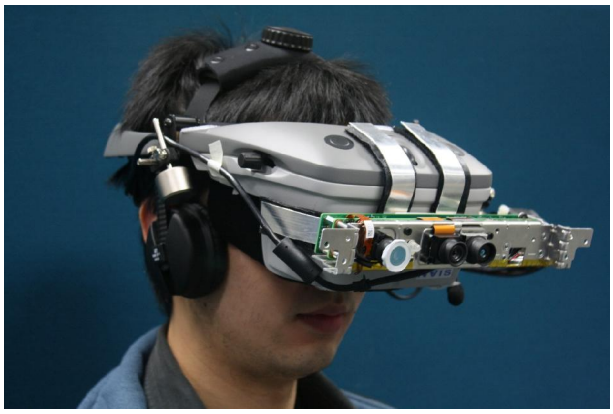


Figure 3: A Kinect-based head mounted display developed by the Monash Vision Group

The rest of this paper is structured as such: In Section 2, we provide details of the methodology used in our approach to the problem of plane-based egomotion estimation. In Section 3, we provide experimental results on testing the accuracy, reliability and limits of the algorithm. We conclude with a discussion and proposals for future work.

2 Methodology

We present our methodology in accordance to the flow of the algorithm as seen in Figure 1. We begin by providing an overview of the relationships between image pixel coordinates, UVQ space and Euclidean space. Next, we discuss our plane fitting methodology using RANSAC in UVQ space, and then proceed to explain the difference between plane matching versus new plane detection. Finally, we state the constraints and provide mathematical derivation for our egomotion estimation algorithm.

2.1 UVQ coordinates

In this paper, we define UVQ coordinate space as the set of coordinate axes centered on the sensor's principle point, with the U-axis parallel to the images row pixels, V-axis parallel to the images column pixels, and the Q-axis pointing out of the sensor. Assuming knowledge of the sensor's intrinsic parameters, the UVQ coordinates of a point P(u,v,q) is obtained from pixel coordinates as follows:

$$u = \frac{j - C_u}{f_u} \tag{1}$$

$$v = \frac{i - C_v}{f_v} \tag{2}$$

In the direction of the column and row index respectively, j and i are the pixel coordinates in the image, C_u and C_v are the principal point offsets, f_u and f_v are the focal lengths. Using UVQ coordinates is advantageous to our application as it allows us to work with an isotropic and homogeneous error model of the depth data obtained from the Kinect. If we define in a similar fashion an Euclidean coordinate frame centered on the sensor's principle point, we can relate points in UVQ space to points in Euclidean space by dividing the Euclidean coordinates with the z term:

$$u = \frac{x}{z}, v = \frac{y}{z}, q = \frac{1}{z} \tag{3}$$

or

$$\begin{bmatrix} u \\ v \\ 1 \\ q \end{bmatrix} = \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4}$$

Having derived the relationships between pixel, UVQ and Euclidean coordinate, we thus consider conversions between coordinate systems trivial for the remainder of this paper.

2.2 Plane detection

RANdom SAmple Concensus (RANSAC) [Fischler and Bolles, 1981] forms the core of plane detection in the depth image in the work discussed here. Three random points are sampled from the depth image to form a plane hypothesis containing the plane normal in UVQ space. A plane normal in UVQ space is obtained from the following derivation.

The Euclidean form of a plane equation is given as:

$$Ax + By + Cz + D = 0 \tag{5}$$

Dividing each term by the z and D parameter gives us,

$$\frac{Ax}{Dz} + \frac{By}{Dz} + \frac{C}{D} + \frac{1}{z} = 0 \tag{6}$$

Using the relationship between UVQ and Euclidean coordinates from Equation 4, we get

$$\alpha u + \beta v + \gamma = q \tag{7}$$

α, β and γ form the plane model in UVQ space, π, and can be written as

$$\alpha = \frac{-A}{D}, \beta = \frac{-B}{D}, \gamma = \frac{-C}{D} \tag{8}$$

The obtained plane model is then fitted to the entire sample space of points and points having a distance to the plane less than a threshold T_1 are accepted as inliers.

$$|\alpha u_i + \beta v_i + \gamma - q_i| < T_1 \tag{9}$$

Note that T_1 is a constant since working in inverse depth coordinates results in an isotropic and homogeneous error model. We emphasise this as one of the main advantages of working in UVQ coordinates. The value of T_1 is determined empirically by the degree of trade off between plane stability and robustness, and may vary depending on the accuracy required of the application. The above process is repeated for a fixed number of iterations and the plane model with the largest number of inliers is then accepted as a detected plane in the image. We then fit this plane model to all remaining points in the sample space using a second constant threshold T_2 to determine all points in the image that belong to this plane. Note that $T_1 < T_2$. The determination of a suitable plane model is done with a smaller threshold to reduce the variability of planes found in consecutive frames due to noise in the depth image.

2.3 Plane matching versus new plane detection

Using the method presented in the previous section, the detection of planes in the environment can be classified into 1) the detection of previously found planes, and 2) the detection of planes new in the field of view. The main difference between the two methods is in the range of the sample space of points in which RANSAC is performed. For each new input image, the algorithm first attempts to find matches for planes found in the previous image before moving on to detect new planes. Matching of planes between two images is performed by superimposing the plane segment found in the previous image onto the new image and then performing RANSAC on points confined to this segment. The algorithm is implemented with the assumption of small inter-frame displacements. Having found a plane match, all points belonging to that plane are removed from the sample space of points and the algorithm iterated for remaining planes. This generally reduces computational time per iteration as the algorithm progresses in its search.

Upon completion of matching of all possible planes, we then proceed to search for new planes in the remaining point set. Again, such a method of match-first-search-last greatly reduces the time needed to search for a suitable new plane model in the remaining sample space. The search for new planes will proceed until either a fixed maximum number of planes are found or until the remaining points in the sample space are too few to form a sufficiently large plane.

2.4 Egomotion estimation

In this paper, we refer to egomotion as the estimation of a sensor's motion in a rigid scene in three dimensional space relative to the sensor coordinate frame defined by the sensor's initial pose. Egomotion estimation is performed by applying a motion model to a plane constraint equation as discussed in the following subsections.

Motion Model

Let R_x, R_y and R_z be the angle of rotation about the X, Y, and Z axis in Euclidean space, and let T_x, T_y and T_z be the translational distance along the X, Y, and Z axis respectively. Assuming small inter-frame translations and rotations, the transformation matrix relating points on a plane in frame t-1 to frame t is defined as:

$$M = \begin{bmatrix} 1 & -R_z & R_y & T_x \\ R_z & 1 & -R_x & T_y \\ -R_y & R_x & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

Egomotion estimation and constraints

The estimation of six degrees of freedom egomotion through plane correspondences is accomplished under the following constraints:

1. At least six points are available to be sampled from the planes to allow the formation of sufficient equations to solve for the six transformation parameters.
2. For any axis in any three-dimensional coordinate system of any orientation, there exists at least one successfully tracked plane with a normal vector having a non-zero component in the direction of that axis.

Described in accordance to Figure 4, constraint 2 can be re-worded as such: For the set of tracked planes used in egomotion estimation, place the x-axis of a Euclidean coordinate system perpendicular to the first plane; the y-axis perpendicular to the x-axis and the line of intersection of the first plane and second plane. For the remaining planes, there should exist at least one plane having a non-zero component along the z-axis. The necessity of this constraint can be explained with the following scenario. Consider an example scene such as in Figure 5.

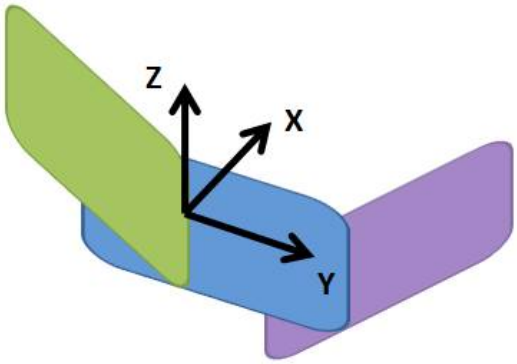


Figure 4: A scenario where egomotion estimation would fail due to zero components of the normal of all planes along the Z-axis.

(a) Colour Image (b) Detected Planes

Figure 5: An example scene where egomotion estimation would fail due to the poor configuration of planes which do not provide sufficient motion constraint as defined in constraint 2.

By moving along any line parallel to the line of intersection of any two planes in the scene, the system would be unable to determine if it was moving since we assume all planes to be infinite and the angle and distance between the sensor and either plane does not change. Another plane having a non-zero component of the normal vector in the direction parallel to the line of intersection of the former two planes is required to constraint the system for 6 degree of freedom pose determination.

In estimating egomotion, the first step we take is to identify suitable planes. A first test would be to determine if a plane is detected in both the current and previous frame and has been successfully matched between frames. Next, we search for the intersection point of any combination of three planes in the set of matched planes. The intersection point of three planes is calculated as follows:

$$P = \frac{d_1 \cdot (N_2 \times N_3) + d_2 \cdot (N_3 \times N_1) + d_3 \cdot (N_1 \times N_2)}{N_1 \cdot (N_2 \times N_3)} \tag{11}$$

where a plane equation can be written as

$$N_i \cdot p = d_i \tag{12}$$

N_i is the plane normal, d_i is the distance of the point p from the plane having N_i as its normal. The symbol '$\times$' represents a cross product and '$\cdot$' represents a dot product. Determining the presence of an intersection point allows us to ensure that constraint (2) is satisfied. The presence of a set of planes that do not satisfy constraint (2) such as in Figure 5 would intersect along lines or not intersect at all.

Finally, we apply the points used to form the planes obtained from the RANSAC procedure in the previous section to the plane-to-plane constraint below. The points are chosen because their distances to the planes π_{t-1} are equal to zero.

$$\pi_t \cdot M \cdot P_{t-1} = \pi_{t-1} \cdot P_{t-1} \tag{13}$$

where π is a plane model in UVQ space, M is the transformation matrix from Equation 10, P is the 3D coordinates of a point in UVQ space. The index t and t-1 refers to the current frame and previous frame respectively. By multiplying the terms and rearranging to separate the transformation parameters we get:

$$L \cdot M_T = R \tag{14}$$

where

$$L = \begin{bmatrix} -\beta_t + \gamma_t v_{t-1} & \dots & \dots \\ \alpha_t - u_{t-1}\gamma_t & \dots & \dots \\ \beta_t u_{t-1} - \alpha_t v_{t-1} & \dots & \dots \\ \alpha_t q_{t-1} & \dots & \dots \\ \beta_t q_{t-1} & \dots & \dots \\ \gamma_t q_{t-1} & \dots & \dots \end{bmatrix}^T \tag{15}$$

$$M_T = \begin{bmatrix} R_x & R_y & R_z & T_x & T_y & T_z \end{bmatrix}^T \tag{16}$$

$$R = \begin{bmatrix} u_{t-1}(\alpha_{t-1} - \alpha_t) + v_{t-1}(\beta_{t-1} - \beta_t) + \gamma_{t-1} - \gamma_t \\ \dots \\ \dots \end{bmatrix} \tag{17}$$

Each row in L and R corresponds to the values of a single point on its respective plane. The total number of rows in L and R corresponds to the number of sample points selected from the planes (Constraint 1) and must be at least six. Using least squares to solve for M_T, we obtain the following:

$$M_T = (L^T L)^{-1} L^T R \tag{18}$$

The transformation parameters are then used to update the current position in each consecutive frame.

3 Results

This section presents the results of four different sets of experiments with the following aims:

- Section 3.1 - To determine the repeatability of the plane detection algorithm.
- Section 3.2 - To evaluate the effects of scene variation on the performance of the algorithm.
- Section 3.3 - To determine the maximum tolerable interframe displacement.
- Section 3.4 - To evaluate the performance of the algorithm with a hand-held Kinect in 6-DOF motion.

The Kinect depth sensor returns a depth image with a resolution of 640 by 480 pixels (after interpolation from 320 by 240 pixels). Our algorithm is single-threaded and implemented using OpenCV libraries on Ubuntu. We conducted all experiments with the same set of algorithm parameters. The un-optimised implementation of our algorithm runs at 10-13 frames per second on an Intel i7 2.93 GHz processor. While using a desktop computer does not provide the best mobility, we expect to be able to run the algorithm on portable computers in the near future after having perform code optimisation and parallelisation.

3.1 Plane normal variance

As mentioned above, the detection of planes is performed by repeatedly and *randomly* selecting 3 points to form a hypothesis and subsequently testing this hypothesis against the set of available points. The stochastic nature of this algorithm results in slight variations in the detected plane normal of a same plane between two consecutive frames even if no motion is involved. We aim to determine the variations in the detected plane normal and its significance to our subsequent experimental results.

A static Kinect was set up to face a single static plane (a white wall). We varied the distance between the camera and the wall within its recommended operating distance [OpenKinect, 2011] and took one thousand measurements of the detected plane normal at each location. Referring to Table 1, the maximum variances of all plane normal components (α, β, γ) at all of the measured distances is 3.5×10^{-5}. In other words, the randomness in the RANSAC procedure has negligible effect on the repeatability of plane detection using a static Kinect.

Component of normal	α	β	γ
Maximum variance	1.0×10^{-5}	0.7×10^{-5}	4×10^{-7}

Table 1: Maximum variances for each plane normal component.

Data Set	**Motion Parameters**					
	R_x	R_y	R_z	T_x	T_y	T_z
Bare	-0.11	0.00	0.02	0.002	0.000	0.000
Slightly Cluttered	-0.30	-0.01	0.03	0.008	0.000	0.001
Very Cluttered	0.01	-0.01	-0.01	0.000	0.000	0.000
Hand-held (Bare)	-0.25	0.70	0.06	0.008	0.003	-0.001
Hand-held (Cluttered)	0.01	0.00	0.00	0.000	0.000	0.000

Table 2: Offset between start and end positions obtained from the ICP algorithm for all data sets presented in this paper. Rotational parameters in degrees; Translational parameters in metres.

3.2 Evaluation of effects of scene variation

In this set of experiments, we aim to test the effects of variations in the environment on the performance of the algorithm. Specifically, we test our algorithm in three different environments: 1. A bare, empty scene; 2. A slightly cluttered scene; 3. A very cluttered and disorganised scene (Figure 6a,b,c respectively), and then compare the performances across these scenes.

As standardisation of methodology across all experiments, we hold as constant across experiments the trajectory traversed by the sensor by using a turntable. We start the execution of the algorithm with the sensor placed at a fixed known position on the edge of the turntable. We then turn the turntable 90 degrees clockwise and subsequently in the reverse direction back to the starting position (a total of 180 degrees) while running the algorithm in real-time. Since it is impossible to accurately return the sensor to the initial starting location with our current setup, we apply the Iterative Closest Point (ICP) algorithm [Besl and McKay, 1992] to the first and last frames of the image sequence to determine the deviations between start and end positions. By using all data points available and allowing the algorithm to run until convergence, the ICP algorithm allows accurate registration/alignment between the two depth images. Taking this offset into consideration (Table 2), it is then possible to determine the ground truth of the end position. On the other hand, the theoretical trajectory traversed by the sensor is calculated by knowing the position of the sensor relative to the turntable, and the angle of the sensor determined from the Kinect's accelerometer readings.

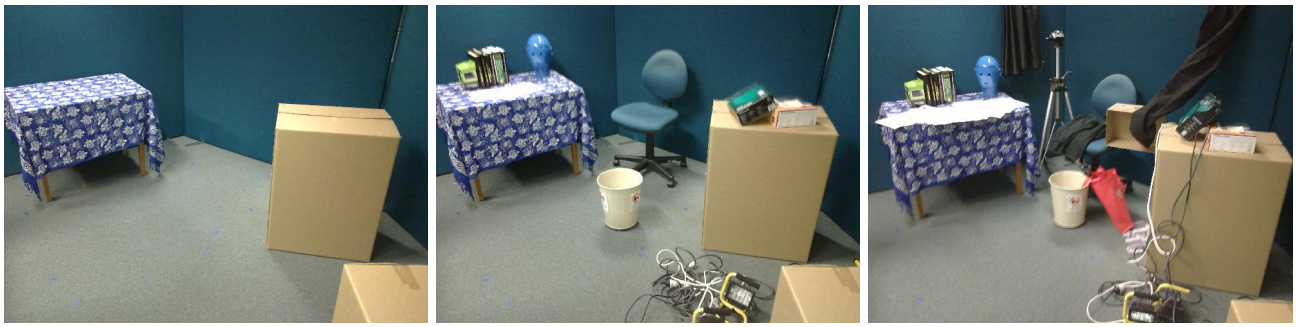
The algorithm was repeated offline on the same data sets 100 times, each initialised with a different random seed, to determine the mean and root-mean-squared (RMS) egomotion errors after returning the camera back to the initial position. Table 3 summarises the results. Figure 7 shows, for each data set, an example of the traversed trajectory as estimated by the algorithm overlaid onto the theoretical trajectory.

Results indicate that our proposed algorithm is able to handle both simplistic and cluttered environments as long as the fundamental constraints as detailed in Section 2.4 are satisfied. However, the algorithm's performance begins to deteriorate when clutter in the environment is excessive to the point where they prevent the consistent detection of planar surfaces, such as from certain view points traversed by the sensor in the very cluttered scene (Figure 6c).

3.3 Maximum tolerable interframe displacement

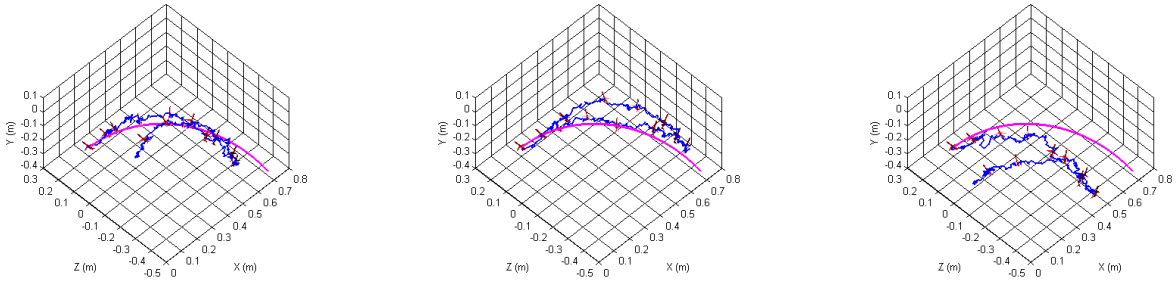
In this section, we are interested in understanding the maximum tolerable interframe translations and rotations of the algorithm by increasing the observed motion velocities of the sensor until the algorithm fails. Using the same data set from the slightly cluttered scene from the previous section, we simulate increased motion velocities by subsampling the data set at 2, 3 and 5 frame intervals. (For example, subsampling at a 3 frame interval means only frames 1, 4, 7, 10... etc are used). Egomotion mean and RMS errors are determined using the same procedures as in the previous section.

Figure 8 shows an example of the traversed trajectory as estimated by the algorithm overlaid onto the theoretical trajectory for each subsampling rate. Table 4 presents the outcome of this subsampling process along with the approximate actual rotational and translational interframe displacement experienced by the sensor. Rotational displacements are the limiting factor in this set of experiments as the translational displacements are comparatively small. Results indicate that the algorithm is able to handle average rotations of 1 degree between frames without suffering deterioration in performance. At a frame rate of 12 frames per second, this translates to an approximate rotation of 12 degrees per



(a) Bare, empty scene (b) Slightly cluttered scene (c) Very cluttered scene

Figure 6: Environments with varying amount of clutter used to test the robustness of the algorithm.



(a) Bare, empty scene (b) Slightly cluttered scene (c) Very cluttered scene

Figure 7: Plot of a single estimated trajectory (blue line) overlaid onto the theoretical trajectory (pink curve) for each of the respective scenes. The red arrows indicate the estimated orientation of the sensor at that point in time.

Scene Type	**Error**	**Motion Parameters**						**Frame rate**
		R_x	R_y	R_z	T_x	T_y	T_z	
Bare	Mean	3.82	2.98	3.01	0.062	0.123	0.084	12.7
	RMS	4.29	3.62	3.65	0.077	0.136	0.091	
Slightly cluttered	Mean	3.54	2.89	3.48	0.057	0.125	0.070	11.3
	RMS	4.12	3.69	4.13	0.080	0.183	0.086	
Very cluttered	Mean	4.21	5.44	4.22	0.073	0.164	0.097	10.6
	RMS	5.24	6.70	4.97	0.104	0.212	0.132	

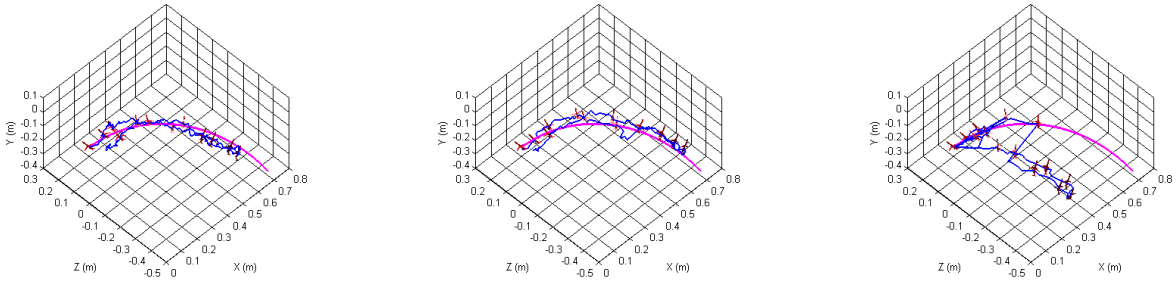
Table 3: Mean and root-mean-squared egomotion error over 100 runs for each scene data set, along with algorithm frame rate. Rotational parameters in degrees; Translational parameters in metres.

second; The data set subsampled at 3 frame intervals (approximately 1.4 degrees of rotation between frames) results in mean and RMS egomotion estimation errors twice as large; while subsampling at 5 frames (approximately 2.4 degrees of rotation between frames) causes the algorithm to perform too poorly for any reasonable egomotion estimation (Figure 8c).

3.4 6-DOF egomotion estimation using a hand-held Kinect

In this section, we evaluate the ability of the algorithm to handle 6-DOF motions of a hand-held Kinect. We adopt experimental methodologies from the previous experiments where applicable, with the major difference being that the Kinect is moved around the environment by hand instead of a turntable. As with previous experiments, we use the ICP algorithm to determine the deviations between start and end positions of the Kinect located atop a fixed tripod (Figure 9b).

We test the algorithm in two scenes - A simple scene and a cluttered scene (Figure 9). The actual trajectories used in the experiments involve translations between 3 to 5 metres and rotations of roughly 90 degrees. An ex-



(a) 2-frame subsampling (b) 3-frame subsampling (c) 5-frame subsampling

Figure 8: Plot of a single estimated trajectory (blue line) overlaid onto the theoretical trajectory (pink curve) for each of the respective subsampling rates. The red arrows indicate the estimated orientation of the sensor at that point in time.

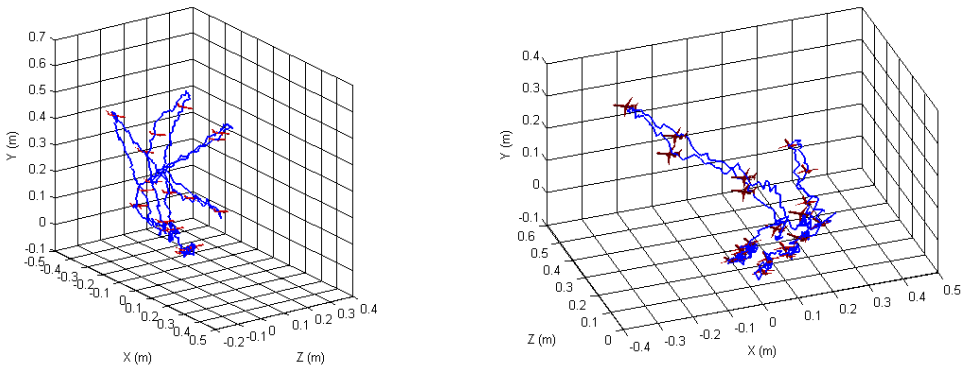
Subsampling rate	**Approx. actual interframe**		**Error**	**Motion Parameters**					
	Rotation	**Translation**		R_x	R_y	R_z	T_x	T_y	T_z
None	0.5	0.005	Mean	3.54	2.89	3.48	0.057	0.125	0.070
			RMS	4.12	3.69	4.13	0.080	0.183	0.086
2 frames	1.0	0.010	Mean	3.28	2.63	2.55	0.052	0.121	0.071
			RMS	3.53	3.20	3.42	0.072	0.182	0.098
3 frames	1.4	0.015	Mean	3.13	3.28	2.61	0.062	0.270	0.138
			RMS	3.41	4.43	3.54	0.084	0.382	0.204
5 frames	2.4	0.025	Mean	4.86	17.18	8.26	0.133	0.788	0.356
			RMS	7.30	16.06	10.60	0.191	1.342	0.602

Table 4: Approximate actual rotational and translational interframe displacement experienced by the sensor for each rate of subsampling, along with respective mean and root-mean-squared egomotion error over 100 runs. Rotational parameters in degrees; Translational parameters in metres.

(a) Simple scene (b) Cluttered scene

Figure 9: The environments in which the performance of the algorithm was tested using a hand-held Kinect. The Kinect is shown in its initial position on a tripod to the left of the cluttered scene.

ample of the estimated trajectories are shown in Figure 10 and the egomotion errors presented in Table 5. Average drift accumulated per frame indicate egomotion estimation accuracy and repeatability on par with that of experiments presented in the previous sections. As with all experiments presented in this paper, please view the video attached to this paper for a more detailed understanding of the motion trajectories.



(a) Simple scene (b) Cluttered scene

Figure 10: Plot of a single estimated trajectory (blue line) overlaid onto the theoretical trajectory (pink curve) for each of the respective scenes using a hand-held Kinect. The red arrows indicate the estimated orientation of the sensor at that point in time.

Data Set	**Error**	**Motion Parameters**					
		R_x	R_y	R_z	T_x	T_y	T_z
Simple	Mean	3.26	6.57	4.54	0.093	0.085	0.045
	RMS	3.84	6.92	4.04	0.112	0.130	0.071
Cluttered	Mean	6.66	5.59	4.46	0.135	0.125	0.098
	RMS	5.01	6.46	4.68	0.165	0.154	0.119

Table 5: Mean and root-mean-squared egomotion error over 100 runs for each hand-held, 6-DOF motion data set. Rotational parameters in degrees; Translational parameters in metres.

4 Discussion and future work

In the experiments presented in this paper, we evaluated the accuracy and repeatability of our algorithm using a fixed trajectory with the help of a turntable. We further demonstrated our algorithm's robustness in being able to operate in various environments and with varying speeds. The algorithm was also tested to its limits of egomotion estimation by increasing the difficulty of the scene and speed of motion until egomotion estimation failure. Taking a step further in the analysis of the limits of the algorithm, we note a number of failure modes not immediately obvious from the presented experimental results. Some of these limitations stem from the assumptions made while deriving the proposed method of egomotion estimation. The following are examples of such situations.

- The assumption of a motion model based on small interframe motions (Equation 10) has limited the performance of the algorithm in situations where the sensor is being moved rapidly, especially in the rotational sense. The simplified motion model was used because of its mathematical simplicity, which subsequently reduces the computational cost of having to calculate values of Sines and Cosines in the complete motion model. Using the complete model would increase mathematical accuracy but would also reduce frame rate - the benefits are self-cancelling. The limitations of small interframe motions can be a significant problem if we are to deploy our algorithm for use by a human user (our main motivation - as an aid for the visually impaired). We intend to overcome this limitation in the future by combining multi sensory input using a Kalman filter (e.g. a gyroscope to track large user head movement in combination with the plane-based egomotion estimation algorithm).
- Our plane-based egomotion estimation algorithm would require the detection of at least three non-parallel planes. Although most indoor environments do satisfy such a constraint, there are situations in which the algorithm would fail to perform, such as when travelling down a long, empty corridor, or when facing a large wall in close proximity. Again, a Kalman filter may benefit us by allowing the integration of the results of two or more complimentary egomotion estimation algorithms (e.g. ICP based or visual feature based egomotion estimation algorithms with the plane based algorithm).

- Our current implementation does not take into consideration moving objects in the field of view. Moving objects with planar surfaces may be detected as planes and mistakenly taken into consideration while estimating egomotion.
- We do not perform loop closure in the work presented in this paper. As a result, drift from egomotion estimation may accumulate over time and become significant over longer trajectories.

Other proposals for the extension and improvement to the work presented here include:

- Implementing variations to the standard RANSAC algorithm.
- Conducting a thorough parameter sweep to determine the best set of parameters for our application, if such a set of parameters exists.
- Implementing re-weighted least squares in our plane matching and selection algorithm to improve on our handling of planes with high noise content.

5 Conclusion

In this paper, we present a method for six degree of freedom egomotion estimation through the detection and matching of plane correspondences between consecutive images. Our approach is based on the use of depth images obtained from the Microsoft Kinect sensor and subsequently processed in inverse depth. Experiments conducted using a turntable, a hand-held Kinect, and in scenes of varying difficulty demonstrate proof of concept and provide understanding on the range of performance of the algorithm. Our un-optimised implementation of the algorithm runs at 10-13 frames per second.

Acknowledgment

The authors wish to thank the Monash Vision Group for providing the laboratory space for our experiments. This work was supported by the Australian Research Council Special Research Initiative in Bionic Vision and Sciences (SRI 1000006).

References

[Besl and McKay, 1992] P. J. Besl and H. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[Chumerin and Hulle, 2008] N. Chumerin and M. M. V. Hulle. Ground plane estimation based on dense stereo disparity. In *Fifth international conference on Neural Networks and Artificial Intelligence*, 2008.

[Cobzas and Sturm, 2005] D. Cobzas and P. Sturm. 3d ssd tracking with estimated 3d planes. In *2nd Canadian Conference on Computer and Robot Vision*, 2005.

[Fischler and Bolles, 1981] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Graphics and Image Processing*, 24(6):381–395, 1981.

[Holz *et al.*, 2011] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. Real-time plane segmentation using rgb-d cameras. In *Proceedings of the 15th Robocup Symposium*, 2011.

[Martinez-Carranza and Calway, 2009] J. Martinez-Carranza and A. Calway. Appearance based extraction of planar structure in monocular slam. In *Proceedings of the Scandinavian Conference on Image Analysis (SCIA)*, 2009.

[mvg, 2011] System and method for processing sensor data for the visually impaired, 2011.

[Nguyen *et al.*, 2007] V. Nguyen, A. Harati, and Roland Siegwart. A lightweight slam algorithm using orthogonal planes for indoor mobile robotics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[OpenKinect, 2011] OpenKinect. Kinect depth camera accuracy. www.openkinect.org/wiki/Imaging_Information, 1 September 2011.

[Simon and Berger, 2002] G. Simon and M. O. Berger. Pose estimation for planar structures. *IEEE Computer Graphics and Applications*, 22(6):46–53, 2002.

[Weingarten and Siegwart, 2006] J. Weingarten and R. Siegwart. 3d slam using planar segments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.