Proceedings of Australasian Conference on Robotics and Automation, 2-4 Dec 2013, University of New South Wales, Sydney Australia

An Efficient Motion Segmentation Algorithm for Multibody RGB-D SLAM

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

Abstract

A simple motion segmentation algorithm using only two frames of RGB-D data is proposed, and both simulational and experimental segmentation results show its efficiency and reliability. To further verify its usability in multibody SLAM scenarios, we firstly apply it to a simualted typical multibody SLAM problem with only a RGB-D camera, and then utilize it to segment a real RGB-D dataset collected by ourselves. Based on the good results of our motion segmentation algorithm, we can get satisfactory SLAM results for the simualted problem and the segmenation results using real data also enable us to get visual odometry for each motion group thus facilitate the following steps to solve the practical multibody RGB-D SLAM problems.

1 Introduction

SLAM (simultaneous localization and mapping) refers to a basic problem for robots to build or update a map of a priorly unknown or partially known environment while simultaneously keeping track of their current location. The corresponding area in computer vision is also known as SfM (structure from motion). Within the last several decades, we have witnessed great progress in this area centring around doing mapping and localization in a static environment, or static SLAM. As a result, some researchers even share a common feeling that the static SLAM problem has been solved [Frese, 2010].

With the great progress we have accumulated in static SLAM, more and more researchers are interested in doing SLAM in an environment with dynamic objects from different perspectives and using various kinds of methods. Some focus on employing robust estimation methods, such as RANSAC or robust kernels, to filter out dynamic objects to make the odometry estimation and thus the SLAM results for static environment more robust [Kerl *et al.*, 2013]. And there are some others proposing to track the moving objects and do static SLAM at the same time [Lin and Wang, 2010; Wang *et al.*, 2007; Agrawal *et al.*, 2005; Ozden *et al.*, 2010; Kundu *et al.*, 2011; Roussos *et al.*, 2012]. The latter is now widely known as multibody SLAM, or multibody SfM in computer vision, and there are even some promising results on realtime videos [Kundu *et al.*, 2011].

In tackling the multibody SLAM problem, it is a natural choice for us to try to make full use of what we have learned in static SLAM. Some researchers introduce motion group for each object as another kind of variables in addition to camera poses and feauture positions, then put all of them into a batch optimization process, essentially similar to the framework of state-of-the-art static SLAM solutions. However, this is an off-line solution. On the other hand, if we do it in two steps: firstly separate the dynamic objects and static environment into different motion groups, then for each group, we can easily get the results by employing the available solutions for static SLAM in a parallel manner. Thus, looking for effective ways to do motion segmentation, constitutes an important step to decouple the multibody SLAM problem into simpler problems.

In terms of motion segmentation, ever since 1990s, numerous segmentation algorithms based on monocular data have been proposed in computer vision. Nevertheless, motion segmentation using RGB data only turns out to be a non-trivial task, especially when the number of motion groups is unkown, or when the motion groups are changing from time to time. So, the relevant methods proposed so far have strong assumptions.

In this paper, we are doing motion segmentation using the RGB-D data of Kinect, and a simple segmentation algorithm is proposed. Both simulational and experimental segmentation results show that this algorithm is efficient and robust. Furthermore, we apply it to solve a simulated mulitibody SLAM problem, and then test it on a real dataset collected by ourselves to verify its usability in the typical multibody RGB-D SLAM scenarios.

In the following sections, we will firstly give a brief review of the related work in multibody SLAM/SfM; then we will present our motion segmentation theory and algorithm, simulational results, and experimental results using our own dataset; in the end, we will discuss our algorithm for multibody RGB-D SLAM and then talk about our future work.

2 Related Work

At this part, we will give a brief review of the related work that has been done in multibody SLAM/SfM, and especially motion segmentation algorithms that have been proposed so far.

2.1 Multibody SLAM/SfM

In SLAM community, [Wang *et al.*, 2007] is the first to propose SLAM and moving object tracking (SLAMMOT) as an independent research topic. As an extension to static SLAM, it proposes to include the state of moving objects into the SLAM state vectors, and then adopt a trial and error methodology to specially detect and track moving objects, i.e., the moving objects are detected by examining the SLAM results under two different hypotheses (moving or static). From laser, monocular, to stereo data [Lin and Wang, 2010], they have been working on dynamic SLAM using different sensors. They demonstrated that moving object tracking could also improve the performance of SLAM. And we notice that they are the only one that have emphasized quantitative evaluations of trajectories of the moving objects, using the results of their laser SLAMMOT as the ground truth [Wang *et al.*, 2007].

Under some conditions, multibody SLAM/SfM can also be treated as an optimization problem essentially similar to static SLAM/SfM. More specifically, if we assume that the number of motion groups are known, and it will not change during the process, then the problem becomes assigning each object to one of the motion groups. So, if we regard the motion group indexes as another kind of variables to the state vector of SLAM, we can still solve the problem through a batch optimization process. For example, [Roussos *et al.*, 2012] propose a batch method to do dense multibody estimation and reconstruction, which can be regarded as an extension of DTAM [Newcombe *et al.*, 2011] to dynamic scenes. Notwithstanding, as we will see that in more general situations, we do not know how many motion groups there are, and how they will change during the process. To some degree, this method's strong assumptions also determine that it can only work for offline applications.

[Agrawal *et al.*, 2005] proposes a RANSAC-based realtime algorithm for detecting motions in stereoSLAM, and demonstrated its effectiveness in scenarios that contains one moving object. But its effectiveness rely on the assumptions that the static features constitute the majority of all the features. On the other hand, as noted by some other researchers [Tron and Vidal, 2007], the efficiency of RANSAC alone will be compromised significantly when there are more moving objects in the scenario. In contrast, our segmentation algorithm can handle multiple moving objects more efficiently, and we do not require that the static features must constitute the majority.

In computer vision, especially in recent years, more and more researchers have turn their attention to multibody SfM using a single camera. In [Ozden *et al.*, 2010], the authors talk about the challenges a practical algorithm for multibody SfM using a single camera is facing and thus the requirements it needs to meet, including online estimation of the changes of the number of moving objects, feature tracking, segmentation and 3D reconstruction simultaneously. Then, they propose a solution using a probabilistic model-scoring method and test it using several datasets. In [Kundu *et al.*, 2011], the authors propose another realtime incremental multibody SLAM algorithm emphasizing on combining multiple cues to deal with degenerate situations.

2.2 Motion Segmentation

Motion segmentation algorithms aim to separate dynamic objects and the static environment from each other according to their different motion types. Generally, there are some basic questions that motion segmentation algorithms need to answer: how many moving groups are there in the scenarios? What are they? Can we detect them as soon as possible? Generally, the number of motions can change from time to time, so the ideal motion segmentation algorithms should be sensitive and quick enough to detect the changes in real time.

In computer vision, ever since 1990's, motion segmentation has been regarded as a very important step for multibody SfM and many other applications. From 2D to 3D, numerous algebraic and statistical motion segmentation techniques based on different camera models have been put forward, and [Tron and Vidal, 2007] has given a detailed review of them. Most of these methods assume that the number of motion groups occurring in the scenes are known beforehand, then through a clustering process, they can allocate each point to one of the groups. Furthermore, the available algorithms call for a complete set of data about the motion process as the input, which means that their segmentation results are always delayed.

Another branch of work, which can be seen as an extension to trial and error method [Lin and Wang, 2010; Wang *et al.*, 2007], determines the number of motions through a model selection process [Chin *et al.*, 2010].

In [Kundu *et al.*, 2011], the authors propose an incre-

mental motion segmentation method by combining optical flow and two-view geometry for monocular data. It relies on strong assumptions to impose geometric constraints in degenerate situations.

On the other hand, with the RGB-D data provided by Kinect, the motion segmentation problem can be easier and thus efficient algorithms suitable for realtime practical applications are more possible. 6D vision [Franke *et al.*, 2005] also combines stereo information to detect moving cars or pedestrians, but it assumes that the egomotion is known with the help of another sensor. In contrast, we are only using RGB-D data to do the segmentation.

As we know, for static objects and moving rigid objects, the 3D relative position between any two points within them will remain the same through out the dynamic process. It is based on this point that [Perera and Barnes, 2011] proposes a simple and efficient method to segment motions using RGB-D data. In RGB-D SLAM, FOVIS [Huang *et al.*, 2011] has also made use of this property to filter out moving objects. But as the authors of [Perera and Barnes, 2011] have pointed out, their algorithm is only using a necessary not sufficient condition to segment motions, which means that even after the segmentation, we still can not safely conclude that any two points within the same motion group really have no relative movements.

However, since relative motion represents one of the essential differences between being relatively static and moving, as we will see, they can be applied to various situations, independent of the types and number of motions. If we can deduce the conditions that are both necessary and sufficient, we can get a more reliable algorithm to segment the motions in multibody SLAM, and thus we can utilize the framework of the currently available solutions for SLAM in static environment to solve multibody problem. This consitutes one of the basic ideas of this paper.

3 An Efficient RGB-D Motion Segmentation Method

We firstly try to find out the necessary and sufficient conditions that we need to check before we can safely conclude that one point has no relative motion to the other point during the process and thus they belong to the same motion group, then we propose an efficient algorithm based on this.

3.1 Problem Restated

Given two consecutive image frames, we can get two corresponding sets of points residing on different moving objects and static environment through feature detection and matching. Since we have depth, we can get their 3D positions relative to their camera centers. Then, the problem becomes: for these two sets of 3D points, how can we divide them into different motion groups?

Here, we define that *a pair of shadow points* means two 3D points whose images have been matched across the two frames. Essentially, if the matching is correct, this pair of shadow points should correspond to the same 3D point in the physical world which is called a *physical point*, and they are just the different representations of this same point in two different coordinates at two different times. Furthermore, we stipulate that the *relative distance* can only be defined and calculated between two shadow points from the same image, thus in the same coordinates. Now, suppose we have two pairs of shadow points (corresponding to two 3D physical points), we can get their relative distances with respect to each of the two images easily. From now on, we will focus on comparing these two relative distance values. Ideally, *if their corresponding 3D physical points come from the same motion group, these two values should remain the same; otherwise, we can safely conclude that these two pairs of shadow points belong to different motion groups.* Similarly, if a third pair of shadow points come, if their relative distances to the initial two pairs of shadow points remain the same, we can also safely say that their corresponding physical point come from the same group. These are the simplest situations, however. Now, if we have another or more pairs of shadow points, is it enough for us to just compare their relative distances to the first three pairs of points? Or, do we need to compare the relative distances between every possible pairs of points before we can safely draw a conclusion? The following section tries to answer this question.

3.2 Theory

With the aid of some basic geometry knowledge, we can deduce the necessary and sufficient conditions for grouping relatively static physical points.

Proposition: a physical point's position can be uniquely specified by three predefined non-colinear physical points through three distances and a side discrimination.

As we know, in $3D$ space, the potential points having a specified distance to a predefined point will constitute a sphere. Then, as shown in Fig. 1, the potential points that have predefined distances $d1$ and $d2$ to any two specified points $P1$ and $P2$ respectively ($d1 + d2 > d12$), will constitute a circle, which is the intersection of the two spheres. If we choose a third point $P3$ that does not lie on a line with the first two specified points $P1, P2$, and specify that the potential points need to have a preset distance $d3$ to the third point, then the potential points meeting the $d1, d2$ and $d3$ conditions simultaneously will lie on the intersection of the circle and the third sphere, which has at most two possible points, located on both

sides of the plane formed by the three pre-specified points respectively and symmetrical with respect to the plane.

So, if we further specify that the potential points should remain on one side of the plane formed by the three pre-specified points, then we can safely conclude that there is only one possible point that can meet the requirements.

Although this proposition just mentions physical points, for their corresponding shadow points, we can regard them as the same points captured by the camera in different states, and we can get their relative motion through comparing their relative distances in different images. So, similarly, for any pair of shadow points, during a motion process, only stipulating that its distances to three pairs of non-collinear shadow points remain the same is not enough, we need to make sure that it remains at the same side of the plane defined by these three specified pairs of points. Only when all of these conditions are met, can we saftely conclude that this pair of shadow points belongs to the motion group defined by the three specified pairs of shadow points.

More generally, we can also see that, for any further coming pair of shadow points, comparing their relative distances and side to the three pairs of specified non-collinear shadow points is enough to judge whether the additional points come from the same motion group or not. In other word, we do not need to compare the relative distances between every two possible pairs of points.

In summary, this proposition makes it possible for us to do motion grouping in an efficient way.

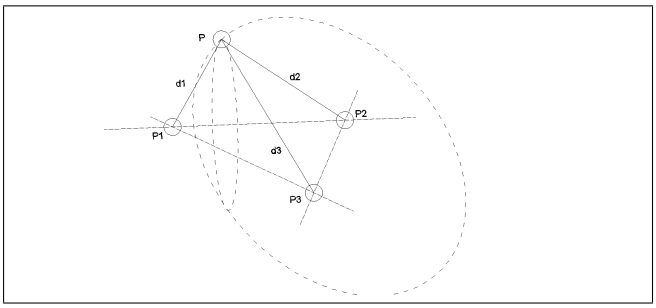

Figure 1: Conditions to constrain a point

3.3 Algorithm

The input to our algorithm are two RGB-D frames, from which we can get two sets of shadow points. And the function of our algorithm is to divide the pairs of shadow points into different groups. Firstly we will introduce a definition.

Definition: If the pair number in any group reaches three, get the plane's normal vector they have determined, then these three pairs of points become the *basic components* of this group, and the group becomes a *mature* group.

And our RGB-D segmentation algorithm is illustrated as follows:

1) For every two sequential frames, do feature detection and matching, and only those pairs of shadow points with depth are kept in an initial set.

2) Choose one pair of shadow points from the set to form a group.

3) For each following pair of points left in the initial set,

3.1) Firstly compare it with the existing groups one by one.

3.1.1) If the current group is mature, calculate the distances of the chosen pair of points to the three basic components of the group, and its relative position to the plane formed by the basic components (which side it is located). If these four results remain the same, then the chosen pair of points belongs to the current group. Othewise, compare with the next group.

3.1.2) If the current group is not mature, compare the distances of the chosen pair of points to the available one or two pairs of shadow points in the group. If the distances do not change across the two frames, then put the pair of points into the current group; otherwise compare with the next group.

3.2) Finally, if it turns out that the chosen pair of points belongs to none of the existing group, allocate a new group for it.

In this way, just after going through all of the pairs of points in one round (no iterations), we can correctly grouped them according to their motions.

The characteristics of this algorithm:

1) Efficiency: theoretically, the computation cost is $O(N_p)$ (in which N_p denotes the total number of points). Detailed analysis can be found in the Appendix.

2) Dependent on sparse features with depth information: since it is based on the essential property of being relatively static as opposed to be realtively dynamic, if we have depth information, we can use this algorithm to segment rigid body motion groups and there will be no degenerate situation. In addition, it just needs a set of sparse features as input.

3) Reliability and robustness: it checks both the necessary and sufficient conditions for rigid body motion segmentation, so theoretically its result should correspond to the real number of motion groups and the points are grouped accordingly. In practice, as we will see, inaccurate sensor data and outliers will inevitably bring in some additional extremely small groups. Nevertheless, if we just filter out such very small groups, we can still get correct results as of the motion group number and their corresponding members.

4) The thresholds are determined by the precision of

sensor data. It means that if we apply this algorithm to $3D$ point data coming from different types of sensors, we need to adapt its threshold values. For Kinect, its depth noise is about $2mm$ at $1m$, and $2.5cm$ at $3m$ [Khoshelham and Elberink, 2012], varying across the distance. In this paper, we choose $2.5cm$ as the threshold.

3.4 Segmentation Results From Simulated Data

We firstly run the algorithm on some simulated data in which we assume there is no noise in positioning and the feature matching is perfect. Some typical data and results are shown below. Fig. 2, 4 and 6 represent three simulated scenarios, in which different color represent different motion groups; Fig. 3, 5 and 7 show the motion segmentation results, in which different color represent deteced motion groups.

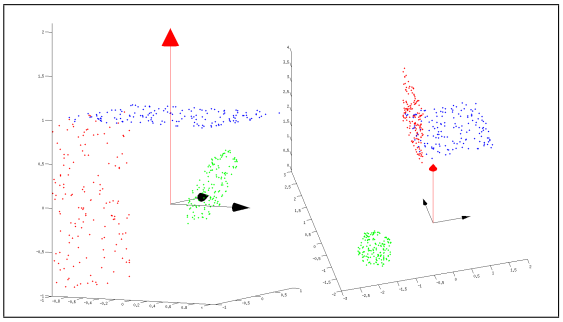

Figure 2: Simulated motion of 3 planes across two frames

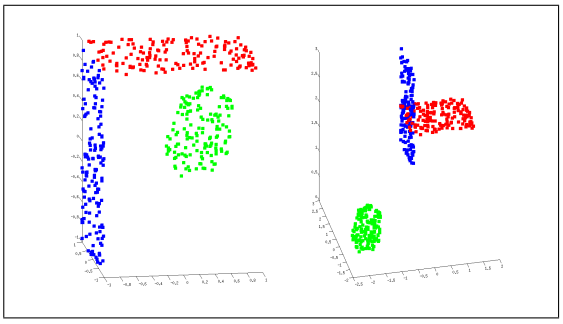

Figure 3: Motion segmentation result for the simulated 3 planes situation

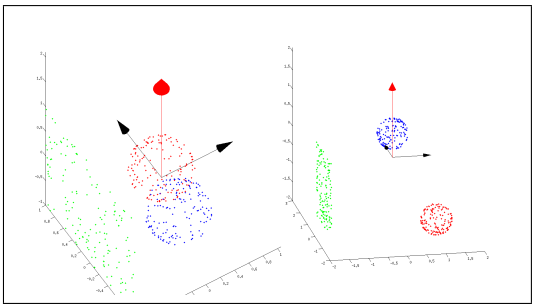

Figure 4: Simulated motion of 1 plane and 2 balls across two frames

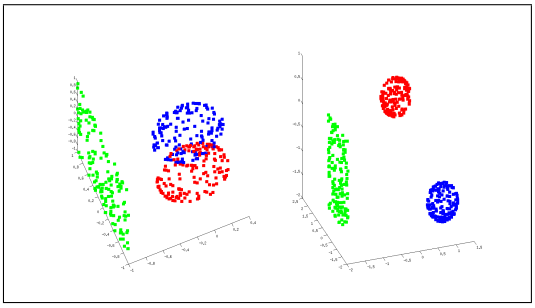

Figure 5: Motion segmentation result for the simulated 1 plane and 2 balls situation

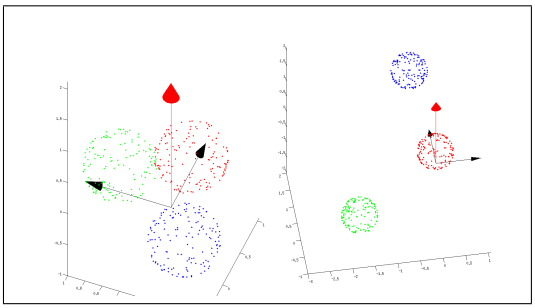

Figure 6: Simulated motion of 3 balls across two frames

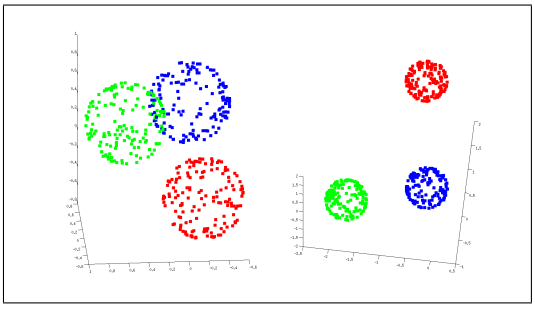

Figure 7: Motion segmentation result for the simulated 3 balls situation

3.5 Efficiency Demonstration

Using a CPU of Intel(R) Xeon@3.33GHz, the efficiency of our algorithm is shown in Table 1. In this table, the second column N_p represents the number of feature points to be grouped, the third column N_g represents the number of motion groups, which is unknown before our segmentation step, and the last column T_{mean} represents the computation time for segmentation. The first three rows represent simulational results, and the last row comes from our experimental data which will be discussed next. With unoptimized Matlab code, we can obtain the segmentaion results at the approximate rate of 7.5 points/ms, independent of motion group number, and the required time is proportional to the number of feature points, suitable for real time applications. All of these have confirmed our theoretical analysis.

Table 1: Computation time in relation with the number of feature points and groups

Dataset NO.	N_p	N_g	T_{mean}(ms)
Simulation1	450	3	55
Simulation2	300	2	38
Simulation3	150	3	19
Experiment1	152	3	21

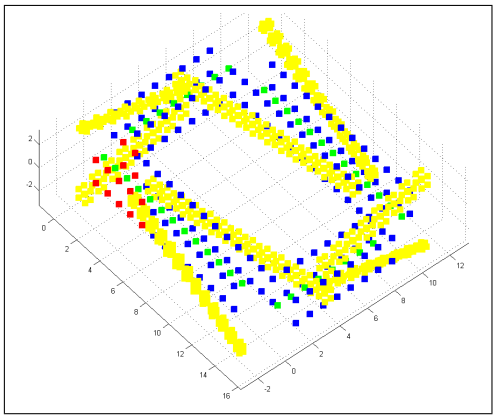

Figure 8: The simulated multibody RGB-D SLAM scenario: blue and red points represent static features, red ones are currently seen by the camera, green ones stand for camera's positions, and yellow ones are the paths of the two moving objects

4 Evaluation with A Simulated Multibody RGB-D SLAM Problem

4.1 The Simulated Multibody RGB-D SLAM Scenario

We firstly simulate a typical multibody RGB-D SLAM scenario as shown in Fig. 8, which contains a depth camera and two moving objects in addition to some static features. Throughout the process, the two objects are moving together with the depth camera along different paths, and we keep recording the ground truth images at discrete steps around a loop.

4.2 Motion Segmentation Results From Simulated Noisy Data

In order to test the robustness of our motion segmentation algorithm, we add some independent Gaussian noises to the u, v (ranging within ± 0.1 pixel) and d (within 1% of the ground truth depth value) values of the recorded images respectively, and then take the results as the input to our algorithm. With properly selected threshold, we can get good segmentation results as shown in Fig. 9.

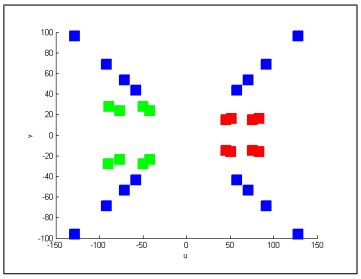

Figure 9: One segmentation result from the simulated data: blue, red and green points represent the three groups the algorithm has found, and they correspond exactly to the static features and the two moving objects respectively

4.3 Multibody SLAM Results From Simulated Noisy Data

Theoretically, based on the segmentation results, according to [Arun *et al.*, 1987], we can get pairwise visual odometry for each motion group, from which we can calculate the initial values of the features' positions and camera's poses. And thus, doing SLAM for each motion group is possible now.

However, here we will do it in a little bit different way. Firstly, we choose the static features and camera as a group to do a static RGB-D SLAM employing Gauss-Newton method for optimization. Once we have gotten the camera poses, we utilize them to calculate the positions of the dynamic features at each time step of the process.

Fig. 10, 11, 12 and 13 show our SLAM results as compared with the ground truth values.

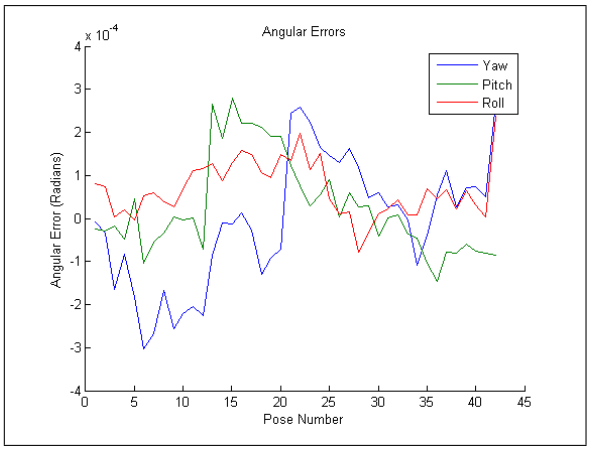

Figure 10: Angular errors of camera poses from the simulated data

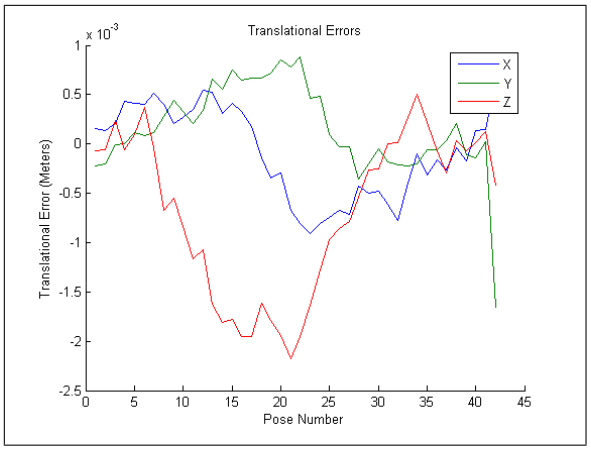

Figure 11: Translational errors of camera poses from the simulated data

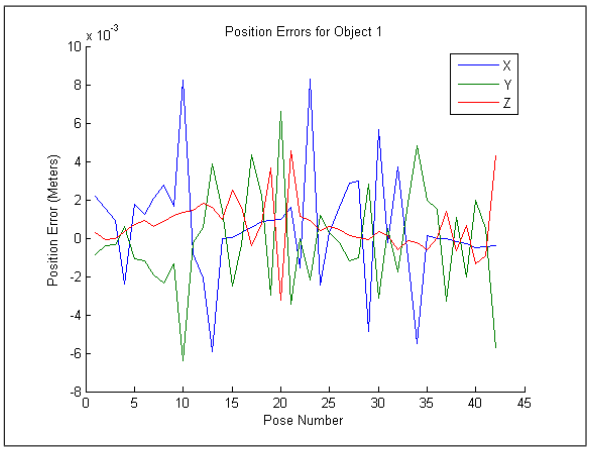

Figure 12: Average position errors of the first moving object from the simulated data

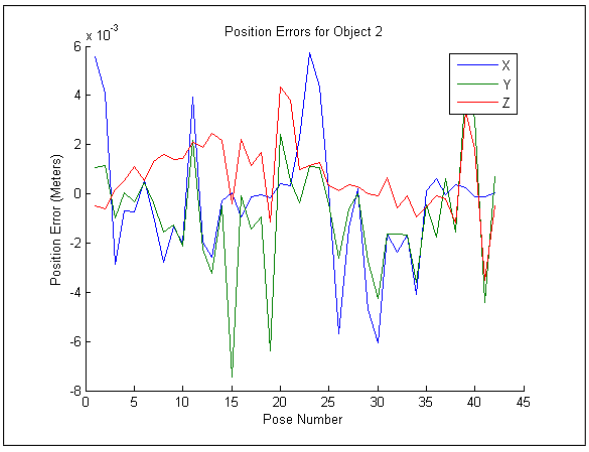

Figure 13: Average position errors of the second moving object from the simulated data

5 Evaluation Based on Our Own Dataset

5.1 Collecting Our Own Dataset

Although many benchmark datasets are available in laser and camera based SLAM, they are still relatively rare in RGB-D SLAM. For the evaluation of static RGB-D SLAM algorithms, [Sturm *et al.*, 2012] constitutes a typical set of benchmark datasets. It includes 39 series recorded in an office environment and an industrial hall, whose ground-truth trajectories are obtained from a motion-capture system. Since it aims to benchmark static RGB-D SLAM, there is only ground truth for camera poses.

We could hardly find a suitable RGB-D dataset containing rigid moving objects only to test our motion segmentation algorithm, so we have collected a small dataset with ground truth values for evaluating both our motion segmentation algorithm and visual odometry results.

As shown in Fig. 14, we choose a special space in our lab with the dimension of $4m \times 2m$. The backgound has also been decorated with some feature rich objects. In the scenario, there are two moving boxes and a Kinect sensor. From one step to the next, the two boxes, in black and blue respectively, and Kinect will translate or rotate independently, or any of them can remain static, which means that the number of motion groups is changing throughout the process. At each discrete step, we measure and record the position and orientation of each object manually. At the same time, we keep recording the synchronized RGB-D data using Kinect and ROS.

Figure 14: Experimental setup: one Kinect, two boxes (one in blue, one in black) are located at the further side of the image

5.2 Motion Segmentation Results From Experimental Data With Outliers

After getting the dataset, we firstly do feature detection and matching using SIFT [Lowe, 2004], and filter out those that do not have depth values. Then we apply our segmentation algorithm to the matched point sets. The sample feature matching and segmentation results are shown in Fig. 15 and 16. In Fig. 16, different color represents different motion groups, agreeing with the ground truth. As we can see that outliers have been seperated into small independent groups, and we can just discard such small groups. So, although the feature matching results unavoidably contain outliers, our motion segmentation algorithm can still handle them.

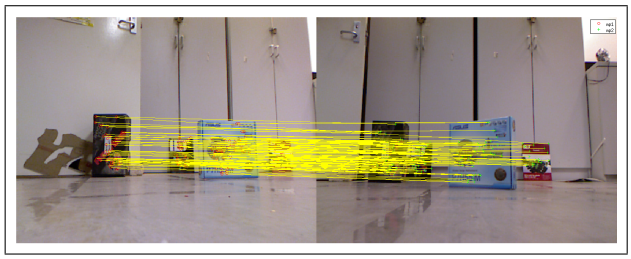

Figure 15: One sample feature matching result from the experimental data

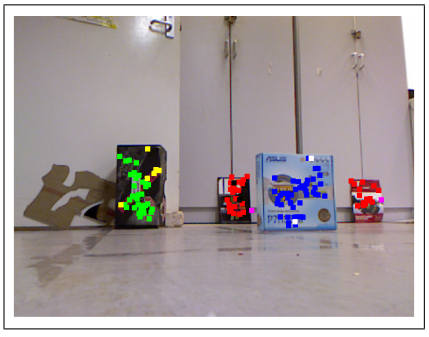

Figure 16: One sample segmentation result from the experimental data

5.3 Initial Values Based on Motion Segmentation and Visual Odometry

From the segmentation results, we choose the $3D$ feature points corresponding to the static group to calculate visual odometry, by accumulating which we can then get the initial values of the camera poses. The angular (Yaw, Pitch and Roll) and translational (x, y and z) errors of camera poses as compared with the ground truth are shown in Fig. 17 and 18 respectively. In Fig. 17, the x-axis represents frame number, and the y-axis represents the angular errors in radians; in Fig. 18, the x-axis represents frame number, and the y-axis represents the translational errors in meters.

As we can see, althougth the segmentation results are good, the visual odometry and thus the initial values turn out to be not as good as we have expected. We think it may be due to several reasons: firstly, after the division, the number of feature points for each motion group is much smaller compared with the ordinary SLAM scenarios, and thus the least squares results will degrade to some degree; secondly, we have not fully preserved the accuracy of the depth information during the image format transformation and saving process. And we will look into this matter later.

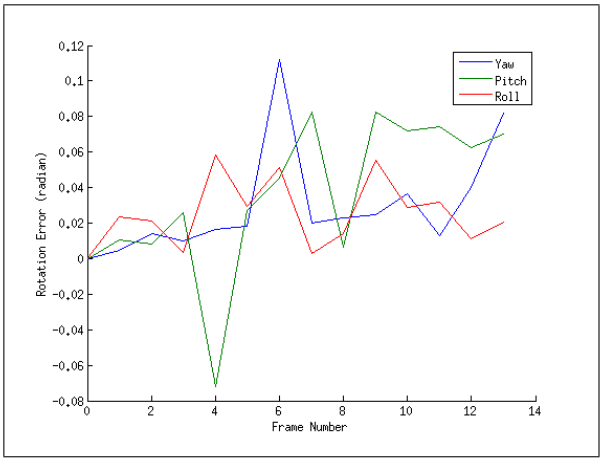

Figure 17: The angular errors of camera poses' initial values as compared with the ground truth

6 Conclusions

In this paper, we have deduced the necessary and sufficient conditions for grouping $3D$ points into different motion sets, upon which a $3D$ motion segmentation algorithm using RGB-D data is proposed. Both simulational and experimental segmentation results show that our algorithm is efficient and robust. Furthermore, we have applied it to solve a simulated multibody SLAM problem. The results are very promising. And we have

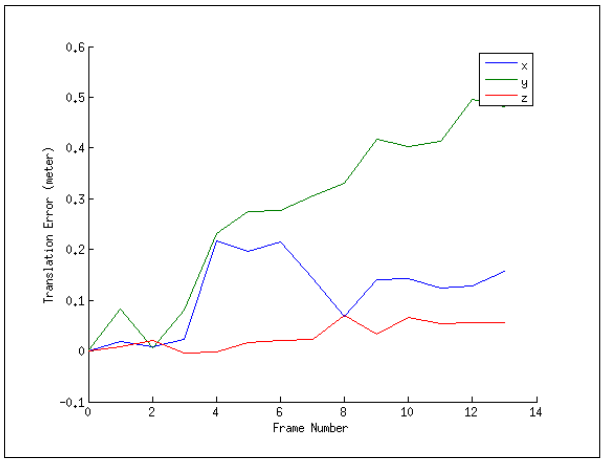

Figure 18: The translational errors of camera poses' initial values as compared with the ground truth

also shown that our algorithm can still produce good segmentation results on real dataset collected by ourselves.

However, in the simulated scenarios, correct data association is assumed. Nevertheless, data association is also a very important step for closing a loop in practice, especially when we want to totally abandon the assumption that static features always constitute the majority in the images. So, to some degree, motion segmentation is just the first step to fully solve practical multibody SLAM problems, we need to combine it with approproriate data association techniques to get good SLAM results, and we also need to compare the results of moving objects with ground truth values in the end. In other word, we need to put motion segmentation and many other elements together to setup a general framework that can handle practical multibody RGB-D SLAM problems gracefully.

On the other hand, although the motion segmentation algorithm proposed here is applicable to many dynamic scenarios composed of different rigid moving objects, when it comes to analyzing human motions which include many non-rigid deformations, it is still a challenging problem. So, once we have formulated a mature framework to handle multi-rigid-body RGB-D SLAM, next we need to extend this framework to deal with scenarios including non-rigid moving objects.

In terms of other future work, we think that buidling up a good dataset for multibody SLAM is an important work we need to address, with which most of the existing work on multibody SLAM can make enough quantitative evaluations. On the other hand, as we know, benchmark datasets have played significant roles in promoting the development of efficient and effective algorithms in many related areas. For example, in MVS (multi-view stereo), it is now a common practice that every new algorithm need to cite its quantitative results from the benchmark datasets to prove its efficiency and accuracy. Multibody SLAM as a relatively new research direction, is also in need of such kind of benchmark datasets.

Appendix: The computation cost of our motion segmentation algorithm

Now, suppose we have N_p pairs of shadow points, and they belong to N_g groups. Although the number of motion groups is unknown before segmentation, and may change a little bit during the process, in practice, it is very small when compared with the number of points. So, here we just regard it as an unknown small constant. Also, since in practice N_p is almost always larger than 7, we only consider situations under such conditions. And the unit of the cost is comparing the relative distance once or comparing the side once.

If $N_g = 1$, there is only one motion group. For the basic components, we just need to do 3 times of comaprisons to get them; for the rest points, each of them need to compare with the basic components (3+1) times. So, the corresponding total cost will be $3+4*(n-3) = 4*N_p-9$.

If $N_g \geq 2$, there are many possibilities for the sequence of the formed groups, so are the computation costs. Here we just consider two extremes. In the simplest case, most of the points belong to the first group and the rest groups contain just the repspective basic components, i.e. 3 pairs of points. For the basic components, we just need to do 3 times of comaprisons to get them; for the rest points, each of them need to compare with the basic components $(3+1)$ times. So, in total the cost for the first groups is $3 + 4 * (N_p - 3) = 4 * N_p - 9$. For the other groups, each of them need to compare with basic components in its former groups before settling up a new group. In the least situation, we will refer to the next group just after one comparison and discovering that the result does not satisfy the grouping conditions. In this case, the cost for i-th group will be $3+3*(i-1) = 3*i$, and for $i = 2..N_g$, in total it will be $3 * (N_g + 2) * (N_g - 1)/2$. Put the cost when $i = 1$ into it, we will get $4 * N_p + 1.5 * N_g^2 + 1.5 * N_g - 12$. And this is the simplest case.

Now we come to the other extreme where most of the points belong the last group. Similarly, we can see that for the basic components from group $i = 1...N_g - 1$, their cost in total is $3*N_g*(N_g-1)/2$. While for the points belonging to the last group, each of them need to compare with the basic components in the prior groups before reaching the last one, so the maximum corresponding cost is $4*N_g$. For the basic components in the last group, their cost is $3*4*(N_g-1)+3$. So, in total, the cost will be $3*N_g*(N_g-1)/2+4*N_g*(N_p-3*N_g)+3*4*(N_g-1)+3 = 4 * N_g * N_p - 10.5 * N_g^2 - 13.5 * N_g - 9$.

Putting two extremes together, we can see that the computation cost c is bounded: $4 * N_p + 1.5 * N_g^2 + 1.5 * N_g - 12 \leq c \leq 4 * N_g * N_p - 10.5 * N_g^2 - 13.5 *$

$N_g - 9$. Since the number of groups are usually much smaller than the number of points and has little changes during the process, we can regard it as a constant. Then, the computation cost is linear to the number of pairs of points, as stated in Section 3.3 of this paper.

References

[Agrawal *et al.*, 2005] Motilal Agrawal, Kurt Konolige, and Luca Iocchi. Real-time detection of independent motion using stereo. In *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, volume 2, pages 207–214. IEEE, 2005.

[Arun *et al.*, 1987] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):698–700, 1987.

[Chin *et al.*, 2010] Tat-Jun Chin, David Suter, and Hanzi Wang. Multi-structure model selection via kernel optimisation. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3586–3593, June 2010.

[Franke *et al.*, 2005] Uwe Franke, Clemens Rabe, Hernán Badino, and Stefan K. Gehrig. 6D-Vision: Fusion of Stereo and Motion for Robust Environment Perception. In *Proceedings of the 27th DAGM Symposium*, pages 216–223, Vienna, Austria, August 2005. Springer.

[Frese, 2010] Udo Frese. Interview: Is SLAM solved? *Künstl Intell*, 24:255–257, 2010.

[Huang *et al.*, 2011] Albert S Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox, and Nicholas Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *International Symposium on Robotics Research (ISRR)*, 2011.

[Kerl *et al.*, 2013] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Robust odometry estimation for RGB-D cameras. IEEE International Conference on Robotics and Automation, 2013.

[Khoshelham and Elberink, 2012] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors (Basel, Switzerland)*, 12(2):1437–54, January 2012.

[Kundu *et al.*, 2011] Abhijit Kundu, K Madhava Krishna, and CV Jawahar. Realtime multibody visual slam with a smoothly moving monocular camera. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2080–2087. IEEE, 2011.

[Lin and Wang, 2010] K.H. Lin and C.C. Wang. Stereo-based simultaneous localization, mapping and moving object tracking. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 3975–3980. IEEE, 2010.

[Lowe, 2004] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[Newcombe *et al.*, 2011] R A Newcombe, S J Lovegrove, and A J Davison. DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320–2327. IEEE, 2011.

[Ozden *et al.*, 2010] Kemal E Ozden, Konrad Schindler, and Luc Van Gool. Multibody structure-from-motion in practice. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(6):1134–1141, 2010.

[Perera and Barnes, 2011] Samunda Perera and Nick Barnes. A Simple and Practical Solution to the Rigid Body Motion Segmentation Problem Using a RGB-D Camera. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 494–500. IEEE, 2011.

[Roussos *et al.*, 2012] Anastasios Roussos, Chris Russell, Ravi Garg, and Lourdes Agapito. Dense multibody motion estimation and reconstruction from a handheld camera. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 31–40. IEEE, 2012.

[Sturm *et al.*, 2012] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the IEEE Int. Conf. on Intelligent Robot Systems (IROS)*, 2012.

[Tron and Vidal, 2007] Roberto Tron and René Vidal. A Benchmark for the Comparison of 3-D motion Segmentation Algorithms. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[Wang *et al.*, 2007] C.C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, 2007.