Robust Egomotion Estimation using ICP in Inverse Depth Coordinates

Wen Lik Dennis Lui, Titus Jia Jie Tang, Tom Drummond and Wai Ho Li

***Abstract*— This paper presents a 6 degrees of freedom egomotion estimation method using Iterative Closest Point (ICP) for low cost and low accuracy range cameras such as the Microsoft Kinect. Instead of Euclidean coordinates, the method uses inverse depth coordinates which better conforms to the error characteristics of raw sensor data. Novel inverse depth formulations of point-to-point and point-to-plane error metrics are derived as part of our implementation. The implemented system runs in real time at an average of 28 frames per second (fps) on a standard computer. Extensive experiments were performed to evaluate different combinations of error metrics and parameters. Results show that our system is accurate and robust across a variety of motion trajectories. The point-to-plane error metric was found to be the best at coping with large inter-frame motion while remaining accurate and maintaining real time performance.**

I. INTRODUCTION

Egomotion estimation is the problem of estimating a moving camera's pose relative to a base coordinate frame, such as the camera's initial position, from a sequence of images. This is important to a wide range of applications, including mobile robotics [1] and augmented reality [2]. In this paper, we present a 6 degrees of freedom (DoF) egomotion estimation approach for low accuracy range cameras such as the widely available (and low cost) Microsoft Kinect. Our system estimates the pose of a range sensor relative to a starting position using a novel inverse depth formulation of the Iterative Closest Point (ICP) algorithm; departing from the traditional approach of using Euclidean coordinates.

Benefits of using inverse depth include,

- Near-isotropic error for range sensors where uncertainty of depth measurements increases for points further from the sensor.
- Avoid unnecessary conversion of disparity-derived sensor data (i.e. the raw depth output of the Kinect) into 3D Euclidean space; a space which has non-homogeneous and non-isotropic noise characteristics. The conversion also unnecessarily increases computational cost.

The main contributions of this paper are,

- Theoretical and practical implementations of two new ICP variants that use inverse depth coordinates; one using a point-to-point error metric and the other using a point-to-plane error metric.
- Robust and real time 6 DoF egomotion estimation using low accuracy depth images from the Kinect.
- Performance evaluation of our system under various algorithm changes to the system (i.e. error metrics, weighting function, convergence thresholds, etc).
- Extensive experiments using synthetic and real world data to evaluate the real time performance of our system under different 3D scenes.

Summarily, our approach differs from existing work in egomotion estimation as follows,

- By contrast to approaches that utilise high accuracy laser range finders [3] or Time-of-flight (ToF) range cameras [4], our approach only relies on low accuracy depth images. Note that the ToF approach also uses additional sensing modalities (reflectance and inertial) whereas our approach only uses depth information.
- By contrast to stereo vision approaches formulated in terms of disparity [5], our approach does not rely on the detection of visual features or image pixel correlations. We operate directly on the inverse depth *point cloud* produced by a range camera.
- By contrast to the Kinect-based egomotion estimation approach of [6] that uses visual features as well as depth information, our approach only uses the Kinect's depth image. By forgoing the use of computationally expensive visual features, our approach is able to operate in real time at Kinect frame rate. Moreover, our approach does not require any assumptions regarding the visual content and illumination of a scene.
- More generally, the proposed ICP formulation uses inverse depth coordinates unlike the 3D Euclidean space used by most ICP variants (i.e. [7]). The use of inverse depth coordinates naturally conforms to the disparity-based depth images produced by the Kinect.

II. RELATED WORK

Egomotion estimation based on range data is a well established field of research. ICP, introduced concurrently in the early 1990s by [8], [9], [10], is a common algorithm used to estimate the egomotion of range sensors. ICP works by iteratively matching points between time-adjacent range data *frames* to converge upon an estimated sensor pose change that *explains* the point movements.

Historically, egomotion estimation using ICP is performed using high accuracy range data from laser-based range finders. Since many range finders have a wide field of view (FoV), this greatly simplifies the registration and alignment of data as the overlapping region between consecutive scans increases. Thorough surveys of egomotion estimation using range data are available from [11], [12].

W.L.D. Lui, T.J.J. Tang, T. Drummond and W.H. Li are with ECSE, Monash University, Australia. W.L.D. Lui, T.J.J Tang and W.H. Li are also with the Monash Vision Group. Email: dennis.lui@monash.edu

Relative to laser-based range finders, Time of Flight (ToF) range cameras have lower accuracy and a narrower FoV. Similarly, active stereo range cameras, such as the Kinect, have even lower accuracy (lower than ToF sensors) and also have a narrow FoV. This limitation is generally addressed by using visual information to detect stable and unique features to improve the robustness of frame-to-frame pose estimates. For example, the work in [4] uses visual features (based on reflectance), ToF range data and inertial estimates to produce egomotion estimates in 3 DoF.

The Kinect-based work in [6] also uses visual features in conjunction with depth images. The system does not run in real time due to the use of computationally expensive visual features as described in the paper. The use of visual features also introduces the implicit disadvantage of requiring scenes with sufficient visual content and consistent illumination.

An alternative method to improve the accuracy of pose estimates is to replace the frame-to-frame tracking approach by a frame-to-model approach as in [7]. Pose estimation, between the Kinect's depth images and its evolving global surface model, is performed by using a point-to-plane ICP variant in a coarse-to-fine registration scheme that rejects outliers based on distance and surface normal difference thresholds. In our approach, the proposed ICP variant operates in inverse depth coordinates and applies a different outlier rejection scheme. As such, our work is focused on the evaluation and comparison of two ICP error metrics formulated in inverse depth coordinates under a frame-to-frame tracking approach which can be extended to a frame-to-model approach.

The proposed approach in this paper also has theoretical similarities with disparity-based stereo vision egomotion estimation. Particularly, it is similar to the feature-based approach in [5], which operates directly in disparity space (linearly proportional to inverse depth for fronto-parallel stereo vision systems).

III. System Overview

The system is made up of an Intel i7 2.8GHz computer with 16Gb RAM and a Microsoft Kinect that features an infrared (IR) projector-camera pair and a RGB camera (refer to Fig.1). Color and disparity-based depth images are produced by the RGB camera and IR projector-camera pair respectively at 30fps. The generation of depth images is achieved based on the detected IR speckle (pseudo random dot pattern) that is created by emitting IR laser through a diffraction grating. As such, the inexpensive and widely available Kinect works well in most indoor environments. Moreover, the Kinect can still provide depth images in absolute darkness due to the use of an active light source for depth estimation.

As described by the inventors in [13], depth is measured via a triangulation process that is based on the detection of transverse shifts of local dot patterns in the IR speckle with respect to its reference patterns at a known distance to the device. This process is repeated for all local regions in the IR speckle and produces a disparity-based depth image. The standard treatment to such data is to transform it into the 3D Euclidean coordinate system, producing a 3D point cloud.

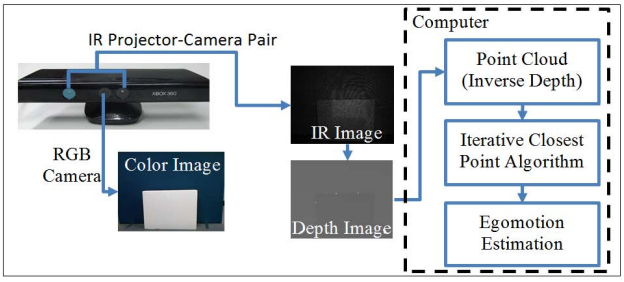


Fig. 1. System Flowchart

As reported in [14], [15], the normalised disparity values returned by the Kinect are inversely proportional to its inverse depth. Additionally, [16], [17] show that there is a non-linear relationship between the normalised disparity values and its depth value in Euclidean space. As such, representing the data in inverse depth coordinates is arguably the most suitable since it preserves the linear relationship with the normalised disparity values and avoids conversion to 3D Euclidean space which has non-homogeneous and non-isotropic noise. Furthermore, [18] illustrated that the depth measurement errors are nearly linear in inverse depth coordinates, making it simpler to propagate its associated uncertainties.

Given a point, $p_e = [x, y, z, 1]^T$, in Euclidean space represented using homogeneous coordinates, its equivalent p, in inverse depth coordinates is,

$$p = 1/z \begin{bmatrix} x & y & z & 1 \end{bmatrix}^T = \begin{bmatrix} u & v & 1 & q \end{bmatrix}^T \quad (1)$$

Given the intrinsic parameters of the depth camera, p can be directly derived from the disparity image as

$$p = \begin{bmatrix} u & v & 1 & q \end{bmatrix}^T = \begin{bmatrix} \frac{i-c_i}{f_i} & \frac{j-c_j}{f_j} & 1 & q(d_{ij}) \end{bmatrix}^T \quad (2)$$

where (i, j) is the image coordinate, (c_i, c_j) is the principal point offset, (f_i, f_y) is the focal length, d_{ij} is the normalised disparity value at image coordinate (i, j) and the function $q(d_{ij})$, which describes the relationship between d_{ij} and q, is modeled as

$$q(d_{ij}) = k_1 d_{ij} + k_2 \quad (3)$$

Based on the experimental results in [14], k_1 and k_2 were approximated as $k_1 = -0.00307$ and $k_2 = 3.33095$. The values of k_1 and k_2 can be further refined using the calibration technique proposed in [19].

By applying Equation 2 to all pixels in the depth image produces a point cloud in inverse depth coordinates. Using the inverse depth point clouds from two adjacent time steps (i.e. t and t+1), we apply the ICP algorithm for motion estimation. The final egomotion of the sensor is then obtained through the incremental accumulation of the small motion estimates between consecutive adjacent time steps. This is summarised into the system flowchart in Fig 1.

IV. ICP in Inverse Depth Coordinates

The proposed ICP algorithm distinguishes itself from existing ICP variants in the literature due to the representation of the depth image in inverse depth coordinates. Nevertheless, the standard stages of the algorithm still applies. According to the categorisation in [20], our proposed ICP algorithm can be described by the following stages,

1) **Selection** - Our ICP variant performs random sampling of points from the depth image at time t.
2) **Matching** - Establish correspondences for points sampled in the selection stage from the depth image at time $t+1$. Our ICP variant applies the project and walk technique to establish point correspondences as proposed in [21] using the nearest neighbour criteria.
3) **Weighting** - Emphasise the quality of correspondences based on some criteria (i.e. compatibility of normals, measurement uncertainties). Our ICP variant applies reweighted least squares to reject outlying measurements. This will be detailed later in Section IV-E.
4) **Rejection** - Our ICP variant rejects point correspondences that exceed a specified distance threshold.
5) **Error metric** - Our ICP variant minimises an error metric formulated in inverse depth coordinates to estimate the motion parameters that would best align the images at time t and t+1.
6) **Iterate** from Step (1) until convergence.

Before proceeding to the description of the point-to-point and point-to-plane error metric in inverse depth coordinates, we will firstly describe the motion model and its underlying assumptions.

A. Motion Model

The following describes the 6 DoF motion model assuming small incremental rotations ($sin(\theta) = \theta$ and $cos(\theta) = 1$) and translations,

$$\mathcal{M} = \begin{bmatrix} 1 & -\theta_z & \theta_y & t_x \\ \theta_z & 1 & -\theta_x & t_y \\ -\theta_y & \theta_x & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

where θ_x, θ_y and θ_z describe the rotations about the x, y and z axes and t_x, t_y and t_z describe the translations along the x, y and z axes.

The Kinect's pose at time t+1 relative to its starting pose, T_0^{t+1}, is updated from its previous pose, T_0^t, by some small incremental motion described by $\mathcal{M}_t^{t+1}$ with T_0=I where I is a 4x4 identity matrix. This is expressed as,

$$T_0^{t+1} = \mathcal{M}_t^{t+1} T_0^t \tag{5}$$

B. Point-to-Point Error Metric

Each corresponding point established in Step (2) of the ICP algorithm can be described as,

$$p_{t+1} = \mathcal{M}_t^{t+1} p_t \tag{6}$$

$$\begin{bmatrix} u_{t+1} & v_{t+1} & 1 & q_{t+1} \end{bmatrix}^T = \mathcal{M}_t^{t+1} \begin{bmatrix} u_t & v_t & 1 & q_t \end{bmatrix}^T \tag{7}$$

The Jacobian for a point pair, J_p, is obtained by finding the partial derivative of $(u_{t+1}, v_{t+1}, q_{t+1})$ with respect to the six unknown motion parameters $(t_x, t_y, t_z, \theta_x, \theta_y, \theta_z)$.

$$J_p = \begin{bmatrix} q_t & 0 & -u_t q_t & -u_t v_t & 1+u_t^2 & -v_t \\ 0 & q_t & -v_t q_t & -1-v_t^2 & v_t u_t & u_t \\ 0 & 0 & -q_t^2 & -q_t v_t & q_t u_t & 0 \end{bmatrix} \tag{8}$$

The full Jacobian, J, is constructed by stacking m number of J_p resulting from each corresponding pair on top of each other such that,

$$J = \begin{bmatrix} J_{p,1} & J_{p,2} & \dots & J_{p,m} \end{bmatrix}^T \tag{9}$$

The six unknown motion parameters can then be estimated using the following least squares solution,

$$\hat{\mathcal{B}} = \left(J^T J\right)^{-1} J^T Y \tag{10}$$

where

$$\hat{\mathcal{B}} = \begin{bmatrix} t_x \\ t_y \\ t_z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \text{ and } Y = \begin{bmatrix} u_{1,t+1} - u_{1,t} \\ v_{1,t+1} - v_{1,t} \\ q_{1,t+1} - q_{1,t} \\ \vdots \\ u_{m,t+1} - u_{m,t} \\ v_{m,t+1} - v_{m,t} \\ q_{m,t+1} - q_{m,t} \end{bmatrix} \tag{11}$$

C. Point-to-Plane Error Metric

The point-to-plane error metric expressed in normal least squares form is,

$$E = \sum_{i=1}^{m} \left[\left(\mathcal{M}_t^{t+1} p_{i,t} - p_{i,t+1} \right) \cdot n_i \right]^2 \tag{12}$$

where $n_i = \begin{bmatrix} \alpha_i & \beta_i & \gamma_i & 0 \end{bmatrix}^T$ is the surface normal of the point $p_{i,t+1}$ and $(\cdot)$ represents a dot product. The 6 motion parameters in Equation 4 are then recovered using the following least squares solution,

$$\hat{\mathcal{B}} = \left(K^T K\right)^{-1} K^T Y \tag{13}$$

where $\hat{\mathcal{B}}$ is described in Equation 11 and K and Y are expressed as the following matrices:

$$K = \begin{bmatrix} q_{1,t}\alpha_1 & \dots & q_{m,t}\alpha_m \\ q_{1,t}\beta_1 & \dots & q_{m,t}\beta_m \\ q_{1,t}\gamma_1 & \dots & q_{m,t}\gamma_m \\ v_{1,t}\gamma_1 - \beta_1 & \dots & v_{m,t}\gamma_m - \beta_m \\ \alpha_1 - u_{1,t}\gamma_1 & \dots & \alpha_m - u_{m,t}\gamma_m \\ u_{1,t}\beta_1 - v_{1,t}\alpha_1 & \dots & u_{m,t}\beta_1 - v_{m,t}\alpha_1 \end{bmatrix}^T \tag{14}$$

$$Y = \begin{bmatrix} -\left(p_{1,t} - p_{1,t+1}\right) \cdot n_1 \\ \vdots \\ -\left(p_{m,t} - p_{m,t+1}\right) \cdot n_m \end{bmatrix} \tag{15}$$

D. Estimating Surface Normals for Point-to-Plane

The point-to-plane error metric in Equation 12 requires the surface normal of corresponding points to be estimated. Assuming that the small neighbourhoods surrounding corresponding points in the depth image are locally planar, their surface normals can then be estimated by plane fitting. A plane in Euclidean space is described by,

$$ax + by + cz + d = 0 \tag{16}$$

This can be expressed in inverse depth coordinates as,

$$\alpha u + \beta v + \gamma = q \tag{17}$$

where $\alpha = -a/d$, $\beta = -b/d$ and $\gamma = -c/d$. The surface normal of a point, p_{t+1} at image location (i_c, j_c), is estimated using least squares on the set of points within its local neighbourhood in the disparity image (i.e. patch size of $w \times h$) as,

$$n = \left(L^T L\right)^{-1} L^T Q \tag{18}$$

where

$$n = \begin{bmatrix}\alpha & \beta & \gamma\end{bmatrix}^T, L = \begin{bmatrix} u_{i,j} & v_{i,j} & 1 \\ \vdots & \vdots & \vdots \end{bmatrix}, Q = \begin{bmatrix} q_{i,j} \\ \vdots \end{bmatrix} \tag{19}$$

In Equation 19, (i, j) are the image coordinates within the local neighbourhood of point p_{t+1} centered at (i_c, j_c). Inspired by the work in [22], we reduce the total time required to estimate surface normals by precomputing $\left(L^T L\right)^{-1} L^T$ into a lookup table.

E. Robust Estimation of Motion Parameters

The standard least squares solutions presented in Sections IV-B and IV-C are not robust in the presence of outliers. Standard approaches to reduce the vulnerability of the system towards the presence of outliers are by applying RANSAC or substitute the least squares estimator with a robust M-estimator which applies less weighting to outlying measurements. The latter approach is chosen here for being more efficient to compute. The weighting function chosen is,

$$w(\triangle r) = 1/\left(\sigma + \triangle r^2\right) \tag{20}$$

where σ is the standard deviation of the data based on estimates from the previous iteration and $\triangle r$ is the residual of the objective function for the corresponding point-to-point or point-to-plane pair. Equations 10 and 13 are replaced by,

$$\hat{\mathcal{B}} = \left(J^T W J\right)^{-1} J^T W Y \tag{21}$$

$$\hat{\mathcal{B}} = \left(K^T W K\right)^{-1} K^T W Y \tag{22}$$

where W is a diagonal matrix weighing the point-to-point or point-to-plane correspondences based on Equation 20.

Similarly, [23] has taken into account the uncertainties associated with the laser rangefinder's measurements and weigh corresponding pairs higher when the measurement uncertainty is lower and vice-versa. The main difference lies in the selection of the weighting function, defined as the reciprocal of the measurement variance. This results in a more complicated solution due to the need to accurately propagate the uncertainties.

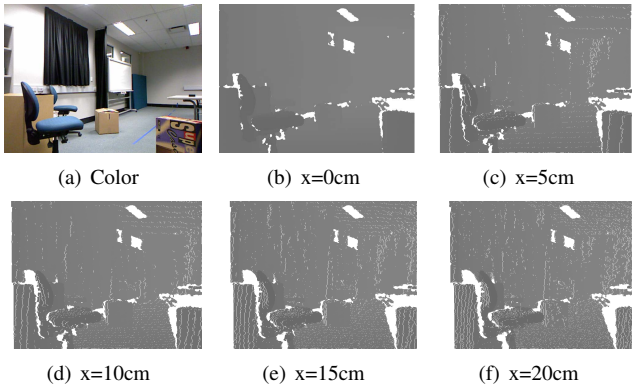
V. EXPERIMENTAL RESULTS

In this section, we will examine the performance of the following error metrics,

- Standard point-to-point error metric (no weights)
- Weighted point-to-point error metric
- Standard point-to-plane error metric (no weights)
- Weighted point-to-plane error metric

Referring to Section IV, the proposed ICP variant is divided into 6 steps. In Step (1), 500 points are randomly sampled in Step (1) from the reference depth image at time t. In Step (2), corresponding points are established in the depth image at time $t+1$ using the project and walk method with a neighbourhood size of 7x7 based on the nearest neighbour criteria and correspondences with distances greater than 0.01 (in inverse depth space) are rejected in Step (4). For the point-to-plane error metric, the surface normal of corresponding points are estimated using Equation 18 with a 3x3 local neighbourhood. The performance of the four error metrics are evaluated using test sequences with synthesised motion and in real world experiments described in the following subsections. All reported running times are obtained from a single threaded C++ implementation using the OpenCV and OpenKinect libraries on an Intel i7 2.8GHz processor with 16Gb of RAM.

A. Test Sequences with Synthesised Motion



(a) Color (b) x=0cm (c) x=5cm

(d) x=10cm (e) x=15cm (f) x=20cm

Fig. 2. Test Sequence with Synthesised Translational Motion

Test sequences used in this evaluation were generated from a single depth image and they replicate some small motion experienced by the Kinect. The two test sequences generated were a pure translational sequence along the x-axis and a pure rotational sequence about the y-axis. The translational sequence consists of 21 frames where the first frame is the original depth image at 0cm translation and consecutive frames are synthesised depth images with inter-frame translations of 1cm along the x-axis. The rotational

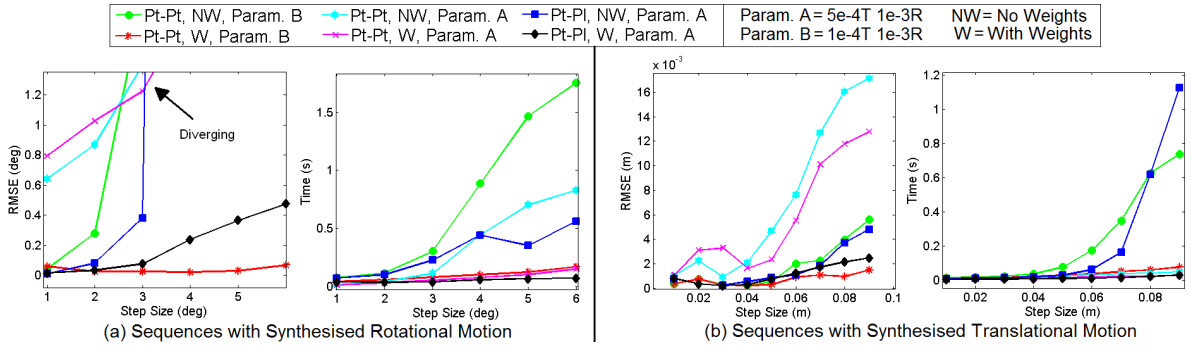


Fig. 3. Results of Test Sequences with Synthesised Motion: Step size indicates the interframe translations or rotations, RMSE for each error metric is derived from 100 trials on the same test sequence, the **T** label in the legend indicates the threshold for ROC of translational parameters and **R** for rotational parameters.

sequence consists of 11 frames where the first frame is the original disparity image at 0° rotation and consecutive frames are synthesised depth images with interframe rotations of 1° about the y-axis.

The first step in generating the test sequences with synthesised motion is to convert the original depth image into a point cloud in inverse depth coordinates. Some motion is applied to the point cloud by using the motion matrix described in Equation 4. Then, the transformed point cloud is back-projected to produce the synthesised depth image. The original depth image used in generating the test sequence is shown in Fig. 2(b) together with the color image of the scene in Fig. 2(a). Fig. 2(c)-2(f) show synthesised depth images at the respective translations from the original depth image. Note that the proposed method for generating test sequences is unable to cope with large motions due to the severity of missing data and aliasing issues.

Based on the two test sequences, the performance of the four error metrics can be evaluated based on the root-mean-square error (RMSE) for different interframe translation and rotation. For example, the translational test sequence has 21 frames with 1cm interframe translation. By skipping every other frame (1,3,5,...), the error metrics can be tested on a sequence with 2cm interframe translation and so on. The availability of the test sequences also allowed us to extensively evaluate the performance of the four error metrics when different thresholds were used to determine convergence. These thresholds specify when the rate of change (ROC) in the estimated motion parameters are small enough to determine that the algorithm has converged. This is important since it provides the algorithm with an additional criteria to break out from the iterative process before the maximum number of iterations specified is reached. The maximum number of iterations was specified as some large number (i.e. 2000) in this evaluation.

We performed an extensive parameter sweep for the following thresholds,

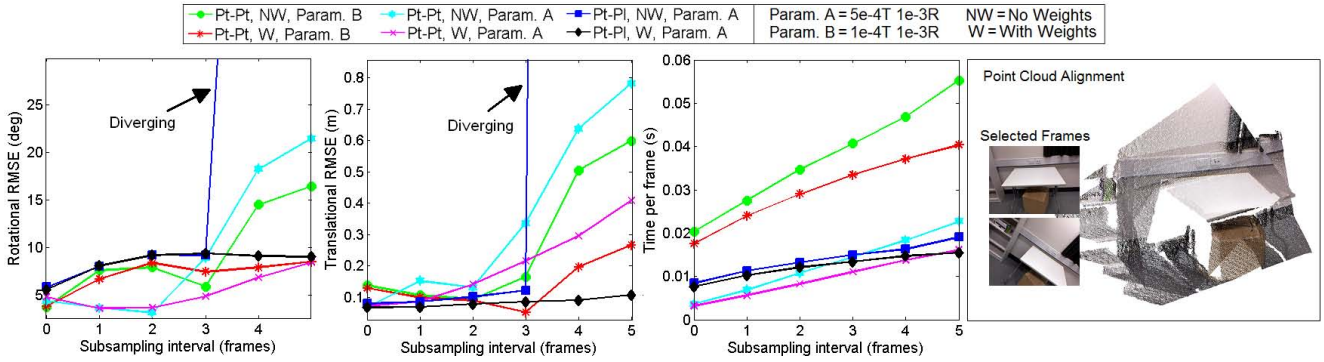
- Threshold for ROC of estimated translational motion parameters ranging from $1e^{-4}$ to $9e^{-4}$ metres at step sizes of $1e^{-4}$.
- Threshold for ROC of estimated rotational motion parameters ranging from $5e^{-4}$ to $4e^{-3}$ radians at step sizes of $1e^{-4}$. We discovered later that the range of values that was being tested for this threshold have negligible effect on overall speed and accuracy. As such, we have chosen $1e^{-3}$ for this threshold.

In this work, the target is to achieve real-time egomotion estimation for a handheld Kinect. In order to satisfy the real-time constraint, the algorithm is expected to converge within 50ms at a translation of 5cm or 3° per frame. As a result, we have selected the thresholds for the error metrics which best satisfy the requirements and illustrated their performances on the rotational and translational test sequences in Fig. 3. The graphs in Fig. 3 clearly show that

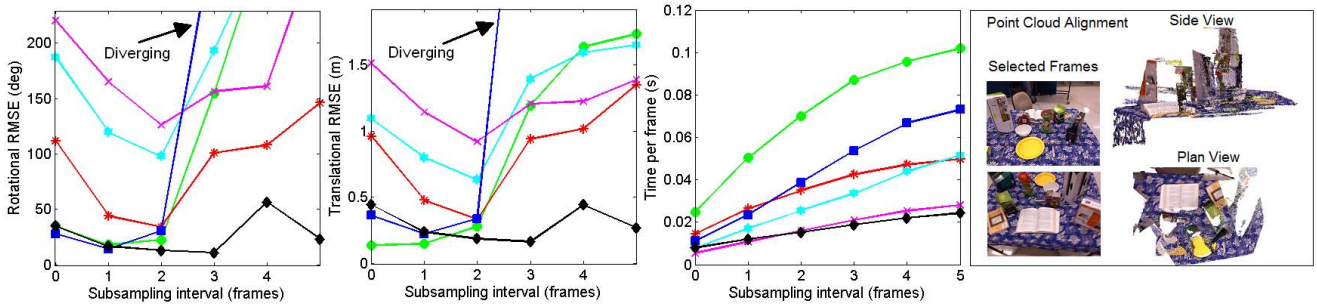
- the weighted error metrics converge quicker and accumulate less error.
- the unweighted error metrics become highly unstable in the rotational test at larger interframe rotation.
- the best performing weighted point-to-point error metric in terms of speed is with the thresholds $5e^{-4}T$ and $1e^{-3}R$. However, this is not as accurate when the following thresholds were used: $1e^{-4}T$ and $1e^{-3}R$ (T represents the threshold for ROC of translational parameters and R for rotational parameters)
- the overall best performing algorithm (best balance between speed and accuracy) is the weighted point-to-plane error metric with $5e^{-4}T$ and $1e^{-3}R$. Nevertheless, this is not as accurate as the weighted point-to-point error metric with $1e^{-4}T$ and $1e^{-3}R$ but it has the advantage of being faster.

B. Real World Experiments

Based on the results from the test sequences with synthesised motion, we will evaluate the performance of the error metrics in real world experiments using the selected thresholds in Fig. 3. For each selected error metric, we have determined the max. number of iterations to run based on the average number of iterations it took to converge at 5cm translation or 3° rotation (whichever is larger) as reported by the test sequence evaluation. The experimental setup is shown in Fig. 5. In Fig. 5(a), the Kinect is placed in a plain



(a) Experiment in Plain Scene



(b) Experiment in Cluttered Scene

Fig. 4. Rotational RMSE, Translational RMSE, Average Time/Frame and Selected Point Cloud Alignment for Real World Experiments: RMSE for each error metric was derived from 25 trials on the same real world sequence, the **T** label in the legend indicates the threshold for ROC of translational parameters and **R** for rotational parameters, and alignment of point cloud is achieved for two selected frames (one from initial starting position and another from mid-trajectory) using estimated sensor pose.

scene (less geometrical variations) whereas in Fig. 5(b), the Kinect is placed in a cluttered scene (geometrically rich).

(a) Plain Scene (b) Cluttered Scene

Fig. 5. Experimental Setup

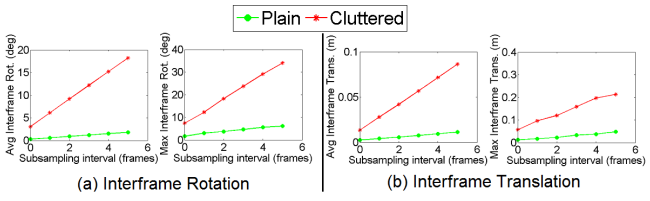
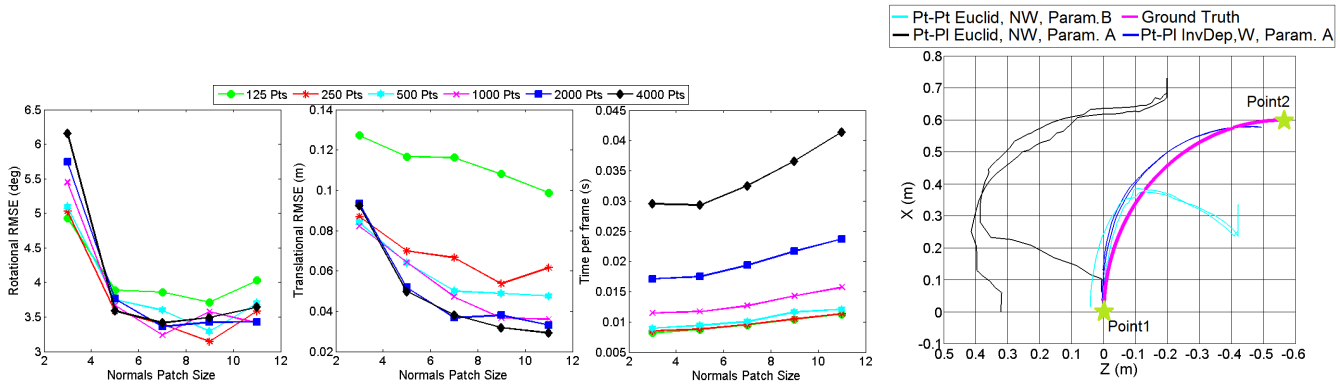


Fig. 6. Average and Maximum Interframe Rotation and Translation: Real world experiments at different subsampling intervals (subsampling interval of 0 indicates the original sequence captured with the weighted point-to-plane metric in real-time)

As discovered in the test sequence evaluation, the weighted point-to-plane error metric with thresholds $5e^{-4}T$ and $1e^{-3}R$ reported the best overall speed and accuracy. Real world experiments were conducted in the plain and cluttered scenes using the weighted point-to-plane error metric with the average framerate reported as 28 fps.

In the real world experiments, the Kinect would start and stop at approximately the same pose on top of a tripod. Nevertheless, there may be some small deviations between the first and last frame of the motion sequences. These deviations can be accounted for by using an accurate version of ICP. Unlike the ICP variants proposed in this paper, this version of ICP is allowed to take as many iterations as required and does not need to satisfy the real-time constraint. We have selected the weighted point-to-point error metric with thresholds $1e^{-4}T$ and $1e^{-3}R$ for this task as it was found to be the most accurate variant. In fact, for such small deviations between two frames, any of the weighted error metrics can produce reasonably accurate results.

The real world experiments were conducted using the weighted point-to-plane error metric at 28 fps and the depth images were stored in memory during run-time. The depth images in memory were only written to disk after the program has terminated. The proposed method for storing the image sequences has negligible effect on run-time performance and allows us to test the same sequence on the remaining error metrics. Most importantly, this allows us to subsample the saved image sequences and evaluate the



(a) (b)

Fig. 7. (a) Effect on accuracy and computational time by variation of total points sampled and normals patch size on the plain scene dataset using the point-to-plane error metric with thresholds $5e^{-4}T$ and $1e^{-3}R$ and (b) Comparison of Inverse Depth (InvDep) and Euclidean (Euclid) ICP variants (W=With Weights, NW=No Weights, Param. A = $5e^{-4}T$ $1e^{-3}R$, Param. B = $1e^{-4}T$ $1e^{-3}R$) on Turntable Experiment. Ground truth trajectory: Point1→Point2→Point1 as marked in diagram.

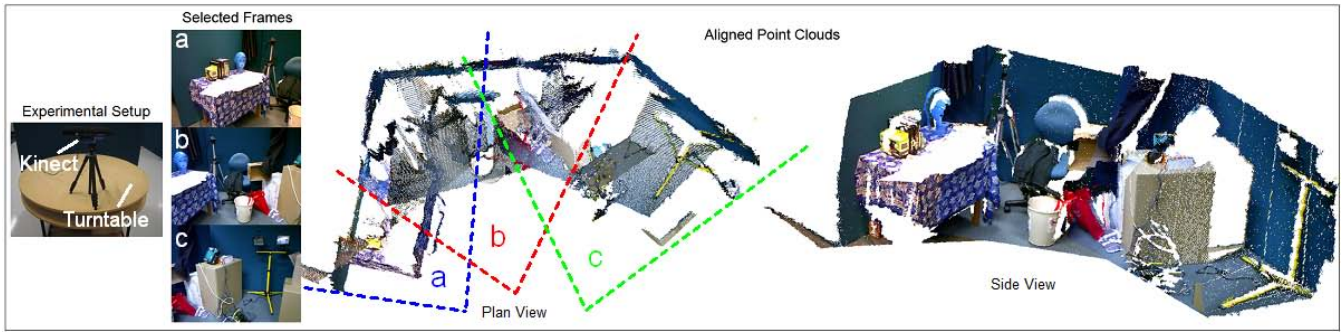


Fig. 8. Experimental setup and point cloud alignment using estimated sensor pose for frames a, b and c selected from the Turntable Experiment. The accuracy of the point cloud alignment is clearly demonstrated by the accurate line up of walls shown in the plan view.

performance of the system on the subsampled sequence to determine when the system would fail. If the original image sequence is $1, 2, 3, \ldots, n$, a subsampled sequence with 1 frame interval is $1, 3, 5, \ldots, n$. The subsampled sequences increase the interframe motion experienced and allow us to push the system to its limits.

The rotational and translational RMSE reported by the 6 selected error metrics with different thresholds for each sequence subsampled at intervals of 0 to 5 frames are illustrated in Fig. 4. Motion sequences subsampled at intervals of 0 frame are the original sequences. Additionally, the average computational time per frame is provided (take note that the speed of motion is not constant). Fig. 6 provides an indication of interframe motion experienced by the Kinect in the subsampled real world sequences. *For more information, please refer to attached video.*

Similar to the test sequence evaluation, the real world experiments show that the point-to-plane error metric with thresholds $5e^{-4}T$ and $1e^{-3}R$ provides the best overall speed and accuracy. In light of this observation, an additional test was performed to show the effect on accuracy and computational time by varying the total points sampled and patch size for normal estimation. The results from this test are illustrated in Fig. 7(a).

C. Turntable Experiment

Due to the inherent difficulty in obtaining ground truth measurements for the previous experiments (apart from the RMSE between the first and last frames and point cloud alignment), our aim in this experiment is to provide additional insight into the accuracy of the proposed ICP variant by comparing the sensor's estimated trajectory with respect to the ground truth trajectory. The ground truth trajectory can be recovered since the sensor is positioned on top of a manual turntable as shown in Fig. 8. Nevertheless, a quantitative frame by frame comparison is not possible without a motorised turntable. The trajectories can be plotted to scale but comparison is more qualitative than quantitative.

In this experiment, we compared the estimated trajectories of our proposed weighted point-to-plane ICP variant in inverse depth coordinates and the unweighted Euclidean implementations of ICP by [24] with respect to the ground truth trajectory illustrated in Fig. 7(b). Using selected frames from the sequence and the estimated sensor pose from the weighted point-to-plane error metric with thresholds $5e^{-4}T$ and $1e^{-3}R$, we illustrate the aligned point clouds in Fig. 8.

VI. DISCUSSION

The results from the experiments show that,

- The weighted error metrics are more robust and are able to tolerate larger interframe rotation and translation.
- The dips in Fig. 4(b) were due to the larger drift accumulated from a longer sequence (smaller subsampling interval). RMSE increases when the system is pushed to its limits at larger subsampling intervals.
- In comparison, the weighted and unweighted point-to-point error metrics with thresholds $5e^{-4}T$ and $1e^{-3}R$ illustrated in Fig. 4(a) may initially indicate higher accuracy as it reported lower rotational RMSE. In fact, when combined with its translational RMSE, it has actually accumulated a larger drift (less accurate).
- The current parameters used (refer to Section IV) were suboptimally selected and were geared towards a balance in computational efficiency and accuracy. As illustrated in Fig. 7(a), accuracy can be improved by using a larger patch size for normal estimation in this particular dataset with a minor increase in computational time. This may be attributed to the scene being largely planar.
- Fig. 7(b) shows the accuracy of our proposed ICP variant in comparison to its Euclidean counterpart. This is not an exhaustive comparison to all existing Euclidean ICP variants in the literature but it does clearly show the accuracy of our proposed ICP variant in inverse depth coordinates with robust outlier rejection.
- The real world experiments clearly agree that the point-to-plane error metric with thresholds $5e^{-4}T$ and $1e^{-3}R$ provides the best overall speed and accuracy.

Based on the previous observations, we conclude that results from the real world experiments and test sequences with synthesised motion are consistent. The only inconsistency is the accuracy of the weighted point-to-point error metric with thresholds $1e^{-4}T$ and $1e^{-3}R$. This error metric was found to be the most accurate variant in the test sequence evaluation but this is not true in the real world experiments when larger interframe motion is present. Nevertheless, it still outperforms the unweighted variants.

Possible extensions to this work include,

- Method for optimal parameter selection.
- Development of multiscale variant (coarse-to-fine registration) and GPU implementation of ICP for better run-time performance as in [7].
- Incorporating information filters, multi-frame adjustment or SLAM techniques to further improve the robustness and scalability of the system.

VII. CONCLUSION

This paper presented an ICP-based egomotion estimation method using inverse depth coordinates. The method runs in real time and operates directly on the disparity-based depth images from low cost range cameras such as the Microsoft Kinect. Experiments showed that the method is robust under a variety of trajectories and environments. The weighted point-to-plane error metric was found to be the best metric for speed and accuracy across all test datasets.

VIII. ACKNOWLEDGMENTS

This work was supported by the Australian Research Council Special Research Initiative in Bionic Vision and Sciences (SRI 1000006).

REFERENCES

[1] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," *Journal of Intelligent Robotics Systems*, vol. 18, no. 3, pp. 249–275, 1997.

[2] D. Chekhlov, A. P. Gee, A. Calway, and W. Mayol-Cuevas, "Ninja on a plane: Automatic discovery of physical planes for augmented reality using visual slam," in *ISMAR*, 2007.

[3] A. Hoover, D. Goldgof, and K. Bowyer, "Egomotion estimation of a range camera using the space envelope," *IEEE Transactions on Systems , Man and Cybernetics*, vol. 33, no. 4, pp. 717–721, 2003.

[4] D. Droeschel, S. May, D. Holz, P. Ploeger, and S. Behnke, "Robust ego-motion estimation with tof cameras," in *European Conference on Mobile Robots*, 2009.

[5] D. Demirdjian and T. Darrell, "Egomotion estimation from disparity images," in *IEEE International Conference on Computer Vision*, vol. 1, 2001.

[6] M. Fiala and A. Ufkes, "Visual odometry using 3-dimensional video input," in *Canadian Conference on Computer and Robot Vision*, pp. 86–93, 2011.

[7] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE International Symposium on Mixed and Augmented Reality*, 2011.

[8] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[9] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *IEEE International Conference on Robotics and Automation*, 1991.

[10] Z. Zhang, "Iterative point matching for registration of free-form curves," Tech. Rep. Research Report 1658, INRIA, 1992.

[11] B. Sabata and J. K. Aggarwal, "Estimation of motion from a pair of range images: A review," *CVGIP: Image Understanding*, vol. 54, no. 3, pp. 309–324, 1991.

[12] J. Salvi, C. Matabosch, D. Fofi, and J. Forest, "A review of recent range image registration methods with accuracy evaluation," *Image and Vision Computing*, vol. 25, no. 5, pp. 578–596, 2007.

[13] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns," 2008. Prime Sense Ltd.

[14] ROS. http://www.ros.org/wiki/kinect_node/Calibration, 2011.

[15] K. Khoshelham, "Accuracy analysis of kinect depth data," in *ISPRS Workshop on Laser Scanning*, vol. XXXVII, 2011.

[16] ROS. http://www.ros.org/wiki/openni_kinect/kinect_accuracy, 2011.

[17] J. Stowers, M. Hayes, and A. Bainbridge-Smith, "Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor," in *IEEE International Conference on Mechatronics*, pp. 358–362, 2011.

[18] J. Civera, A. J. Davison, and J. Montiel, "Inverse depth to depth conversion for monocular slam," in *IEEE International Conference on Robotics and Automation*, pp. 2278–2783, 2007.

[19] D. Herrera, J. Kannala, and J. Heikkilä, "Accurate and practical calibration of a depth and color camera pair," in *International Conference on Computer Analysis of Images and Patterns*, pp. 437–445, 2011.

[20] S. Rusinkiewicz, "Efficient variants of ICP algorithm," in *Proceedings of the 3rd International Conference on 3D Digital Imaging and Modeling*, pp. 145–152, 2001.

[21] R. Benjemaa and F. Schmitt, "Fast global registration of 3d sampled surfaces using a multi-z-buffer technique," in *First International Conference on Recent Advances in 3D Digital Imaging and Modeling*, pp. 113–120, 1997.

[22] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *IEEE International Conference on Robotics and Automation*, 2011.

[23] C. Dorai, J. Weng, and A. K. Jain, "Optimal registration of object views using range data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1131–1138, 1997.

[24] J. Wilm, "Matlab central - iterative closest point." http://www.mathworks.com/matlabcentral/fileexchange/27804-iterative-closest-point, July 2011.