Motion Segmentation based Robust RGB-D SLAM

Youbing Wang and Shoudong Huang
Faculty of Engineering and IT
University of Technology, Sydney
Sydney, Australia
youbing.wang@student.uts.edu.au,
shoudong.huang@uts.edu.au

***Abstract*—A sparse feature-based motion segmentation algorithm for RGB-D data is proposed which offers us a unified way to handle outliers and dynamic scenarios. Together with the pose-graph SLAM framework, they constitute an effective and robust solution that enable us to do RGB-D SLAM in wide range of situations, although traditionally they have been divided into different categories and treated separately using different kinds of methods. Through comparisons with RANSAC using simulated data and testing with different benchmark RGB-D datasets against the state-of-the-art method in RGB-D SLAM, we show that our solution is efficient and effective in handling general static and dynamic scenarios, some of which have not be achieved before.**

Keywords—robust SLAM; motion segmentation

I. INTRODUCTION

In simultaneous localization and mapping (SLAM) scenarios, there are three long standing problems that are pervasive and till now they are regarded as different problems and being treated in different ways, either from the front end or back end.

Firstly, in traditional pose-feature SLAM, while doing data association at the front end, there are unavoidable outliers coming from feature matching and sensor noises. How to get rid of them is a classical problem. Nowadays, RANdom SAmple Consensus (RANSAC) [1] is widely utilized to find the motion model for the camera that can encompass most available feature points as inliers. Once such a model is found, the points that are inconsistent with the model are regarded as outliers and discarded. Since the proportions of this kind of outliers are usually very low, RANSAC can do very well in most cases [2, 3], although it is an iterative and non-deterministic method.

Secondly, dynamic scenarios can bring in another kind of outliers. As we know, the traditional solution framework of SLAM assumes that everything is static except the camera. And when it comes to dynamic scenarios, the effects of moving objects can be very serious. In this case, we do not care about what the moving objects are and how they are moving; instead, our sole purpose is just to filter out these extra outliers together with those mentioned above. RANSAC method can still work in some special cases, but as we will see, its performance will degrade rapidly as the fraction of outliers increases. Dense visual SLAM [4], which represents the latest achievement in RGB-D SLAM, makes use of the color and depth information at every pixel to do SLAM and employ t-distribution to enhance its robustness; while in [5], weighting function is utilized to handle moving objects. On the other hand, instead of relying solely on the data association stage to get rid of all of the outliers, robust kernels such as Huber function [6], and recently some other robust approaches [7-9], have also been proposed and adopted at the back end to cope with unwanted outliers leaked from the front end. The basic idea of them is to minimize the effects of outliers including dynamic elements while doing the least squares optimization instead of explicitly identifying and separating them away beforehand.

Thirdly, multibody SLAM concerns about depicting all of the parts, either static or moving ones, in the scenarios. In such situations, we usually firstly resort to motion segmentation algorithms to decompose the dynamic scenarios into different motion groups and then apply the current solution framework of static SLAM to handle each of them.

As a matter of fact, in multibody SLAM, moving objects become one of our focuses instead of unwanted, useless pseudo noises; at the same time, we still need to deal with outliers caused by feature matching and sensor noises. So, to some degree, multibody SLAM is a more general task, and those situations we have mentioned before constitute some of the special cases of it. It means that motion segmentation algorithms need to be able to cope with such kind of situations as well. In other word, motion segmentation algorithms can be a general solution to all these three kind of problems.

In this paper, we propose a sparse feature-based motion segmentation. Although it is originally targeted for multibody RGB-D SLAM, we argue that a good motion segmentation algorithm can act as a unified solution to the three different problems mentioned above, and we will see that it is efficient and effective in handling the three kinds of situations in the same framework. We have tested it using simulated data and several benchmark RGB-D datasets, some of which are difficult ones, and our results are quite promising. So, we are confident that as a supplement for the traditional solution framework of static SLAM, this algorithm will enable us to do robust pose-graph SLAM in more general static and dynamic scenarios.

The structure of this paper is as follows. Firstly, we talk about RANSAC and motion segmentation algorithms in Section II. Then we propose our motion segmentation algorithm in Section III, in which no motion model or object model is assumed. After that, we choose standard RANSAC as a reference, and make detailed comparisons between them using simulated data; based on that, we further test our

algorithm using several publicly available benchmark datasets and compare our results with those of the state-of-the-art in RGB-D SLAM, i.e., dense visual SLAM [4] in Section IV. And finally we summarize the paper and talk about future work.

II. Related Work

In this section, we will talk about RANSAC and give a brief review of the related work that has made use of RANSAC to deal with outliers and moving objects in SLAM, and the available motion segmentation algorithms in multibody SLAM.

A. RANSAC and Its Application in SLAM

As an important robust estimator, RANSAC has become a standard and indispensable tool to deal with outliers in sparse feature-based SLAM [10, 11] as well as many other areas. It was firstly proposed in [1], and the standard RANSAC is composed of iterations of three simple steps: generate random hypothetical minimal inliers, fit a model, and get the consensus set. It will keep iterating till the possibility of obtaining a good model with the corresponding consensus set is above a preset value or the maximum iteration times have been reached.

Theoretically, with appropriate threshold values, given enough time, RANSAC can always find the most accurate model, but when the number of iterations is limited, the obtained solution is not guaranteed to be optimal. As we all know, in practice, time is a precious resource, so the balance between accuracy and time is always an important topic of RANSAC. Furthermore, for one of the common cases where we are only concerned about the predominant model in the data, only when outliers just constitutes a small part of the whole data, standard RANSAC turns out to be a simple and efficient solution; as the number and portion of moving parts and other outliers increase, RANSAC will become inefficient very quickly. To improve the efficiency, accuracy and robustness of the standard RANSAC, many extensions have been proposed. The basic conclusion is that RANSAC can be improved in one regard but at the expense of the others, and interested reader can refer to [12] for a detailed survey and performance evaluation of them.

On the other hand, the standard RANSAC can not be directly used to estimate multi-model in a scenario, although [13] has made use of it to detect a moving object which can be regarded as a simple case of multibody SLAM. So, some researchers propose to do sequential RANSAC [14]: apply RANSAC to get a model and its inliers, remove the found inliers from the data and then for the remaining data apply RANSAC again. The iterations will keep going till no model can be recovered from the data. However, during the process, inaccurate inliers detection will affect the estimation of the following models. To tackle this problem, multiRANSAC [15] is proposed, which does the estimation in a parallel manner. Nevertheless, it requires the user to specify the number of model instances, which is usually unknown in most cases.

So, generally speaking, although many variants of RANSAC have been proposed to partially address the shortcomings of the standard RANSAC, the problem has not been fully solved yet. And as far as we know, up till now, none of them has been applied in multibody SLAM yet.

In this paper, we are mainly concerned about robust estimation of camera poses, or ego-motion, in various situations where the static features usually amount to be the predominant part of the data, so we choose to compare our algorithm with the standard RANSAC here.

B. Motion Segmentation in Multibody SLAM

In multibody SLAM, motion segmentation algorithms have been regarded as an important step to decompose the dynamic scenario into different motion groups. And almost all of the available motion segmentation algorithms do segmentation without discrimination, which means that while doing segmentation, they do not care about which group belongs to the static environment.

In terms of targeted applications, most of the available motion segmentation algorithms are for RGB data only, quite recently some work on RGB-D data is also emerging. In our recent work [16], a brief review of the related work is given and an efficient algorithm for motion segmentation in RGB-D SLAM based on the necessary and sufficient conditions of being relatively static is proposed. However, since the segmentation algorithm solely relies on the distance values whose accuracy is limited by the capability of the sensor, i.e., Kinect, while it is good at detecting large movements, it may fail to detect small movements within the precision range of Kinect. Furthermore, the approach is restricted to scenarios composed of rigid moving objects.

As to the output, the segmentation results can be dense or sparse. In the former case, every pixel in an image will get a label associated with a motion group; while the latter only give identifications for some of the pixels in the image. To get dense results, some algorithms start from dense optical flow, which is usually obtained by employing available mature algorithms in computer vision. And therefore their accuracy depends on that of optical flow results. On the other hand, some others firstly conduct a long trajectory analysis to get a sparse segmentation result, then through a label diffusion process to get the dense result [17]. The accuracy also depends on that of its initial stage.

III. Our Motion Segmentation Algorithm

The inputs to our motion segmentation algorithm are two frames of RGB-D data, from which we can get detected and matched sparse feature pairs, and then we will separate them into different motion groups.

As we know, for each dynamic scenario composed of some moving objects, its corresponding $3D$ flow field is made up of several coherent parts, so each pairs of inlier points is not singular. Instead, each of them can find some close neighbours that have similar flow vectors. On the other hand, for those who can not find similar neighbours, it is highly possible that they are outliers. It is based on this observation that we firstly select 2 similar neighbouring pair of points and then together with the given pair to calculate the rotation matrix R and translation T across the two frames during the segmentation process.

More specifically, when judging whether two vectors are similar or not, we are using the following formula as in [18]:

$$s = \frac{|V_i - V_0|}{0.5*(|V_i| + |V_0|) + \varepsilon} \quad (1)$$

In this formula, s represents the similarity score, V_i represents the $3D$ flow vector of one of the neighbours, V_0 represents the $3D$ flow vector of the current chosen point, and ε is a constant whose value is set according to the noise level of the data. After some primitive verification, we discover that if ε is set to 1, and if $s < 0.05$, the corresponding neighbour could be regarded as similar to the current point. And these two values are fixed in this paper except for the last dataset where adaptive thresholds are used.

After getting the initial R and T with similar neighbours, we will iterate between updating R and T and inliers till no more inliers can be included. This process is similar to locally optimal RANSAC [19].

The detailed steps of our algorithm are as follows:

A. Sparse-Feature Based Motion Segmentation Algorithm

1) Given two frames of RGB-D data, do feature detection and matching, and then only keep those features with depth information.

2) Convert obtained features into (x, y, z) form, and then get their corresponding 3D flow.

3) For one pair of matched features, choose its neighbours who have similar 3D flow as a cluster to get an initial guess of R and T, based on which corresponding inliers are obtained. And we will keep iterating between updating R and T and inliers until no more inliers can be included.

If for one feature, we cannot find a minimum number (3 is used in this paper) of similar neighbours, it is highly possible that it is an outlier and we will skip it.

4) From what is left choose another pair of matched features, and then go on as step 3) has described.

5) After all of the features have been classified, choose the largest group as the static group.

B. Solution for Robust SLAM and Multibody SLAM

The chosen group and its corresponding R and T will be put into a pose-graph optimization process to get the camera poses. For robust SLAM, all of the work is done; and for multibody SLAM, it is also a good start. For example, it does not require that every moving object needs to have enough features on it in every frame; once ego-motion is known, it can help us identify various moving object and their motions types, including both rigid and non-rigid objects.

IV. Simulation And Experimental Results

Firstly, to show the efficiency of our algorithm, we compare it with the standard RANSAC (without causing confusion, we will just call it RANSAC thereafter) using simulated data. As we will see, as the fraction of the targeted group decreases, the performance for RANSAC degrades quickly while ours remains almost the same. Please note that the computation time include just one round of RANSAC, which can only tell us the inliers corresponding to the biggest group, while our algorithm will segment all the points into different groups in one go.

Then, we test our algorithm using several benchmark RGB-D datasets, which include both static and dynamic scenarios. The segmentation results are verified by the accurate visual odometry achieved, based on which pose-graph SLAM results can be obtained.

A. Results from Simulated Dynamic Scenarios

1) Simulated Dynamic Situations:

We randomly generate one thousand of 3D points (x, y, z), separate them into different groups, and then translate and rotate them using different sets of motion parameters (R and T), all of which constitute the ground-truth values.

From the 3D points, we can get their images at their different positions through a simulated RGB-D camera (its intrinsic matrix is the same as the Kinect in the real dataset), then we add Gaussian noises to the u, v and d values respectively (within 0.5 pixel for the u and v values, 1% for the d values).

2) Simulation Results

Given the two sets of u, v and d values, we then apply our algorithm (we call it *MS3D*) and RANSAC to do the segmentation respectively. The threshold for both MS3D and RANSAC is chosen to be 0.08 m.

As of the iteration times k of RANSAC, assuming that the probability of choosing an inlier component is w, if we want to ensure that we can get a subset of n-component coming from the inlier set with the probability of p, the traditional way is calculated as follows:

$$k = \frac{\log(1-p)}{\log(1-w^n)} \quad (2)$$

While directly applying this formula to our data, we find that in most cases, RANSAC cannot find a solution at all. According to our understanding, this formula just tell us that within k times, we can find one subset of inliers; on the other hand, as we know, for noisy data, we cannot expect that one subset can give us a good estimation. So, in our comparisons, we have used 30 times of k as the limit. Even at this larger limit, there are still some occasions that RANSAC can not find R and T.

If we change the portion of the predominant group, the run time for RANSAC will increase dramatically while that of our algorithm remains almost the same, as shown in Table I and Table II. More specifically, in Table I where there are two motion groups, when the percentage of the targeted group decreases from 100% to 51%, to attain the same accuracy, the consumed time for RANSAC increases from 0.01 second to

2.7 seconds (discarding those situations in which it fails to find the results) while the running time for our algorithm remains to be around 0.03 second. Similar trend can be seen in Table II which represents the five motion-group scenario.

B. Experimental Results from Static and Dynamic Scenarios

We then test our algorithm using several typical benchmark RGB-D datasets provided by TUM [20], among which are static (fr3_long_office_household), dynamic (fr3_sitting_xyz, fr2_desk_with_person) scenarios, and even a very challenging dataset that no visual odometry results have been reported before (fr3_walking_static).

TABLE I. COMPARISON OF AVERAGE COMPUTATION TIME: SCENARIOS COMPOSED OF TWO MOTION GROUPS

Percentage for The Targeted Group (%)	Average Run Time (Seconds)	
	RANSAC	***MS3D***
100	0.01	0.02
90	0.3	0.03
80	0.5	0.03
70	0.9	0.03
60	1.6	0.03
51	2.7	0.03

TABLE II. COMPARISON OF AVERAGE COMPUTATION TIME: SCENARIOS COMPOSED OF FIVE MOTION GROUPS

Percentage for The Targeted Group (%)	Average Run Time (Seconds)	
	RANSAC	***MS3D***
80	0.5	0.3
70	0.9	0.2
60	1.6	0.2
50	2.9	0.2
40	5.8	0.2
30	14.2	0.2

For each dataset, we are not using all of the data; instead, we firstly select some keyframes according to two criteria: either every two consecutive frames at least have a minimum number of common features (we choose 100 here), or have a minimum distance (we specify that the norm of the relative translation (in meters) and rotation angles (in radians) together should be bigger than 0.03) from each other. There are no further manual pruning involved in addition to them. Then the relative translation and rotation, i.e., visual odometry, is obtained after sparse feature (we are using SIFT) detection, matching and segmentation.

Based on the visual odometry results from every two consecutive frames, we can get the initial values for the camera poses; furthermore, we also check every frame with its closest neighbours within a certain distance (we use $0.2\,m$ as the limit). Those frames that have at least 100 common features with the current frame amount to be good loop closures.

With all of these R and T values, we then employ G2O [21] to do a pose-graph based SLAM. The respective visual odometry and SLAM results for each dataset are shown below (the last one only has visual odometry results because its loop closures are impaired due to shortage of common features).

Our visual odometry comes right after motion segmentation, and its accuracy can directly reflect the quality of motion segmentation results. According to [20] and [4, 5], the RMSE of relative pose error (RPE) in meters per second is a good evaluation standard for visual odometry; while the RMSE of absolute trajectory error (ATE) is especially suitable for evaluating the SLAM results. In this paper, we follow these rules and make use of the final results of dense visual SLAM [4] (called DVSLAM thereafter) as a reference.

1) rgbd_dataset_freiburg3_long_office_household

In this dataset, the camera is moving around a large table, so the outliers mainly come from feature detection, matching and/or sensor errors.

a) Motion Segmentation Result

As shown in Fig. 1, our motion segmentation algorithm can detect outliers effectively.

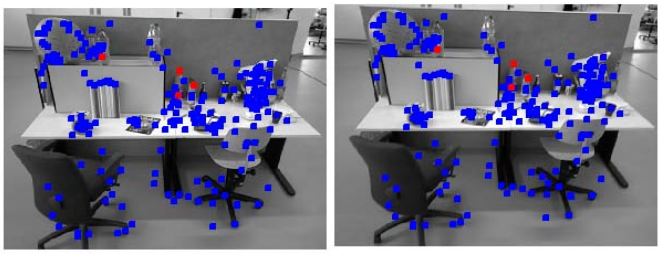

Fig. 1. Two sample motion segmentation results (best viewed in color): blue points respresent detected inliers, while red points represent outliers caused by inaccurate depth data or feature matching.

b) Visual Odometry Result

The RMSE of RPE for our visual odometry is $0.023654\,m/s$; while the RMSE of RPE for DVSLAM is $0.053943\,m/s$.

c) After Pose-graph SLAM

Our RMSE of ATE is $0.049933\,m$, and the obtained camera poses as compared with the ground truth values are shown in Fig. 2; while The RMSE of ATE for DVSLAM is $0.083816\,m$.

2) rgbd_dataset_freiburg3_sitting_xyz

In this dataset, two people are sitting in front of a table before the camera, chatting with small body movements.

a) Motion Segmentation Result

As shown in Fig. 3, our motion segmentation algorithm can detect moving points and other outliers effectively.

b) Visual Odometry Result

The RMSE of RPE for our visual odometry is $0.025184\,m/s$; while the RMSE of RPE for DVSLAM is $0.048998\,m/s$.

c) After Pose-graph SLAM

Our RMSE of ATE is 0.018450 m , and our camera poses as compared with the ground truth values are shown in Fig. 4; while the RMSE of ATE for DVSLAM is 0.060418 m .

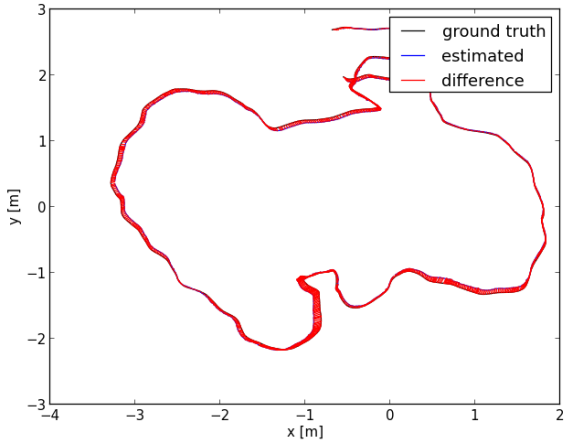

Fig. 2. Our SLAM results for the first dataset compared with the ground truth (best viewed in color): gray lines represent ground truth, blue lines represent the SLAM results, and red lines represent the differences between them at various corresponding timestamps.

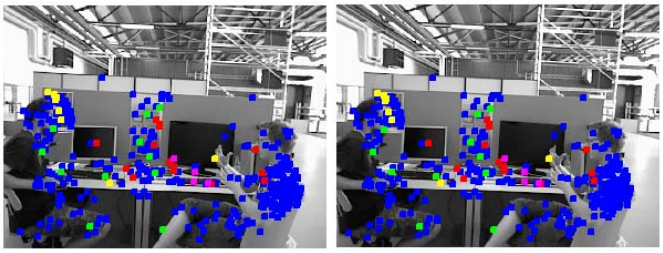

Fig. 3. Two sample motion segmentation results for the second dataset (best viewed in color): blue points respresent detected inliers, while points in other colors represent outliers caused by inaccurate depth data, feature matching, or moving objects.

3) rgbd_dataset_freiburg2_desk_with_person

In this dataset, one people comes to sit before the desk, simulating working there with some big body movements and making changes to parts of the environment.

a) Motion Segmentation Result

As shown in Fig. 5, our motion segmentation algorithm can separate the moving parts of the human body and other outliers from the static group effectively.

b) Visual Odometry Result

The RMSE of RPE for our visual odometry is 0.018190 m/s ; while the RMSE of RPE for DVSLAM is 0.018318 m/s .

c) After Pose-graph SLAM

Our RMSE of ATE is 0.055157 m , and the camera poses as compared with the ground truth values are shown in Fig. 6; while the RMSE of ATE for DVSLAM is 0.073133 m .

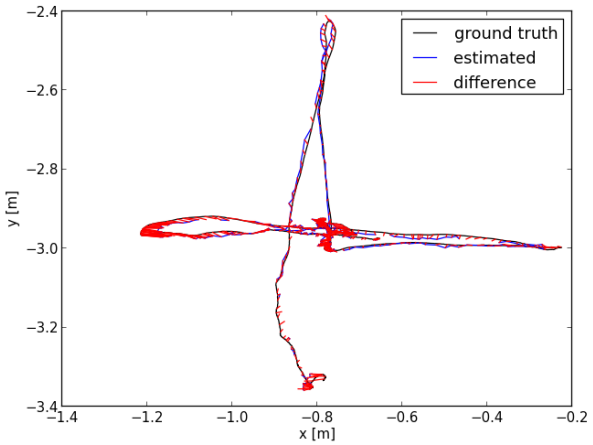

Fig. 4. Our SLAM results for the second dataset compared with the ground truth(best viewed in color): gray lines represent ground truth, blue lines represent the SLAM results, and red lines represent the differences between them at various corresponding timestamps.

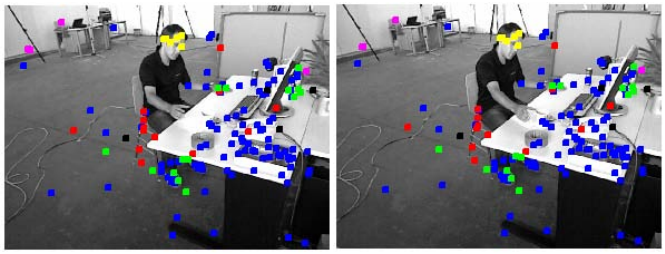

Fig. 5. Two sample motion segmentation results for the third dataset (best viewed in color): blue points respresent detected inliers, while points in other colors represent outliers caused by inaccurate depth data, feature matching, or moving objects.

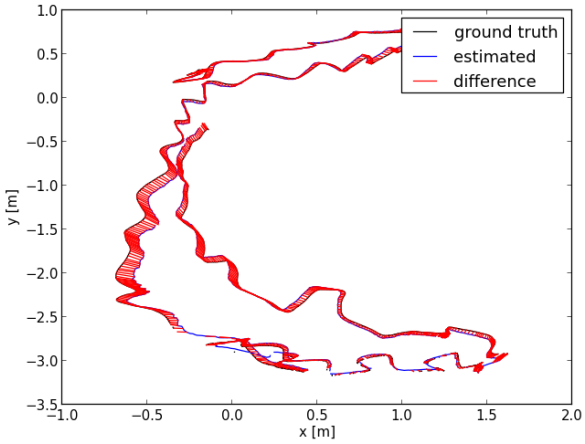

Fig. 6. Our SLAM results for the third dataset compared with the ground truth (best viewed in color): gray lines represent ground truth, blue lines represent the SLAM results, and red lines represent the differences between them at various corresponding timestamps.

4) freiburg3_walking_static

As shown in Fig. 7, in this dataset, two people are moving in front of the camera, there are blur caused by quick camera

movements and in some frames useful features are limited. Till now, no one has publish reasonable results on this dataset yet.

Since there are wide ranges of dynamic changes in the scenarios, a fixed threshold value is no longer valid for segmenting all of the frames into different motion groups. Therefore, we have modified our algorithm so that it can change the threshold value on the fly to adapt to different situations. However, similar to [22], it is based on a basic assumption that there are at least 20 common features in each pairs of frames. Nevertheless, we can only obtain relatively better visual odometry results at current stage.

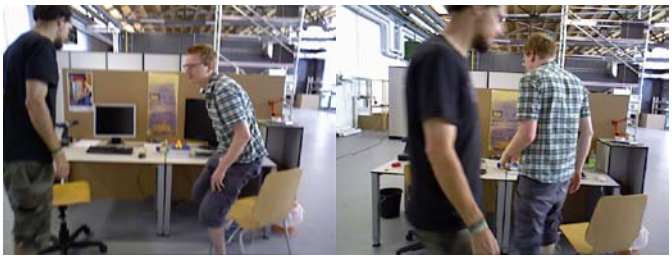

Fig. 7. Two snapshots of the fourth dataset: freiburg3_walking_static.

For our visual odometry, the RMSE of ATE is $0.161254\,m$, and the RMSE of RPE is $0.084001\,m/s$; while for DVSLAM, the RMSE of ATE is $0.469814\,m$, and RMSE of RPE is $0.309064\,m/s$.

V. Conclusion

In summary, we propose a general motion segmentation algorithm for robust RGB-D SLAM. We have shown that together with the solution framework of static SLAM, it is able to handle outliers and dynamic objects in an efficient, effective and unified manner. Using real datasets, we have achieved promising results that are comparable or better than what has been achieved using dense method [4, 5].

While dealing with the "walking" datasets of TUM, motion blur and textureless areas are pervasive; as a result, the number of useful features drops quickly, preventing us from getting better results out of them. To fully address this problem, image deblurring and ICP need to be integrated into the current framework to make it more robust and versatile.

Also, to make current motion segmentation algorithm adaptive to datasets including wide ranges of dynamic scenarios is still an open question. [22] and [23] have made some beneficial endeavors, but more work is needed to fully solve this fundamental problem.

References

[1] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381-395, 1981.

[2] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 2007, pp. 225-234.

[3] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held RGB-D camera," in *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, 2011.

[4] C. Kerl, J. Sturm, and D. Cremers, "Dense visual slam for RGB-D cameras," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, pp. 2100-2106.

[5] C. Kerl, J. Sturm, and D. Cremers, "Robust Odometry Estimation for RGB-D Cameras," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[6] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, pp. 73-101, 1964.

[7] N. Sunderhauf and P. Protzel, "Switchable constraints for robust pose graph slam," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 1879-1884.

[8] Y. Latif, C. D. C. Lerma, and J. Neira, "Robust Loop Closing Over Time," in *Robotics: Science and Systems*, 2012.

[9] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," in *Robotics: Science and Systems*, 2012.

[10] 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," *The International Journal of Robotics Research*, vol. 31, pp. 647-663, 2012.

[11] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the RGB-D SLAM system," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 1691-1696.

[12] S. Choi, T. Kim, and W. Yu, "Performance Evaluation of RANSAC Family," presented at the British Machine Vision Conference (BMVC), 2009.

[13] M. Agrawal, K. Konolige, and L. Iocchi, "Real-time detection of independent motion using stereo," in *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, 2005, pp. 207-214.

[14] E. Vincent and R. Laganiére, "Detecting planar homographies in an image pair," in *Image and Signal Processing and Analysis, 2001. ISPA 2001. Proceedings of the 2nd International Symposium on*, 2001, pp. 182-187.

[15] M. Zuliani, C. S. Kenney, and B. Manjunath, "The multiransac algorithm and its application to detect planar homographies," in *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, 2005, pp. III-153-6.

[16] Y. Wang and S. Huang, "An Efficient Motion Segmentation Algorithm for Multibody RGB-D SLAM," in *Australasian Conference on Robotics and Automation (ACRA)*, Sydney, 2013.

[17] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *Computer Vision–ECCV 2010*, ed: Springer, 2010, pp. 282-295.

[18] S. M. Smith, "ASSET-2: Real-time motion segmentation and object tracking," *Real-Time Imaging*, vol. 4, pp. 21-40, 1998.

[19] O. Chum, J. Matas, and J. Kittler, "Locally optimized RANSAC," in *Pattern Recognition*, ed: Springer, 2003, pp. 236-243.

[20] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. of the IEEE Int. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[21] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011, pp. 3607-3613.

[22] H. Wang, T.-J. Chin, and D. Suter, "Simultaneously fitting and segmenting multiple-structure data with outliers," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, pp. 1177-1192, 2012.

[23] H. Wang, J. Cai, and J. Tang, "AMSAC: An Adaptive Robust Estimator for Model Fitting," in *IEEE International Conference on Image Processing (ICIP)*, Melbourne, Australia, 2013.