2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
September 14-18, 2014. Chicago, IL, USA,

Experimental Study of Odometry Estimation Methods using RGB-D Cameras

Zheng Fang[1] and Sebastian Scherer[2]

Abstract— **Lightweight RGB-D cameras that can provide rich 2D visual and 3D point cloud information are well suited to the motion estimation of indoor micro aerial vehicles (MAVs). In recent years, several RGB-D visual odometry methods which process data from the sensor in different ways have been proposed . However, it is unclear which methods are preferable for online odometry estimation on a computation-limited, fast moving MAV in practical indoor environments. This paper presents a detailed analysis and comparison of several state-of-the-art real-time odometry estimation methods in a variety of challenging scenarios, with a special emphasis on the trade-off among accuracy, robustness and computation speed. An experimental comparison is conducted using public available benchmark datasets and author-collected datasets including long corridors, illumination changing environments and fast motion scenarios. Experimental results present both quantitative and qualitative differences among these methods and provide some guidelines on choosing the "right" algorithm for an indoor MAV according to the quality of the RGB-D data and environment characteristics.**

I. INTRODUCTION

Micro Aerial Vehicles (MAVs) have numerous potential applications in military and civil fields, such as surveillance, monitoring, exploration and rescue. Stable and precise control of an autonomous MAV demands fast and accurate estimates of the vehicle's pose and velocity. Especially, MAVs operating in cluttered indoor environments need pose updates at high rates with little latency for position control. At the same time they are only capable to carry a limited payload of sensors and processors. The lightweight commodity RGB-D cameras that have become available in recent years are well suited for such application scenarios.

RGB-D cameras can provide rich 2D and 3D information that help solve the motion estimation problem. In the past few years, several visual odometry [1]–[4] estimation methods using RGB-D cameras have been proposed. Existing RGB-D visual odometry methods have shown promising results with high accuracy if well-registered depth information is provided. However, reliability is still an issue that prevents these methods from being used for on-board guidance of a fully autonomous MAV. The methods may fail in different types of challenging scenarios as they process the sensor data differently. Understanding accuracy changes/reductions of these methods with respect to different challenges is important, and helps us design a robust motion estimation method for guiding an autonomous MAV.

[1]Zheng Fang is with State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110189, China fangzheng@mail.neu.edu.cn

[2]Sebastian Scherer is with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213, USA basti@cmu.edu

This paper experimentally compares several existing *real-time* RGB-D visual odometry methods including libviso2 [3], fovis [2], DVO [4], FastICP [5], GICP [6] and 3D-NDT [7]. The reasons why these six methods are selected are: First, they represent six different ways of processing RGB-D data to estimate odometry; Second, they all can run in real-time on a single CPU, which means they can be potentially used on a computation-limited indoor MAV. The study of this paper focuses on motion estimation using data from RGB-D cameras only. Further comparison including inertial-aided methods is not yet considered. The performances of these methods are studied in a variety of challenging scenarios such as in long corridors, low-light environments and fast motion scenarios. These scenarios involve low-quality features or featureless moments, which the existing method may or may not be able to handle. Experimental comparison of the aforementioned methods shows that the performance of each odometry estimation method depends on the quality of RGB-D data as well as environment characteristics. Though some of these methods can get very good estimation in certain environments, none of them can perform very well in all environments. The experiment results provide some guidelines on how to use different visual odometry methods according to the quality of RGB-D data and environment characteristics.

The rest of this paper is organized as follows. In section II, we discuss the related work. Section III describes the selected representative odometry estimation methods. We validate the performance of each method by using real datasets in section IV and we conclude in section V.

II. RELATED WORK

In the past few years, several odometry estimation methods have been proposed by using visual data and/or depth data. These methods can be roughly divided into three categories according to what kind of data is used.

The first category is ***image based methods*** which usually depend on a lot of information from RGB images. There are three different kinds of algorithms that are widely used. *The first kind* is sparse visual feature based methods [1], [3]. These methods usually extract sparse visual features first and then find the feature correspondences by matching the descriptors of each feature. After that, they usually use RANSAC-based methods to reject the outliers. Then, the 3D information of each feature is calculated through triangulation. Finally, the transformation matrix is calculated by minimizing the re-projection errors. *The second kind* is still sparse feature based methods but combines depth

978-1-4799-6933-3/14/$31.00 ©2014 IEEE

information and visual information [2] [8]. These methods also detect and extract visual features using different kind of feature detectors (such as SIFT, FAST, SURF), and then find the feature correspondences. But they don't triangulate the features to get depth information, instead they use the depth information from the RGB-D's depth image directly. And finally, the transformation matrix is also calculated through minimizing the re-projection error. *The third kind* is called dense feature based methods [4], [9] which are quite different from sparse visual feature based methods because they use the whole image to estimate the transformation. These methods assume a world point p observed by two cameras is assumed to yield the same brightness in both images. The goal is to find the camera motion that best satisfies the photo-consistency constraint over all pixels.

The second category is ***depth based methods*** which usually depend on much information from depth images. There are also several different kinds of methods. *The first kind* is 3D feature based methods [10], [11], which can be seen as an extension of the 2D cases. In these methods, 3D features like point, line or plane features will be detected and extracted. Then, the feature correspondences will be found using different kinds of matching methods. After that, the transformation matrix can be calculated by minimizing the distance error function of the feature correspondences. These methods usually are widely used in registration of 3D laser point cloud. However, it's worth noting that the point clouds from 3D laser scanners usually have less noise than RGB-D cameras and the long measurement range also helps them find more 3D features in a single scan. Unfortunately, commodity-level RGB-D cameras have very limited measurement range and their data are also very noisy compared to laser scanners. Therefore, usually 3D feature registration methods don't work very well for RGB-D cameras. Besides, most of them cannot run in real-time. *The second kind* is Normal Distribution Transform (NDT) based methods [12], [13]. These methods consider a point cloud as a distribution. They use the Normal Distribution Transform to map a point cloud to a smooth surface representation, described as a set of local probability density functions, each of which describes the shape of a section of the surface. Then, they can find the relative transformation between two point clouds by finding the optimal point-to-distribution or distribution-to-distribution matching. NDT is a very good solution for 3D point cloud registration, however it is sensitive to segmentation because large spatial cells filter out relevant details, whereas small cells augment the computational cost. Besides, it is also sensitive to initial guess and not very fast. *The third kind* is using the point cloud directly [5]. These methods usually use ICP based methods which are widely used in 3D registration area. The advantage of these methods is that they can get very accurate estimation results using dense point cloud. The disadvantage is that it is time consuming. But in recent years, researchers begin to propose fast ICP based registration methods.

The third category is based on ***both image and depth data*** [14]–[19]. For example, the Adaptive Iterative Closest Keypoint algorithm exploits both point position and visual appearance information [15]. It calculates the initial registration based on appearance and refines the final registration using 3D points. Leishman et al. [16] proposes a switching strategy which can switch to 2D laser odometry, 3D visual odometry and 2D visual odometry automatically according to the quality of RGB and depth images. Whelan et al. [17] integrates fovis and dense visual odometry with ICP for dense RGB-D mapping on a powerful CPU and GPU. Dryanovski [18] proposes a two-stage incremental registration algorithm for RGB-D images. In the first stage, edge features are detected from color and depth images, then these edge points are fed into ICP to calculate the initial motion estimation. In the second stage, the initial guess is refined by applying the GICP algorithm on the frame-to-frame dense point cloud. Herry [19] also proposes a two-stage RGB-D ICP using sparse features and ICP. Andreasson [7] proposes to use 2D image features and NDT to get a more robust and faster estimation. The key idea is to detect 2D visual features and find corresponding regions from previous frame to current frame and then use RANSAC to find a consistent alignment. After that, the geometrical information around the selected features is utilized as an input to a fast 3D-NDT distribution-to-distribution method to refine the transformation.

III. Examined Odometry Estimation Methods

As described in Section II, existing visual odometry methods can be divided into three categories according to what kind of sensor data is used. Here, we select six representative *real-time* odometry estimation methods and compare them by using the real datasets. Note that there are also some excellent methods [11], [17] that can estimate the camera motion accurately, but we don't select them because either they are too slow or they depend on a powerful GPU, which are unable to be applied on MAVs. The selected methods are shown in Table I. In the following, the selected methods are briefly described.

A. Image Based Methods

Many odometry estimation methods proposed in the past several years are based on visual information. Here, we select 3 representative methods since they use the image information in different ways. RGB-D cameras like the Kinect are monocular vision systems. If one wants to use them to estimate the motion by using image data only, it is necessary to know external information to solve the unknown scale problem. Actually, the selected methods here, they all use depth information. But since they depend much on image information, we call them image based methods.

1) Libviso2: Libviso2 [3] is a fast algorithm for computing the 6 DOF motion of a moving mono/stereo camera. For the stereo version of libviso2, the key idea is to extract robust sparse features, find the feature correspondences, get the depth information through triangulation and finally minimize the reprojection error of sparse feature matches to get the transformation. For the monocular version, it assumes that the camera is moving at a known and fixed height over

TABLE I

SELECTED METHODS AND THEIR COMPUTATIONAL PERFORMANCE ON THE TEST2 DATASET

Category	Method	Data type	Features	Algorithm Runtime(ms) Mean	Min	Max	StdDev	Avg CPU Usage
Image Based	Libviso2	RGB+Depth	2D Visual Features	39.5	18.9	180.2	18.9	29.8%
	Fovis		2D Visual Features & Depth	20.3	10.8	47.9	4.5	13.5%
	DVO		2D Image Intensity & Depth	52.4	20.1	242.8	11.2	22.6%
Depth Based	FastICP	Point Cloud	3D Point Cloud	50.3	13.3	350.0	34.0	26.1%
	GICP		3D Geometric Features & Cloud	87.8	16.7	281.4	20.8	35.5%
Image & Depth	NDT	RGB+Point Cloud	2D Visual Features & Cloud	29.9	11.5	133.6	11.8	21.9%

ground and uses this constraint to estimate the unknown scale. In order to use libviso2 with RGB-D cameras, we actually change the RGB-D camera into a virtual stereo camera using the depth information. We create a virtual camera using the depth information of each pixel with a fixed baseline as the Xtion RGB-D camera. If the RGB pixels don't have depth information, we just discard them. Then, we feed both RGB images of real camera and virtual camera into libviso2 to calculate the odometry.

2) Fovis: Fovis [2] is a visual odometry method that estimates the 3D motion of a camera using a source of depth information for each pixel. It first detects FAST features in each image. Then, the depth corresponding to each feature is extracted from the depth image. Features that do not have an associated depth are discarded. After that, each feature is assigned an 80-byte descriptor. Features are then matched across frames by comparing their feature descriptor values using a mutual-consistency check. And then, the inliers are detected by computing a graph of consistent feature matches and using a greedy algorithm to approximate the maximal clique in the graph. Finally, the motion estimate is computed from the matched features in three steps. Fovis uses a keyframe technique to reduce short-scale drift.

3) DVO: In contrast to sparse feature based methods, dense visual odometry [4] methods want to fully exploit both the intensity and the depth information provided by RGB-D sensors. Dense visual odometry uses all color information of the two images and the depth information of the first image. This approach is based on the photo-consistency assumption, which means a world point p observed by two cameras is assumed to yield the same brightness in both images.

$$I_1(X) = I_2(\tau(\xi, X)) \tag{1}$$

where $\tau(\xi, X)$ is the warping function that maps a pixel coordinate $X \in \Re^2$ from the first image to a coordinate in the second image given the camera motion $\xi \in \Re^6$. The goal is to find the camera motion ξ that best satisfies the photo-consistency constraint over all pixels.

B. Depth Based Methods

There are also many people working on motion estimation using only depth data. Actually, this is widely studied in computer graphics area where it is necessary to register different views of scans to reconstruct an object or environment. Many excellent registration algorithms have been proposed in the past decades, such as 3D point based method [20], plane based method [11] and 3D Normal Distribution Transform [12]. Most of these registration algorithms are very accurate, but usually are slow and computationally expensive. Therefore, most of them can not be used on computation-limited MAVs. Here, we select two methods which can be potentially used on MAVs. One is FastICP method which mainly uses only the point cloud for motion estimation. The other one is based on GICP [6], which can be seen as a method using 3D features of the point cloud for motion estimation.

1) FastICP: The Iterative Closet Points (ICP) algorithm was proposed by Besl and McKay in 1992 [21] to solve the 3D rigid shape registration problem. The classical pair-wise rigid registration problem can be described as: given a set of source points $X = \{x_i, i = 1, ..., n\}$ and $Y = \{y_i, i = 1, ..., n\}$, we want to find the optimal transformation by minimizing the energy function as:

$$\underset{\mathbf{R,t,Y}}{\operatorname{argmin}} \sum_{i=1}^{N_p} \varphi(\mathbf{R}x_i + \mathbf{t}, y_i) \tag{2}$$

where $\mathbf{R}$ is a rotation matrix, $\mathbf{t}$ is a translation vector, φ is the error metric function.

The key concept of the standard ICP algorithm can be summarized in two steps: 1) the alignment is fixed, a set of closest corresponding points Y is computed. 2) the 3D point correspondences are fixed, compute a transformation which minimizes distance between corresponding points. Iteratively repeating these two steps typically results in convergence to the desired transformation. According to [22], the following six steps influence the performance of ICP:

1. *Select* a subset of points in one or both point clouds.
2. *Match* these points to samples in the other point cloud.
3. *Weigh* the corresponding pairs appropriately.
4. *Reject* certain pairs by some strategies.
5. *Assign* an error metric.
6. *Minimize* the error metric.

In this paper, ethzasl-icp-mapping [5] developed by Pomerleau is chosen for comparison because it mainly focuses on robotic applications. This fast ICP follows the standard ICP pipeline as described in [22].

2) Generalized ICP: 3D features in point cloud such as corners or planes are very useful for registration. Several registration methods based on 3D point features or planes have been proposed in the past few years [10], [11], [20].

However, most of these methods are too slow to be used on a computation-limited MAV. Here, we implement a simple 3D feature based odometry estimation method which uses the local surface normals and plane features.

The method is based on Generalized ICP (GICP) [6]. Generalized ICP combines the iterative closest point algorithm and "point-to-plane" metric into a single probabilistic framework. Generalized-ICP is based on attaching a probabilistic model to the minimization step of standard ICP. In order to improve performance and increase the symmetry of the model, the generalized-ICP algorithm models locally planar surface features from both scans instead of just the "model" scan. Therefore, it can be thought as "plane-to-plane" method.

Assuming that the closest point correspondences have been found in two point clouds, $A = \{a_i\}_{i=1,\ldots,N}$ and $B = \{b_i\}_{i=1,\ldots,N}$ are indexed according to their correspondences. In the probabilistic model, it is assumed that there exist an underlying set of points, $\hat{A} = \hat{a}_i$ and $\hat{B} = \hat{b}_i$, which generate A and B according to $a_i \sim N(\hat{a}_i, C_i^A)$ and $b_i \sim N(\hat{b}_i, C_i^B)$. Here, C_i^A and C_i^B are covariance matrices associated with the measured points. For an arbitrary rigid transformation $\mathbf{T}$, define $d_i^{(\mathbf{T})} = b_i - \mathbf{T}a_i$, the generalized ICP wants to optimize the following function:

$$\mathbf{T} = \underset{T}{\operatorname{argmin}} \sum_i (d_i^{(\mathbf{T})^T} (C_i^B + \mathbf{T} C_i^A \mathbf{T}^T)^{-1} d_i^{(\mathbf{T})}) \quad (3)$$

This method can be considered as a kind of method using 3D features for odometry estimation. In this method, salient 3D geometric features in current point cloud are firstly selected using the surface normals. Then, the transformation between the selected 3D points and the local point cloud map is calculated by using GICP algorithm which actually uses local planar features of both scans to minimize the error function. This method is implemented by using the ROS [23] and PCL [24] GICP library. The following two issues are considered in our implementation:

Selecting the points with the most constraints: When a MAV is in a long corridor, usually the RGB-D camera cannot detect the wall at the end of the corridor because the maximum measurement range of our depth camera is less than 7 meters. Therefore, small objects on the wall and the door frames are vital to determine the translation. If one uses all the points, usually the algorithm will fail to estimate the translation since the points on the wall dominate the estimation. In order to select the most constrained points, we use a normal space sampling method [22]. The idea is to choose points such that the distribution of normals among the selected points is as large as possible. By doing so, some small objects on the wall and door frame will be selected while only few points on the wall will be selected, which is important for estimating the translation.

Constructing a local map: Frame to frame methods are widely used in odometry estimation. However, it is not robust in practice. Another option is frame to key frames method. We can calculate the overlap ratio of two point clouds, and if the overlap is lower than a given threshold, then the key frame is updated. The relative transformation is always calculated between the current frame and key frames. This method is much better than a frame-to-frame method, but it is still not very good in our experiments. Here, we use a local map based method, where we construct a local point cloud map and calculate the relative transformation between the current frame and the local map. This method is more robust and accurate than the previous two methods.

C. Methods Based on Both Image and Depth

Odometry estimation methods based on both image and depth information have become very popular in recent years. The most commonly used idea is using 2D image features to get a initial guess and then use point registration methods (such as ICP, NDT) to refine the estimation. Also, there are some people try to combine image based error metric and depth based error metric into one integrated error metric, then optimize the integrated error function to find the optimal transform [17]. However, these methods depend on a powerful computer for complex optimization. Here, we choose a real-time local visual feature boosted NDT method [7] for comparison.

1) Real-time 3D NDT: The key idea is to detect 2D visual features and find corresponding regions from previous frame to current frame and then use RANSAC to find a consistent alignment. After that, the geometrical information around the selected features is efficiently utilized as an input to a fast 3D-NDT-D2D method [12] to refine the transformation. Instead of computing the 3D-NDT representation of the full depth images, this method only considers the immediate local neighbourhoods of each of the detected local visual features points. By doing so, the number of Gaussian components can be decreased substantially. Thus, the whole estimation process can be calculated in real-time.

IV. EXPERIMENTS AND ANALYSIS

In this part, we first compare the accuracy of each method by using the TUM RGB-D dataset [1] which has accurate ground truth for evaluation [25]. Then, in order to evaluate the robustness, we also record some datasets by carrying our MAV and simulating practical flight in very challenging environments, such as long clear corridors, structured and cluttered environments, dramatic illumination changing environment, etc. Then, we validate the performance of each algorithm on an Asus UX31E Ultrabook (Quad-core 1.7GHz CPU and 4GB Memory) running ROS fuerte and PCL1.7. The RGB-D sensor is Asus Xtion Pro Live which records the RGB-D image at 15Hz with 640×480 resolution.

It should be noted that all the methods evaluated in this paper are sensitive to choice of parameters that might influence the final accuracy and timing results. From our experiments, for a certain dataset, better result may be achieved for a method if parameters are carefully tuned according to that specific dataset. However, this same set of parameters may

[1] http://vision.in.tum.de/data/datasets/rgbd-dataset/

work undesirably for another dataset. Since we want to get a general comparison of these methods, here we choose the default parameters for each method.

A. Accuracy Comparison using Benchmark Dataset

In this section, we use TUM RGB-D datasets to test the estimation accuracy of each method. The dataset contains the color and depth images along with the ground-truth trajectory. The data is recorded at full frame rate (30 Hz) and sensor resolution (640×480). The ground-truth trajectory is obtained from a high-accuracy motion-capture system with eight high-speed tracking cameras (100 Hz). Here, we choose 2 datasets to evaluate the accuracy of each method. The experimental results are shown in Table II.

We use the relative pose error metric [25] to measure the drift of visual odometry system. We calculate the ratio of pixels in the image that contain a valid depth compared to the total number of pixels in the image to estimate the quality of the point cloud. We compute the average and standard deviation of the grayscale-pixel values of the RGB image as a measurement of the amount of light available in the image, which is a simple way to estimate the quality of an image.

The first dataset is freiburg2/desk. For this sequence, the RGB-D data is recorded in a typical office scene with two desks, a computer monitor, chairs, etc. The Kinect is moved around two tables so that the loop is closed. The average translational and angular velocity are $0.193m/s$ and $6.338deg/s$ respectively. The mean intensity changes from 73.1 to 153.8, and the standard deviation changes from 76.1 to 42.1. However, the depth coverage changes from 85.3% to 53.9%. Therefore, the RGB information is good while the depth information is not very good. The mean and median relative pose error of each method is shown in Table II. From the results, it is clear that if the RGB image is good, visual information based method can get more accurate estimation. Since the quality of the point cloud in this dataset is not very good, the accuracy of depth information based methods is lower than visual information based methods. The reason why visual information based methods are good is that they can accurately and robustly detect and match feature correspondences since the RGB information is very good and the motion is relatively slow in this dataset.

The second dataset is freiburg1 room. In this dataset, the sequence is filmed along a trajectory through a typical office. It starts with the four desks but continues around the wall of the room until the loop is closed. The depth coverage changes from 83.8% to 54.9%. The mean intensity changes from 169.5 to 77.8 and standard deviation changes from 101.6 to 41.8. The experiment results are also shown in Table II. In this dataset, there are some fast rotations. The average translational velocity is 0.334 m/s and the average angular velocity is 29.882 deg/s, which is much faster than freiburg2/desk dataset. Therefore, it is a little bit difficult for sparse feature based methods. The advantage of the visual feature based method is not that obvious, as you can see the mean translation and rotation error of depth based methods are similar to visual information based method.

B. Robustness Validation in Challenging Environments

In this part, we collect some datasets in different kinds of environments to test the robustness of each method. Since it is very hard to get ground truth in large indoor environments, the camera is started and finished at the same position. Therefore, we can use loop-closing error to evaluate the estimation performance of each method to some extent. We define the loop-closing error as the gap between the two ends of a trajectory output compared to the total length of the trajectory. The loop-closing errors of each method are shown in Table III. Note that in Fig. 1−Fig. 4, the trajectory is projected onto $x-y$ plane. Therefore, if the x, y errors in the figure are very small but the loop-closing error is very big, that means there is a big drift in z coordinate.

TABLE III
LOOP-CLOSING ERROR

Methods	Loop-closing error			
	Test1	Test2	Test3	Test4
	length:40m	length:73m	length:25m	length:8.5m
Libviso	3.77%	1.45%	failed	13.6%
Fovis	7.41%	**1.12%**	failed	7.79%
DVO	**2.00%**	7.26%	16.86%	6.72%
FastICP	6.91%	4.58%	7.03%	**3.16%**
GICP	2.88%	4.17%	**3.59%**	4.05%
NDT	5.83%	5.74%	failed	5.13%

Test 1: long corridor environment. Very clear long corridors usually pose big challenges for visual odometry methods. In this experiment, the mean intensity of images changes from 164.5 to 59.4 and the depth coverage ratio changes from 90.9% to 79.3%. It seems that both RGB and depth information are very good. However, in this long corridor, the floors and walls are very smooth. There are many places where only small objects on the wall or door frames can be used to estimate the translation. Therefore, this environment is very challenging for both visual and depth based method since there are only few visual features and geometry features. Fig. 1 shows the experimental results of different methods. As you can see, in this environment, DVO has the best result, while Fovis fails before the robot enters into the last corridor section. The reason why DVO can succeed is that it depends on the whole image other than sparse features which are difficult to detect in this environment. Fovis's failure is due to many repetitive textures around the corner of the last corridor section, as shown in the top right picture in Fig. 1, where Fovis fails to detect and track features when there is a quick turn around that corner. But, Libviso2 can still track the features here. It seems Libviso2's feature tracking method is more robust than Fovis. Depth based methods can achieve similar performance. GICP based method is better than FastICP which completely fails to estimate the translation in the last corridor section. The reason why GICP can succeed is that points with most constraints are used instead of the whole point cloud.

Test 2: structured and cluttered environment: The second experiment is in a more complex environment. In this environment, sometimes there are very long corridors, sometimes

TABLE II

MEAN AND STANDARD DEVIATION OF TRANSLATIONAL AND ROTATIONAL ERRORS OF EACH METHOD

Methods	fr2/desk				fr1/room			
	$\bar{x}(m)$	σ	$\bar{\theta}(deg)$	σ	$\bar{x}(m)$	σ	$\bar{\theta}(deg)$	σ
Libviso	0.036	0.031	1.516	0.735	0.063	0.051	2.431	1.432
Fovis	0.012	0.007	0.526	0.307	0.056	0.035	2.377	1.328
DVO	0.024	0.012	0.982	0.512	0.058	0.045	2.396	1.539
FastICP	0.022	0.022	0.942	0.618	0.066	0.103	3.012	2.704
GICP	0.056	0.045	2.464	1.948	0.062	0.032	3.134	1.498
NDT	0.024	0.015	1.228	0.735	0.065	0.032	2.851	1.412

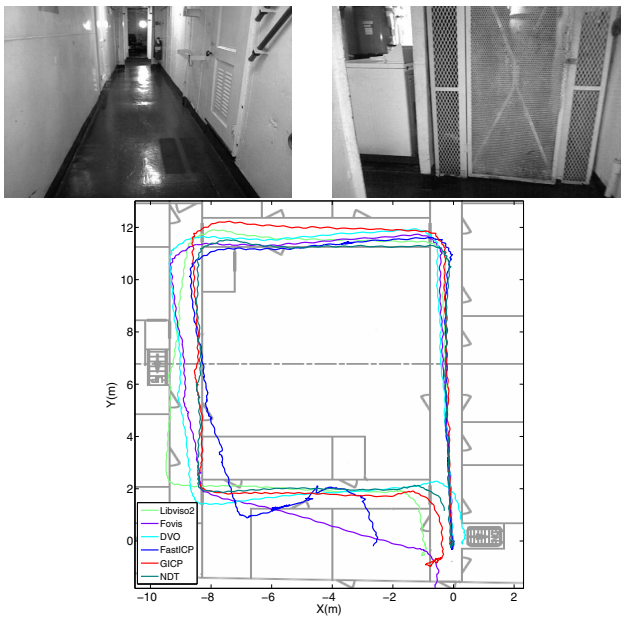

Fig. 1. Test 1: Long Corridor Environment

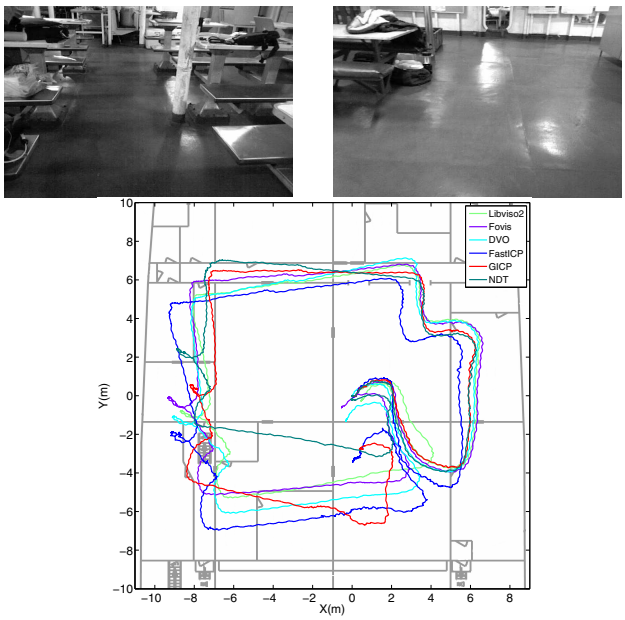

Fig. 2. Test 2: Complex Environment

there are cluttered rooms and sometimes there are very spacious rooms. We use this environment to check whether all the methods can adapt to the changes of environment structure. The mean intensity of images changes from 173 to 47 and the depth coverage ratio changes from 88.7% to 47.0%. As you can see, the quality of both image and depth changes much bigger than the first experiment. Fig. 2 shows the experiment results. From the experiment results, you can see that Fovis has the smallest loop-closing error. GICP works very well until there is a quick turn on the left side where all methods begin to become poor. The reason is that the quick turn happened at a place very close to a wall, where Xtion RGB-D camera can't get good RGB and depth data since the fast motion and the minimum measurement range of Xtion is 80cm. Therefore, it is very difficult for all the methods to estimate the transform accurately.

Test 3: Illumination changing environment. The third experiment is a conference room where there is a big desk and many chairs. In this experiment, dramatic illumination changes occur when the robot enters into and goes out of the conference room. The mean intensity changes from 169.2 to 3.5. And the conference room is very dark where the intensity is just about $3.5 \sim 30$, which is very challenging for visual based methods. Fortunately, we can still get a good point cloud in this environment, where the depth coverage ratio changes from 71.5% to 90.1%. The experiment result is shown in Fig. 3. It is no doubt that depth based methods will be better than visual based methods in this environment. In our experiment, we found that sparse visual feature based methods failed completely, while GICP based method achieves best performance. To our surprise, dense visual odometry can still work almost all the time except in very dark area (the mean intensity is less than 10). It seems that dvo is much more robust than sparse feature based methods in this test.

Test 4: Fast motion scenario. This dataset is recorded in a typical office environment where there are many tables, desks and computers. Therefore, this environment has lots of very good geometric and texture features for motion estimation. When recording the data, we keep rotating our robot with a relatively fast speed. The experimental result is shown in Fig. 4. As you can see from the RGB images,

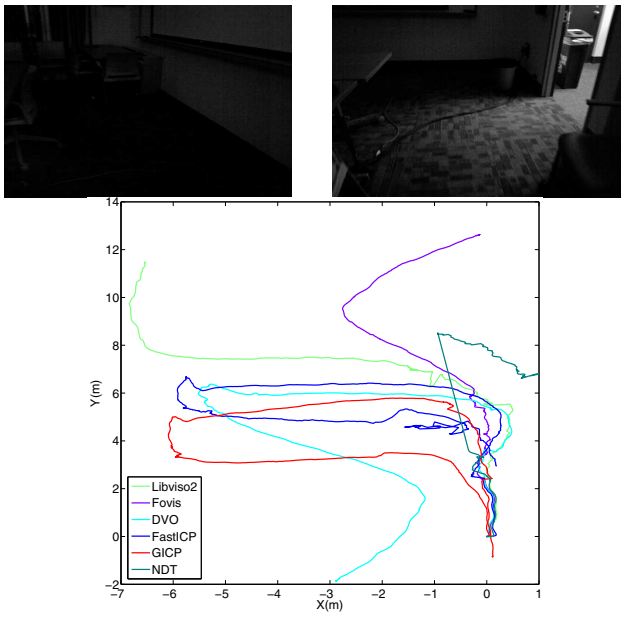

Fig. 3. Test3: Illumination changing Environment

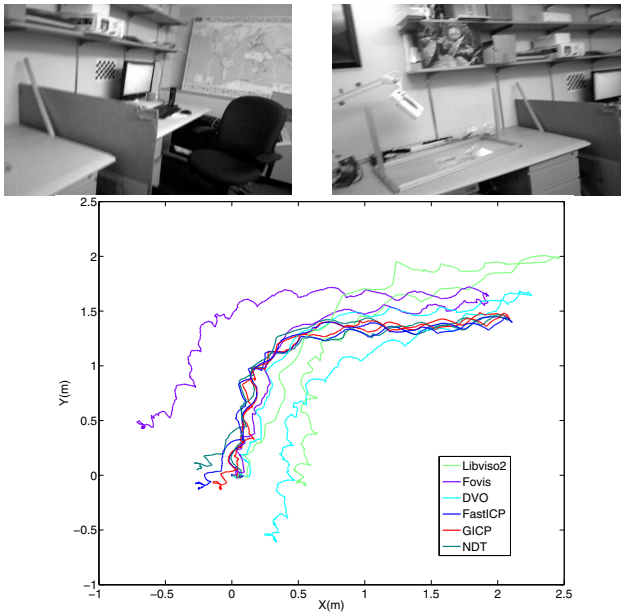

Fig. 4. Test3: Fast motion scenario

there is an obvious motion blur. The loop-closing error of each method is shown in Table III. From the results, we can see that for relative fast motion, depth based method are better than visual based method. The reason is that fast motion can cause motion blur which is very bad for feature detecting and matching while the point cloud is not too bad. Besides, NDT's performance against fast motion is also better than visual information based methods. The reason is that the selected NDT method is a two-stage method that combines image and depth information. Though the fast motion decreases the estimation accuracy of the first stage, the second stage can refine the estimation of the first stage by using NDT registration on dense point clouds.

C. Speed and CPU Usage Comparison

Since we are interested in comparing visual odometry methods that can be potentially used on indoor MAVs, the computational performance of each method is also very important. The faster the speed and the lower the CPU usage, the better for a MAV's on-board computing. We test the speed and CPU usage of each method on an Asus UX31E Ultrabook (Quad-core 1.7GHz CPU, 4GB Memory). The experimental results are shown in Table I. In our experiment, Fovis has the best speed and the lowest CPU usage, while depth based methods are much slower than image based methods. Though we haven't compared all these methods on our MAV onboard embedded computer (Quad-core 1.7GB CPU, 2GB memory) yet, the experiment results from running on a laptop computer indicate relative computation cost of the evaluated methods.

D. Analysis and Discussion

From the experiment results, it is clear that though some of the examined methods can get good results in specific environments, none of them can perform very well in all kinds of environments. They all have their own advantages and disadvantages.

For visual information based methods, the advantage is that they are faster and more accurate than point cloud based methods. But they have following disadvantages. First, the environments must have enough illumination. Second, the environments must have good texture features to be detected. Third, most of them discard the feature points that don't have depth information. Therefore, for spacious environments where depth information is not sufficient, they also cannot get good estimation.

For depth information based methods, the advantage is that they can be used in very dark environments. However, they also have some shortcomings. First, the effective measurement range of RGB-D camera is very limited. Therefore, there are often no enough points with constraints in spacious areas like atrium and long hallway. Second, the depth data of the consumer-level depth camera is very noisy. But we still need to downsample it to reduce the computation time. Therefore, the estimation accuracy of point cloud based method is not as accurate as visual information based methods in most cases.

For methods which use both image and depth information, usually they take advantages of both information. However, the method selected in this paper which is a two-stage coarse to fine estimation method. The initial guess is very important for the second stage. Therefore, if the first stage fails, usually the total estimation is not very good. A more robust way may be using a switch or joint optimization strategy to combine them together.

By analysing the characteristics of each method and the RGB-D data, we can get some ideas on how to use the RGB-D camera for robust and accurate odometry estimation. If the RGB and depth information are both available and good. We think we should use both kinds of information as much as possible for robustness consideration. If the RGB image

has abundant features but very few depth information, we should still consider how to use those features that don't have depth information. If the depth image has abundant geometric features but the RGB information is bad, we can depend on depth information based methods. If both RGB and depth information are unavailable, we can only use other sensor information for short time prediction, for example fusing visual odometry with IMU. Besides, for fast motion scenarios, IMU information can be considered for finding feature correspondences since IMU information is almost always available on MAVs.

V. CONCLUSIONS

In this paper, a detailed analysis and comparison of several visual odometry methods using RGB-D cameras has been presented. Representative approaches were compared on real data from public available datasets and author-collected datasets in several challenging environments. As the experimental results shown, the performance of each odometry estimation method depends on the quality of RGB-D data and the environment characteristics. The experiment results provide some guidelines on how to use different visual odometry methods according to the quality of RGB-D data and environment characteristics.

If the image has good grey value or visual features, we should consider image based methods first since they are faster and more accurate than depth based methods. But for featureless or dark environment, depth based methods are the best choices. More specifically, in environments with abundant texture features, if the image grey value is also good, fovis is the best choice for accuracy and speed. If the illumination is relatively dark, DVO is the best choice since it works much better than sparse feature based methods in bad illumination environment. If the illumination is very bad (Mean grey value is smaller than 10) or in featureless environments, then FastICP and GICP are the best choices. In environments with abundant geometric features, usually both FastICP and GICP can get good estimation. But, if there are only few geometric features in the environments, the accuracy of GICP developed in this paper is better than FastICP. The choice of the best algorithm depends on the quality of RGB-D data and the characteristics of the practical environment.

ACKNOWLEDGMENT

Research presented in this paper was funded by China Scholarship Council, NSFC under Grant No.61040014 and Fundamental Research Funds for the Central Universities under Grant No.N120408002. The authors would also like to thank J. Zhang, S. Nuske, S. Jean for their help.

REFERENCES

[1] D. Scaramuzza and F. Fraundorfer, "Visual Odometry [Tutorial]," *IEEE Robot. Autom. Mag.*, vol. 18, pp. 80–92, Dec. 2011.

[2] A. Huang and A. Bachrach, "Visual odometry and mapping for autonomous flight using an RGB-D camera," *Int. Symp. Robot. Res.*, pp. 1–16, 2011.

[3] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3d reconstruction in real-time," *2011 IEEE Intell. Veh. Symp.*, pp. 963–968, June 2011.

[4] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *2013 IEEE Int. Conf. Robot. Autom.*, pp. 3748–3754, IEEE, May 2013.

[5] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP variants on real-world data sets," *Auton. Robots*, vol. 34, pp. 133–148, Feb. 2013.

[6] A. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP.," *Robot. Sci. Syst.*, 2009.

[7] H. Andreasson and T. Stoyanov, "Real time registration of RGB-D data using local visual features and 3D-NDT registration," *Proc. Int. Conf. Robot. Autom. Work. Semant. Perception, Mapp. Explor.*, 2012.

[8] S. Quijada and E. Zalama, "Fast 6D Odometry Based on Visual Features and Depth," *12th Int. Conf. Intell. Auton. Syst.*, vol. 193, no. June, pp. 245–256, 2013.

[9] T. Tykkala, C. Audras, and A. I. Comport, "Direct Iterative Closest Point for real-time visual odometry," in *2011 IEEE Int. Conf. Comput. Vis. Work. (ICCV Work.*, pp. 2050–2056, IEEE, Nov. 2011.

[10] J. Xiao, B. Adler, and H. Zhang, "3D point cloud registration based on planar surfaces," in *2012 IEEE Int. Conf. Multisens. Fusion Integr. Intell. Syst.*, pp. 40–45, 2012.

[11] K. Pathak, A. Birk, N. VasÌĬkevicÌĬius, and J. Poppinga, "Fast Registration Based on Noisy Planes With Unknown Correspondences for 3-D Mapping," *IEEE Trans. Robot.*, vol. 26, pp. 424–441, June 2010.

[12] T. Stoyanov, M. Magnusson, H. Andreasson, and a. J. Lilienthal, "Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations," *Int. J. Rob. Res.*, vol. 31, pp. 1377–1393, Sept. 2012.

[13] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3D-NDT," *J. F. Robot.*, vol. 24, pp. 803–827, Oct. 2007.

[14] I. Dryanovski and R. G. Valenti, "Fast visual odometry and mapping from RGB-D data," in *2013 IEEE Int. Conf. Robot. Autom.*, pp. 2305–2310, IEEE, May 2013.

[15] J. Ekekrantz, A. Pronobis, J. Folkesson, and P. Jensfelt, "Adaptive iterative closest keypoint," in *2013 Eur. Conf. Mob. Robot.*, pp. 80–87, IEEE, Sept. 2013.

[16] R. C. Leishman, D. P. Koch, T. W. Mclain, and R. W. Beard, "Robust Visual Motion Estimation using RGB-D Cameras," pp. 1–13, 2013.

[17] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense RGB-D mapping," in *2013 IEEE Int. Conf. Robot. Autom.*, no. i, pp. 5724–5731, IEEE, May 2013.

[18] I. Dryanovski, C. Jaramillo, and J. Xiao, "Incremental registration of RGB-D images," in *2012 IEEE Int. Conf. Robot. Autom.*, pp. 1685–1690, IEEE, May 2012.

[19] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," *Int. J. Rob. Res.*, vol. 31, pp. 647–663, Feb. 2012.

[20] R. Rusu, N. Blodow, Z. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *2008 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 3384–3391, IEEE, Sept. 2008.

[21] P. Besl and H. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[22] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proc. Third Int. Conf. 3-D Digit. Imaging Model.*, pp. 145–152, IEEE Comput. Soc, 2001.

[23] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *ICRA Work. Open Source Softw.*, vol. 32, pp. 151–170, 2009.

[24] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE Int. Conf. Robot. Autom.*, pp. 1–4, IEEE, May 2011.

[25] J. Sturm, W. Burgard, and D. Cremers, "Evaluating Egomotion and Structure-from-Motion Approaches Using the TUM RGB-D Benchmark," *IEEE/RJS Int. Conf. Intell. Robot*, 2012.