# Incremental Registration of RGB-D Images

Ivan Dryanovski*, Carlos Jaramillo and Jizhong Xiao, *Senior Member, IEEE*

***Abstract*— An RGB-D camera is a sensor which outputs range and color information about objects. Recent technological advances in this area have introduced affordable RGB-D devices in the robotics community. In this paper, we present a real-time technique for 6-DoF camera pose estimation through the incremental registration of RGB-D images. First, a set of edge features are computed from the depth and color images. An initial motion estimation is calculated through aligning the features. This initial guess is refined by applying the Iterative Closest Point algorithm on the dense point cloud data. A rigorous error analysis assesses several sets of RGB-D ground truth data via an error accumulation metric. We show that the proposed two-stage approach significantly reduces error in the pose estimation, compared to a state-of-the-art ICP registration technique.**

## I. INTRODUCTION

The 6D pose estimation problem we address is with respect to an RGB-D camera observing a static scene in a fixed world coordinate frame. Our approach is incremental and works by comparing the current RGB-D scan to the previous RGB-D scan to obtain the relative change in pose. Estimating the camera pose is useful for many robotics tasks such as mapping or control. The focus of our technique is to achieve the best possible accuracy, while maintaining real-time performance.

We first discuss the data available from the RGB-D device. An *RGB-D scan* in the context of this paper is defined as a pair of images of equal size: a 2D RGB image $I_{RGB}$ and a 2D depth image $I_D$. The images are pre-registered, meaning that a pixel $[u, v]$ in the RGB image and the corresponding pixel in the depth image both refer to the same point $\mathbf{p} = [p_x, p_y, p_z]^T$, measured in camera coordinates. The *depth* of a point (value of the pixel in the depth image) is equal to $p_z$.

Given the depth image and camera parameters, one can calculate the 3D projection of each pixel to obtain a 3D point cloud. We assume that the correct color-to-depth image registration and 3D projection are handled by the device driver, and are outside the scope of this paper.

This work is supported in part by U.S. Army Research Office under grant No. W911NF-09-1-0565, and U.S. National Science Foundation under grants No. CNS-0619577 and No. IIS-0644127

*Ivan Dryanovski is with the Dept. of Computer Science, The Graduate Center, The City University of New York (CUNY), 365 Fifth Avenue, New York, NY 10016 (corresponding author, e-mail: idryanovski@gc.cuny.edu

Carlos Jaramillo is with the Dept. of Computer Science, The Graduate Center, The City University of New York (CUNY), 365 Fifth Avenue, New York, NY 10016 (e-mail: cjaramillo@gc.cuny.edu

Jizhong Xiao is with the Electrical Engineering Department, City College of New York, Convent Ave & 140th Street, New York, NY 10031 (e-mail: jxiao@ccny.cuny.edu

We note that we can treat an RGB-D scan in two different ways: as a 3D point cloud, where each point has additional color information, or as two 2D images. We can go back and forth between the two representations at any time. This redundancy is useful because we can exploit the structure of the data based on our needsa and enables us to leverage algorithms from both image processing libraries such as OpenCV [1] and geometric processing libraries such as Point Cloud Library (PCL) in a ROS framework [2].

Given two successive point clouds of the same static scene observed from different viewpoints, one can find the change in pose of the camera by obtaining the rigid transformation that best maps one point cloud onto the other. A common technique to do that is the Iterative Closest Point (ICP) algorithm [3]. Since a typical RGB-D scan can have hundreds of thousands of points, performing ICP on the full point cloud is a computationally expensive procedure. A common way to alleviate this problem is to down-sample the data in order to speed up computation, at the cost of precision. This describes a fundamental trade-off in the performance of ICP: Registering using dense point clouds yields better alignment, but is done with lower frequency. Registering down-sampled point cloud results in lower incremental accuracy, but we can process a greater number of scans. If we process scans with a low frequency, the translation and rotation of the camera between two processed frames can be too large, leading to a convergence failure in the algorithm. Thus, for the best ICP performance, a careful balance between data size and frequency has to be achieved.

In this paper, we describe a two-stage registration approach which attempts to address the trade-off between speed and accuracy. The first stage of the algorithm involves finding pose-invariant edge features in the RGB-D scan. For this, we use Canny edge detection operating on the intensity of the RGB image. The detected edges from the color image are correlated to depth pixels in the depth image using a custom filtering technique, in order to produce pose-invariant 3D edges. Once the 3D features are extracted, they can be incrementally registered using ICP at a high frequency. The second stage involves performing ICP on the full point cloud data. For this, we use a G-ICP [4], a state-of-the-art ICP variation which uses a point-to-plane metric. The second stage uses the accumulated results of the first stage as the initial guess for the point cloud alignment.

We show that the two-stage procedure results in alignment with significantly less incremental error when compared to single-stage G-ICP registration. Another significant advantage of our approach is that the edge features detected from the RGB data can be used to register scans where G-ICP alone would fail due to geometrically ambiguous scenes - for example - going down a hallway that has flat walls and no geometric landmarks.

The drawback of the two-stage approach is that since the two loops are ran concurrently, and thus two threads are required. Furthermore, any incremental technique introduces error that builds up over time. Thus, an additional loop closure step needs to be performed. This can be done with a variety of methods, including graph optimization algorithms such as TORO [5][6] or g2o [7]. Since we only concern ourselves with improving incremental registration, those topics are beyond the scope of this paper.

This paper is organized as follows: We introduce related state-of-the-art work in II. We discuss the edge feature detection procedure in III. Section IV explains the two-stage incremental registration approach. We present experimental results obtained by using the pose estimation system on ground truth RGB-D datasets in V. Finally, Section VI summarizes our achievements.

## II. RELATED WORK

6D pose estimation of robots via scan registration and SLAM techniques for robots equipped with either range sensors [8], [9] or perspective camera [10], [11], [12] has been studied widely. Some approaches utilized data from both 3D range scanners and perspective camera to estimate the pose of the robots [13], [14], [15]. The current generation state-of-the-art sensors instantaneously acquire both color images (RGB) and depth of the surrounding environment. The most popular sensors are the Kinect sensor and SwissRanger SR-3000. The Kinect camera acquires range and color images at VGA resolution and the depth accuracy falls as the range distance measured. SwissRanger cameras have a good depth accuracy at its full range of 12-15 meters but with a smaller field-of-view, although the color images acquired are not truly RGB data. Most recent efforts to build dense point cloud maps that integrate both range and color data in the map include [16], [17]. The 6D pose of the robot is estimated in the map building process. The most popular approach to estimate pose is via the scan registration [3]. Other registration techniques that are related to the ICP algorithm are [18], [19].

Edge extraction from the data points is a vital step and minimizes the number of data points to be considered for registration. Most recent edge extraction techniques from range images include [20], [21], [22], [23]. In [12] edge features are extracted from stereo images to achieve 3D registration in indoor environment in real-time.

## III. EDGE FEATURE DETECTION

We denote the RGB image as $I_{RGB}$ and the depth image as $I_D$, and assume that they are registered, i.e., the RGB pixel $I_{RGB}(u, v)$ and the depth pixel $I_D(u, v)$ correspond to the same world point. Note that while the RGB image is complete, parts of the depth image will be empty. This will occur any time the device is not able to determine the distance to the given point. In this paper, we will refer to such gaps in the depth image as *shadow* pixels. We use **NaN** (not-a-number) notation for shadow pixels in the depth.

The goal of the feature detection step is to obtain the 3D coordinates of features in each RGB-D frame. We are interested in finding *pose-invariant* features, whose coordinates in the world frame do not change as the pose of the camera changes. We note that simply down-sampling the cloud uniformly to a sparse set of points will not produce such pose-invariant features, and hence will not be as useful for the coarse aligning step.

### A. Edge detection

We begin by computing an intensity image $I_I$ from the RGB image:

$$I_I = \frac{1}{3}(I_{RGB}[R] + I_{RGB}[G] + I_{RGB}[B]) \tag{1}$$

Next, we smooth the intensity image using a Gaussian blur operation, and apply a Canny detection algorithm to find edges. We denote the resulting image $I_C$, where $I_C(u, v) = 1$ if the pixel belongs to an edge, and $0$ otherwise.

Next, we have to assign a depth value to each edge pixel in $I_C$, in order to compute the 3D coordinates of the edge. Directly looking up the value of the pixel in the depth image creates problems for two reasons. First, the corresponding depth pixel might be a shadow pixel. Second, there could be an ambiguity about where the edge is actually located. If we detect a color edge which is somewhere on a flat surface (for example, a black poster on a white wall), then all is well - either the black pixel of the poster edge or the white pixel of the wall edge can serve as a pose-invariant feature. However, consider the case where we have a dark object in the foreground, against a light background. An edge which appears associated with the foreground will be pose-invariant, while an edge appearing on the background will be pose-variant. This problem is further exacerbated by the fact that the RGB-to-depth registration of the sensor might not be perfect, leading to the effect that the colors of objects in the foreground may "bleed" into the background by several pixels.

To resolve these problems, we filter the edges in the following way. For every edge pixel in $q = I_C(u, v)$, we perform a local search in $I_D$, centered at $(u, v)$. The depth value of $d(q)$ of the pixel is the smallest depth we find in that window of size $2w + 1$:

$$d(q) = min_{u',v'} I_D(u', v') \tag{2}$$

where

$$u' \in [u - \frac{1}{2}w, u + \frac{1}{2}w]$$

$$v' \in [v - \frac{1}{2}w, v + \frac{1}{2}w]$$

The results of the edge detection with a value of $w = 2$ and filtering techniques can be seen in Fig. 1.

Finally, we note that this detection procedure obtains pose-invariant edges most of the time, but is not guaranteed to always do so. Consider the case when the edge detected

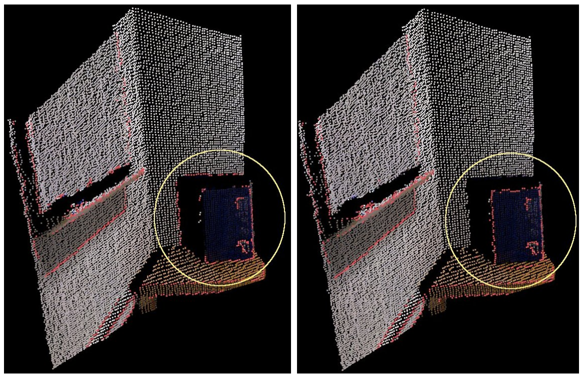


Fig. 1. Left: erroneous edge association - the edge is associated with the background, instead of the box (highlighted in circle). Right - correct edge detection after performing local search for foreground pixels.

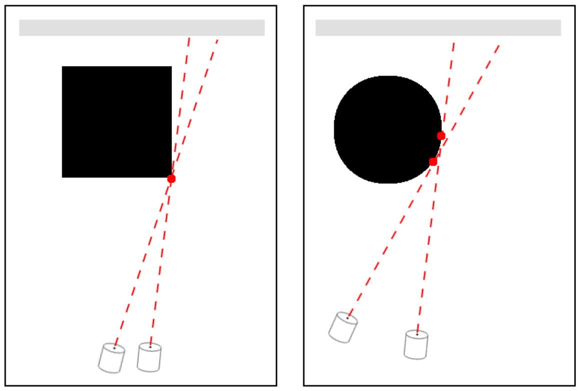


Fig. 2. A camera observing a dark object against a lighter background. Left: sharp-edged objects result pose-invariant edge features. Right: Curved-edge objects can create pose-variant edge features (edge appears at different position as the camera pose changes).

is between an object in the foreground and an object in the background, and the object in the foreground has a curved surface. As the camera viewpoint changes, the edge between the foreground and background will appear at different coordinates in the world frame (Fig. 2). However, in the presence of enough pose-invariant edges, the edge registration procedure is able to deal with pose-variant edge by treating them as outliers.

### *B. Computation of edge descriptors*

After detecting the edge features, we perform a description step. Each point which belongs to an edge is described by the angle of the intensity gradient vector at that pixel. First, we compute the $x-$ and $y-$ gradient images $I_{Sx}$ and $I_{Sy}$ from the intensity image $I_I$ by applying the Sobel operator. Next we calculate the gradient vector orientation $\theta$ for a given edge pixel $q(u, v)$, given by:

$$\theta = \arctan \frac{I_{Sy}(u, v)}{I_{Sx}(u, v)} \tag{4}$$

Note that we would like to preserve the angle in the continuous range of $[0, 2\pi]$.

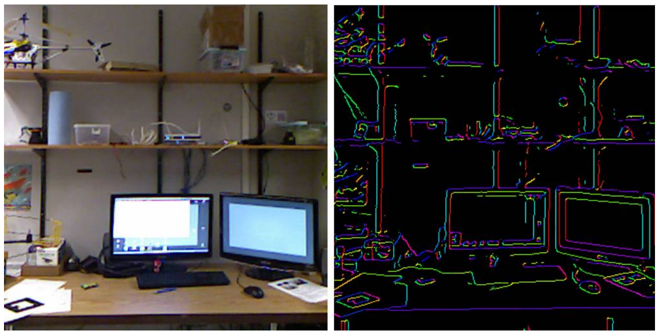


Fig. 3. Left: RGB image. Right: result of the edge detection algorithm. The hue of the color corresponds to the orientation of the gradient.

Fig. 3 shows the result of the edge descriptors computation. In the right image, hue is used for visualization only as it corresponds to the orientation of the gradient. For example, the vertical edges appear as either red or cyan, where red occurs when there is a transition from dark to light, and the cyan occurs when there is a transition from light to dark.

Since the gradient vector is computed in image-space, the edge of an object in the scene will appear to have different gradient orientations when viewed from different positions. However, for small changes of the camera pose, the same static edge will have a similar $\theta$ value. Thus, we can expect the same edge to have the same $\theta$ between two successive RGB-D frames to within some tolerance parameter. We will use this angular descriptor when determining correspondences between edges in the registration step.

## IV. INCREMENTAL REGISTRATION

We employ a two-stage incremental registration procedure. The first stage (high-frequency loop) aligns the 3D points detected as edges. The second stage (low-frequency loop) aligns the dense point cloud data, by using the estimation from the high-frequency loop as an initial guess. The two loops are executed in parallel.

In the following section, we describe the details of each stage.

### *A. High-frequency loop (edge-based ICP)*

We developed a variation of the ICP algorithm, which we refer to as Edge-ICP, to perform alignment of the edge features. The classic ICP algorithm aligns a point cloud $D$ (data) to a prior point cloud $M$ (model), where any point $P$ is defined as $P = [P_x, P_y, P_z]^T$. The distance function for computing the nearest neighbor is usually the Euclidean distance, defined for two points $P$ and $Q$ as

$$d(P, Q) = \sqrt{(P_x - Q_x)^2 + (P_y - Q_y)^2 + (P_z - Q_z)^2}$$

The nearest neighbor search can be optimized using a kd-tree structure [24].

For the Edge-ICP algorithm, we define each point $P$ in the point cloud as

$$P = [P_x, P_y, P_z, P_\theta]$$

where $P_\theta$ is the gradient angle of the corresponding edge pixel. We define a new metric $d'$ for computing nearest neighbors:

$$d'(P,Q) = \begin{cases} d(P,Q) & \text{if} \quad |P_\theta - Q_\theta| < t_a \\ \infty & \text{if} \quad |P_\theta - Q_\theta| \geq t_a \end{cases}$$

Informally, the distance between two edge features is the same as the Euclidean distance if their gradient angles are within a certain threshold, and infinity otherwise. Thus, when looking for the nearest neighbor to an edge feature point P, only edges with similar gradient angles are considered as possible candidates. The magnitude of the angular threshold $t_a$ is not of great importance; a value of $30°$ or $45°$ is sufficient to prune out edges which are pointing in completely different directions.

Attempting to apply this metric to the classic ICP algorithm results in a problem if we want to use kd-tree optimization for the nearest neighbor step. kd-tree structures organize data in Euclidean space, while the distance metric we defined is non-Euclidean. To resolve that problem, we propose a modified ICP algorithm, presented in Algorithm 1.

The difference between the two algorithms is the way in which correspondences are found. Instead of querying for a single nearest neighbor, we perform a search for a fixed number of neighbors (a typical value would be 10 or 20). Assuming that the set of nearest neighbors $N$ arrive sorted by their euclidean distance to the data point $D^i$, we begin iterating over them, starting with the closest candidate. We accept the first candidate neighbor which obeys the gradient angle tolerance. If we encounter a neighbor which is further than a certain maximum distance $t_d$ from the query point, we give up the search for the point $D^i$.

### B. Low-frequency loop (dense ICP)

The function of the low-frequency loop is to refine the estimation produced by the high-frequency loop. In order to do this, we perform ICP on consecutive dense RGB-D point clouds. We uniformly down-sample the incoming point clouds using a voxel grid filter with a resolution of 2.5 cm.

**Algorithm 1** Edge-ICP (pointcloud $M$, pointcloud $D$)

```
1: while not converged do
2:   for each point $D^i$ in $D$ do
3:     $N$ = get_nearest_neighbors($D^i$, $M$)
4:     for each $N^j$ in $N$ do
5:       if $|D^i_\theta - N^j_\theta| < t_a$ then
6:         $Q^i = N^j$
7:         break
8:       else if $d(D^i, N^j) > t_d$ then
9:         $Q^i = \emptyset$
10:        break
11:      end if
12:    end for
13:  end for
14:  $T$ = compute_transformation($D$, $Q$)
15:  $D$ = transform_pointcloud($D$, $T$)
16: end while
```

Among the available variants of the ICP algorithm, we have chosen *generalized ICP*, or G-ICP [4]. Generalized ICP can be used to align scans using a point-to-plane metric, which improves convergence speed in structured environments.

A summary of the two stages is presented in Table I.

TABLE I
TWO-STAGE REGISTRATION DETAILS

| Algorithm | Loop Frequency | Data | Metric |
|---|---|---|---|
| Edge-ICP | high | sparse | point-to-point; non-euclidean gradient angle correspondences |
| G-ICP | low | dense | point-to-plane; Euclidean |

## V. EXPERIMENTS AND RESULTS

We compare the performance of the various algorithms using 9 experimental datasets, provided by [25]. The datasets include RGB-D scans at QQ-VGA resolution taken from a Kinect camera moving in different trajectories, and observing scenes with varying amounts of visual features. The datasets also provide a ground truth value for the 6-DoF pose of the camera, recorded by a motion capture system. The details of the nine Kinect sequences are summarized in Table II in the Appendix.

The metric used for benchmarking is based on the metric for visual SLAM algorithms proposed in [26]. The metric computes the accumulated error between the transformation estimated by the registration between times $t_i$ and $t_{i-1}$, versus the transformation reported for the same time interval by the ground truth data. This error is formally defined as

$$E = \sum_{i=1}^{n} \left( (\hat{x}_i \ominus \hat{x}_{i-1}) \ominus (x_i \ominus x_{i-1}) \right) \tag{5}$$

where the $\ominus$ operator on two transforms $A$ and $B$ is

$$A \ominus B \equiv B^{-1}A \tag{6}$$

The error $E$ is a transform which includes a rotational component $E_R$ and translational component $E_T$, where $E_R$ is a 3-by-3 rotation matrix, and $E_T$ is a 3-dimensional vector. We define the total positional error $e_p$ as the size of the translation error vector

$$e_p = ||E_T|| \tag{7}$$

We define the total angular error $e_r$ as the principal angle of the rotation matrix $E_R$, given by

$$e_r = |\cos^{-1}(0.5 \text{ tr}(E_R) - 1)| \tag{8}$$

The average positional and rotational error are obtained by dividing the total error by the total run time.

We have performed 3 sets of experiments. The first set analyzes the performance of standalone G-ICP under various parameters. In this way, we attempt to find the *best* possible performance of standalone G-ICP registration.

The second set of experiments deals with Edge-ICP, and demonstrates the effects of the gradient orientation descriptor on the accuracy of the registration.

The final (and most informative) set of experiments compares the performance of standalone G-ICP v.s. our proposed two-stage approach, which incorporates Edge-ICP and G-ICP.

### *A. G-ICP parameter analysis*

There are many factors which affect G-ICP convergence speed and accuracy, and exploring them is beyond the scope of this paper. We focus on testing how the number of data points in the point clouds affects accuracy. We manipulate the number of points by uniformly down-sampling the point cloud using a voxel grid filter with a given resolution. For the rest of the G-ICP parameters, we modified the default values provided by the authors so the neighboring distance between points is 20 cm.

### *B. Edge-ICP parameter analysis*

We evaluate the usefulness of the gradient orientation descriptor for Edge-ICP by comparing the Edge-ICP under different parameter conditions (Fig. 4). The $x-$ axis is the number of nearest neighbors which Edge-ICP considers while looking for an acceptable correspondence based on gradient orientation, before giving up. The graphs show that using the edge gradient orientation while computing correspondences gives better results. The graphs show that examining 16 or 20 neighbors are usually enough in order to find the best correspondence.

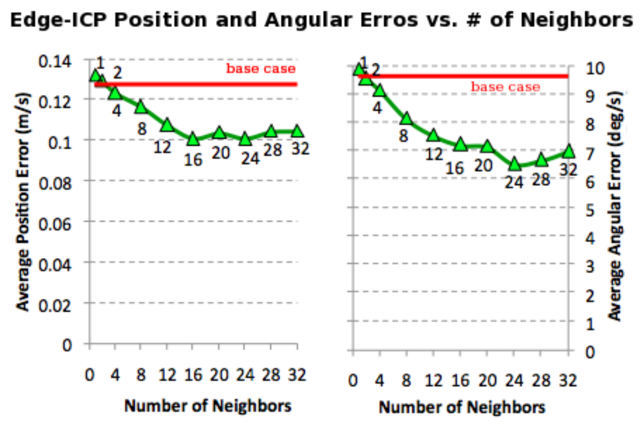


Fig. 4. Comparison of average positional and angular error using Edge-ICP. The base case (in red) represents refers to using edge features without considering gradient orientation correspondences. The $x-$ axis is the number of nearest neighbors which Edge-ICP considers while looking for an acceptable correspondence based on gradient orientation, before giving up.

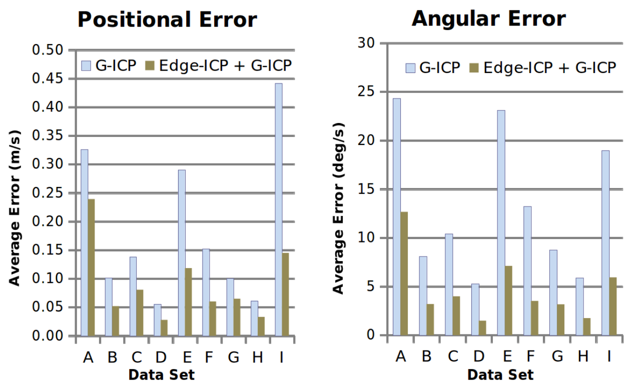


Fig. 6. Comparison of average positional and angular error using proposed two-stage approach (Edge-ICP + G-ICP) v.s best-case standalone G-ICP.

### *C. 2-stage v.s. standalone registration results*

We compare the performance of the two-stage (Edge-ICP and G-ICP) registration method to standalone G-ICP registration.

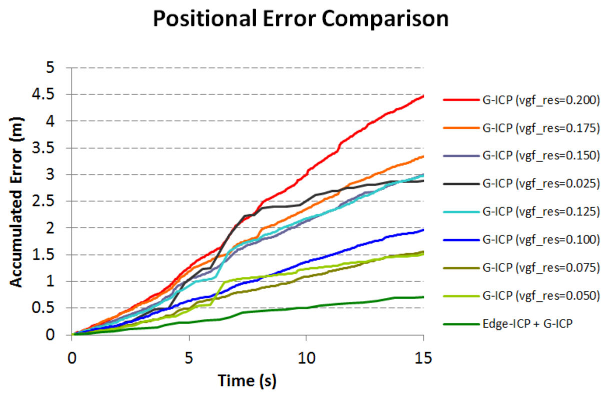
Fig. 5 shows the development of the total error over time for one of the sets of RGB-D data. We evaluate the performance of our two-stage method (Edge-ICP + G-ICP) v.s. a number of standalone G-ICP trials using different down-sampling parameters. In the graphs, vgf_res refers to the voxel grid filter resolution used (bigger values result in less points in the point cloud).

Fig. 6 shows the average positional and angular error for all the 9 datasets we tested. Note that the standalone G-ICP results refer to the best-case performance that G-ICP achieved, using all the explored parameter options. On the other hand, when evaluating our own algorithm (Edge-ICP + G-ICP), we used a fixed downsampling resolution parameter (2.5 cm) for all 9 datasets. Our suggested approach outperforms standalone generalized ICP in both positional and angular error for every single test set, sometimes by a factor of 3 or 4 times. On a Quad Dual-Core Intel CPU, the high-frequency loop of the pose estimation systems ran at 15-30Hz, while the low-frequency loop ran at 2-4 Hz.

## VI. CONCLUSIONS

We introduced a real-time pose estimation technique for RGB-D cameras. The system employs a two-stage registration algorithm. The inner stage runs at a high frequency and registers data using Edge-ICP and oriented gradient edge features. The second stage, which runs at a lower frequency, uses ICP with a point-to-plane metric to refine the estimate of the first loop. The two-stage approach performs significantly better than standalone ICP implementations when tested on real-world datasets with ground truth, reducing both the total positional and total angular error.

For our future work, we plan on testing the pose estimation system onboard micro-air vehicles, and evaluating its applicability to autonomous flight control and indoor 3D mapping and exploration.



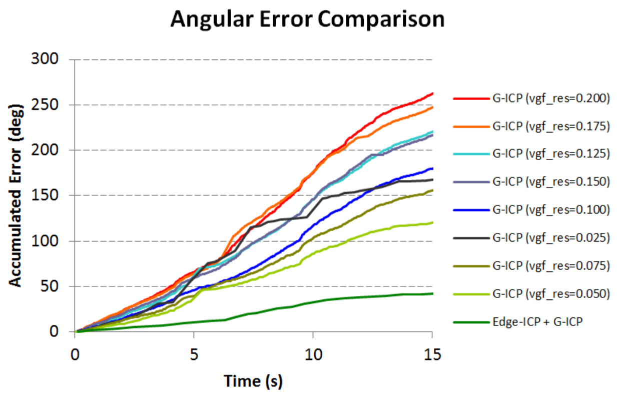


Fig. 5. Accumulated positional and angular error for data set B. Proposed two-stage approach (Edge-ICP + G-ICP) is compared against standalone G-ICP applied on data down-sampled at different voxel grid filter resolutions. The legend is ordered by the total error. Edge-ICP + G-ICP outperforms all G-ICP runs.

## References

[1] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[2] W. Garage, "ROS - Robot Open Source," http://www.willowgarage.com/pages/software/ros-platform, 2009.

[3] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on pattern analysis and machine intelligence*, pp. 239–256, 1992.

[4] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Proc. of Robotics: Science and Systems (RSS)*. Citeseer, 2009.

[5] G. Grisetti, C. Stachniss, G. Slawomir, and B. Wolfram, "TORO project at OpenSLAM.org," http://openslam.org/toro.html, 2007.

[6] G. Grisetti, C. Stachniss, and B. Wolfram, "Non-linear Constraint Network Optimization for Efficient Map Learning," in *IEEE Transactions on Intelligent Transportation Systems*, 2009, pp. 428–439.

[7] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, Shanghai, may 2011, pp. 3607–3613.

[8] H. Surmann, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45, no. 3-4, pp. 181–198, Dec. 2003.

[9] D. Borrmann, J. Elseberg, K. Lingemann, a. Nuchter, and J. Hertzberg, "Globally consistent 3D mapping with scan matching," *Robotics and Autonomous Systems*, vol. 56, no. 2, pp. 130–142, Feb. 2008.

[10] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: real-time single camera SLAM." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–67, June 2007.

[11] L. Paz, P. Piniés, J. Tardós, and J. Neira, "Large Scale 6-DOF SLAM with Stereo-in-Hand," *Robotics, IEEE Transactions on*, vol. 24, no. 5, pp. 946–957, 2008.

[12] M. Tomono, "Robust 3D SLAM with a stereo camera based on an edge-point ICP algorithm," in *IEEE International Conference on Robotics and Automation, ICRA '09*. IEEE, 2009, pp. 4306–4311.

[13] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using visual appearance and laser ranging," in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 1180–1187.

[14] R. Kaushik, J. Xiao, S. Joseph, and W. Morris, "Polygon-based laser scan registration by heterogeneous robots," in *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1618–1623.

[15] B. Huhle, M. Magnusson, W. Straß er, and A. J. Lilienthal, "Registration of colored 3D point clouds with a kernel-based extension to the Normal Distributions Transform," in *IEEE International Conference on Robotics and Automation, ICRA '08*, 2008, pp. 4025–4030.

[16] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D Mapping: Using depth cameras for dense 3D modeling of indoor environments," in *the 12th International Symposium on Experimental Robotics (ISER)*, 2010.

[17] R. B. Rusu and N. Blodow, "Fast point feature histograms (FPFH) for 3D registration," in *IEEE International Conference on Robotics and Automation, ICRA '09*, 2009, pp. 3212–3217.

[18] M. Magnusson, H. Andreasson, A. Nuchter, and A. Lilienthal, "Appearance-based loop detection from 3D laser data using the normal distributions transform," in *IEEE International Conference on Robotics and Automation, ICRA '09*, no. 2, 2009, pp. 23–28.

[19] A. Diosi and L. Kleeman, "Fast laser scan matching using polar coordinates," *The International Journal of Robotics Research*, vol. 26, no. 10, p. 1125, 2007.

[20] S. Coleman, B. Scotney, and S. Suganthan, "Edge detecting for range data using laplacian operators," *Image Processing, IEEE Transactions on*, vol. 19, no. 11, pp. 2814–2824, 2010.

[21] C. Ye, "Robust edge extraction for SwissRanger SR-3000 range images," in *IEEE International Conference on Robotics and Automation, ICRA '09*, 2009, pp. 2437–2442.

[22] B. Steder, R. Rusu, and K. Konolige, "Point Feature Extraction on 3D Range Scans Taking into Account Object Boundaries," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2010, pp. 2601–2608.

[23] R. B. Rusu, S. Cousins, and W. Garage, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA), 2011*, 2011, pp. 1–4.

[24] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang, "Fast Approximate Nearest-Neighbor Search with k-Nearest Neighbor Graph," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011, pp. 1312–1317.

[25] F. Pomerleau and S. Magnenat, "Kinect with ground truth from Vicon," http://www.asl.ethz.ch/research/datasets, 2011.

[26] F. Colas, D. Cremers, N. Engelhard, and R. Siegwart, "Towards a benchmark for RGB-D SLAM evaluation," *RSS 2011 Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, pp. 1–2, 2011.

## Appendix

TABLE II

KINECT SEQUENCES USED IN BENCHMARK

| Sequence | File name | Clutter | Speed | Motion Type |
|---|---|---|---|---|
| A | 0high-0medium-0fly.bag | high | medium | flying |
| B | 0high-0slow-0fly.bag | high | slow | flying |
| C | 0high-0medium-0tr.bag | high | medium | translation |
| D | 0high-0slow-0tr.bag | high | slow | translation |
| E | 0medium-0medium-0fly.bag | medium | medium | flying |
| F | 0medium-0slow-0fly.bag | medium | slow | flying |
| G | 0medium-0medium-0tr.bag | medium | medium | translation |
| H | 0medium-0slow-0tr.bag | medium | slow | translation |
| I | 0high-0slow-0rot.bag | high | slow | rotation |

These data sets are publicly available from http://www.asl.ethz.ch/research/datasets