Dynamic 3D Bounding Box Estimation for Video Segmentation from a Non-Stationary RGB-D Camera

Naveed Ahmed

Department of Computer Science
University of Sharjah
Sharjah, United Arab Emirates
Email: nahmed@sharjah.ac.ae

***Abstract*—We present a new method for video segmentation of RGB-D video data acquired from a non-stationary RGB-D camera. Our method uses a novel method of dynamic 3D bounding box estimation for moving objects that correctly classifies foreground and background elements of a scene even in the presence of high camera motion. Starting from the acquisition of a dynamic object using a non-stationary RGB-D camera, our method finds the mapping between different frames of the video data using the image features. The mapping between the image features derives our novel method of dynamic 3D bounding estimation that correctly segments the moving object in spite of the camera motion. The segmented video data can be employed in a number of applications, e.g., video editing, motion capture, action recognition, visual FX processing, or free-viewpoint video.**

Keywords–RGB-D Video; Non-Stationary RGB-D Camera; Video Segmentation, 3D Animation.

I. INTRODUCTION

Background segmentation is one of the important steps that is employed at the start of a number of Computer Vision algorithms. It deals with separating the foreground from the background by correctly classifying the parts of the image in either category. The algorithm when applied on the video data to separate two or more segments is called Video Segmentation. Traditionally, the video segmentation is done against a static background using the color cameras, and a number of methods have been proposed in this area [1]. The resulting segmented video from these methods is then employed in a number of applications [2] [3] [4]. A video acquired from a stationary camera is suitable for a static setting similar to a recording studio that requires some dedicated space for acquisition. The cameras are mounted at some specific points and the moving object is confined to a specific area.

In recent years, with the advent of mobile phones and smaller cameras, more and more video is captured from non-stationary cameras. In addition, the arrival of consumer grade RGB-D cameras, e.g., Microsoft Kinect [5], have opened new ways to capture true 3D video using a single camera. The captured 3D video is used in applications ranging from video games to 3D visualizations on mobile, or virtual reality devices. In order to correctly visualize a moving object captured from a non-stationary RGB-D camera, it is important to segment the video data correctly into background and foreground. The problem is very challenging because, in the data from a moving camera, the background and foreground are both moving and thus the algorithms that rely on the static background fail on this type of data.

Recently, a number of new methods are proposed in the area of segmentation of video data from non-stationary cameras [6] [7]. Lim et al. [8] estimate the fundamental matrix from frame correspondences to label the foreground and background pixels. Kwak et al. [9] extended this approach with a non-parametric framework. Zhang et al.[7] estimate the full camera motion for the true 3D reconstruction of the scene and then used the depth information to label the foreground and the background. Sheikh et al. [10] and Cui et al. [11] presented factorization-based approach from the tracked points. Narayna et al. [12] presented a method of video segmentation from a moving RGB camera using optical flow. Yi et al. [13] presented another method for video segmentation from a moving RGB camera using dual-mode Single Gaussian Model. Background segmentation using depth data has also received lots of attention in the last couple of years [14]. Koutlemanis et al. [15] presented a method of foreground detection with a moving RGB-D camera while using an initial background frame as the reference model. Zamalieva et al. [16] employed a tracklets-based method to estimate the epipolar geometry using the temporal fundamental matrix [17] of the scene and a labeling method for video segmentation.

In this paper, we present a new method for video segmentation of RGB-D video data acquired from a non-stationary camera. We start from the data acquired from a non-stationary Microsoft Kinect v2 RGB-D camera. Unlike the method from Koutlemanis et al. [15], our method does not rely on any a prior background information. Similar to Zamalieva et al. [16], we start by estimating the feature points and their matches over an interval in the video sequence. Our matching algorithm does not need dense matching, as used by [16], therefore instead of using optical flow, we only need a sparse matching. Afterward, we present a novel dynamic 3D bounding box estimation method that provides a solution in three-space, which is employed to segment all the frames of the RGB-D video sequence. The result of our work is a segmented video data using a novel algorithm from a non-stationary RGB-D camera.

In the following sections, we detail our segmentation method, first, the data acquisition and calibration is presented in Section II, afterward, the video segmentation algorithm is explained in Section III, followed by results (Section IV) and conclusion (Section V).

 ISBN: 978-1-61208-464-0

Figure 1. (left) RGB frame, (middle) Depth frame. (right) 3D point cloud resampled from mapping the depth frame to three-space coordinates. Mapping between the depth and RGB frames is used to determine the color of each 3D point.

II. Data Acquisition and Calibration

We acquire the RGB-D data of a moving subject using Microsoft Kinect v2 sensor [5]. This most recent version of Kinect can record full HD (1920x1080) RGB data at 30 frames per second. Compared to Kinect v1 (640x480), it is a significant increase. Kinect v2 can record the depth data at the resolution of 512x424 at 30 frames per seconds. In comparison to Kinect v1 depth data resolution of 320x240, the increase is modest but the underlying acquisition mechanism is enhanced to capture the higher density of depth data.

In our work, we use a single moving Kinect to acquire the RGB-D data at these default resolutions and frame rate. Our method is not confined to a single Kinect and can be easily extended to a multiple Kinect setup, as demonstrated by Ahmed et al. [18]. We capture both depth and color streams that are directly saved to the memory to avoid IO overheads and then later written to the disk once the recording is finished. Our acquisition tool is implemented using Microsoft Kinect SDK 2.0. It allows the acquisition of both RGB and depth data through a USB 3.0 interface. While acquiring the data we manually move the Kinect (rotation and translation) so that the condition for the non-stationary camera is fulfilled. To this end, we capture two images, one RGB and one depth, for each frame of the captured sequence. A captured RGB, and depth frame can be seen in Figure 1(left, and middle).

The acquired RGB and depth data have different resolutions, i.e., RGB data is stored in an 8 bit color image of size 1920x1080, whereas the depth data is stored in a 16 bit greyscale image of size 512x424. A camera acquisition setup requires a number of calibrations to determine internal (intrinsic) and external (extrinsic) camera parameters. The intrinsic camera parameters are required to determine the projection of 3D world on the camera's image plane, while the extrinsic parameters are required to estimate the camera's position in the real-world. A Kinect is comprised of two cameras, RGB and depth. Thus in addition to the intrinsic calibration of each camera, an extrinsic calibration between the two cameras is needed, so that the depth camera can be mapped to the color camera or vice versa. Moreover, the depth camera returns one depth value for each pixel in the depth image that has to be mapped to the real-world three-space coordinates, if the depth data is to be visualized in the form of a 3D point cloud. Since we are only working with a single Kinect, our method does not directly need an extrinsic calibration between multiple cameras.

We use Microsoft Kinect SDK 2.0 to determine all the required camera calibration parameters. Kinect SDK 2.0 provides the extrinsic calibration from color to depth that is stored in a 1920x1080 file with two floating point values stored for each pixel. In addition, the mapping from a depth value to the real-world three-space value is acquired after the data acquisition is finished to store the data also in the form of a 3D point cloud. A resampled 3D point cloud with the mapping to RGB data can be seen in Figure 1(right).

III. Video Segmentation

As explained in the previous section, the acquired RGB and depth data are resampled in a 3D point cloud with the RGB mapping. This 3D point cloud data shows all the acquired points comprising of the moving actor and the static background, as can be seen in Figure 1(right). In general, for most of the applications that use the video data of a moving actor for further analysis, the only relevant part of the video is the actor rather than the static background. The process of separating the foreground from the background is the well-known process of "Background Subtraction" or in the case of video data "Video Segmentation", which has been employed in a number of methods for a number of years. Traditionally, most of the algorithms for video segmentation rely on the stationary camera to correctly estimate the static background [1], or rely on an initial manual estimate of the foreground or background. These methods have been proved extremely useful for a number of applications, but do not work if the data is acquired from a moving camera. In recent times, more and more data is acquired from the moving cameras, thus it is imperative to have methods that can perform the video segmentation for the non-stationary cameras.

For the non-stationary camera video segmentation, our method relies on the acquired RGB-D video data along with the mapping from depth to RGB data (Section II). Our method does not make any assumptions about the background, and does not need any initial estimate for the background. As the starting point, we extract Speeded Up Robust Features [19] (SURF) in all the RGB frames. These feature points are then matched over two frames to estimate the camera and object motion. Our main assumption is that given the moving camera, the motion of the background and the dynamic background will be different in terms of their velocity. The background motion being static will be dependent on the camera motion, whereas the foreground being dynamic will have its own motion in addition to the camera motion. It is to be noted that the background refers to the static part of the scene and it can have different depth values, and thus the depth information

 ISBN: 978-1-61208-464-0

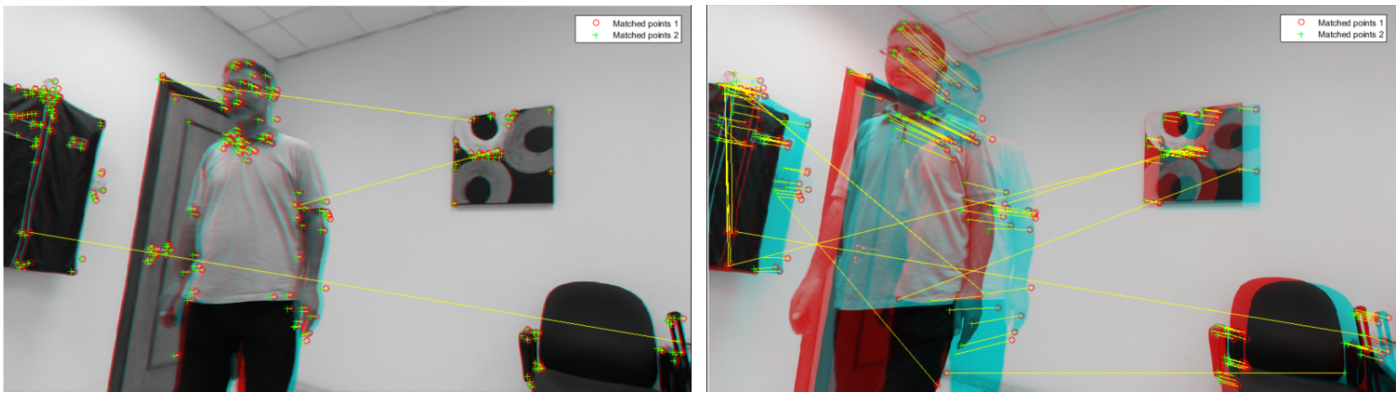

Figure 2. (left) shows the feature matching if two consecutive frames are used. (right) shows the feature matching over the interval of 10 frames. The motion is more pronounced when the frame distance is greater. Incorrect matches are automatically filtered before any additional processing.

alone cannot segment the foreground. Similarly, our method would not work if there is no movement in the foreground.

The matching of feature points provides us with the movement of each feature point over time. The matching gives both the speed and direction (velocity) of each feature point. The main assumption of our algorithm is that foreground and background should exhibit different velocity in terms of their feature points and in addition, the higher proportion of feature points will belong to the background. Kinect records data at 30 fps, thus frame to frame motion might be very small if the camera or the object are not moving very fast, and it would be impossible to differentiate between the foreground and the background based on the limited motion. Therefore, instead of comparing two consecutive frames, we compare frames at an interval of 10 frames that provides enough motion to differentiate between the potential foreground and background feature points. The interval size is chosen after analyzing the camera motion from the SURF matching. A sequence with the fast moving camera can have a smaller interval. In our method, we make sure that the matching distance should be at least greater than 2% of the image width. This satisfy the criteria of having a difference of more than 98% dissimilarity between the two frames. A comparison of feature point matching can be seen in Figure 2.

The initial match at the interval of 10 frames provides us with the starting point for our algorithm. Based on these matches, we first discard incorrect matches. The incorrect matches can occur due to the underlying SURF feature matching algorithm that is not the contribution of our work, rather we use it as it is. Incorrect matches are easier to discard using simple sanity checks based on the standard deviation of the speed and direction of all the matches. The feature matches that do not lie within the 95% of the confidence interval are discarded.

The filtered matches are then classified into multiple groups. We pick a feature match at random. The selected feature match is then used to find all the feature matches that are closer to it in terms of its velocity. We use a threshold of $\pm 10\%$ to find similar feature matches. Once the iterative process is finished, a group of feature matches is formed. From the remaining matches, a new feature match is randomly selected and the same iterative process of finding similar feature matches from the remaining set is repeated. This process is repeated till all the feature matches are classified in one of the groups. At the end of the process, a number of groups of matches will be found. The number of groups will depend on the type of the motion of the moving object. For example, a dynamic object with a rigid body motion couple with a static background will have small number of groups. The background will move according to the camera motion, while the dynamic object will have only one additional movement in addition to the camera motion. Other non-linear motions, e.g., a human, would depict different types of motion, i.e., arms moving in one direction while the body moves in the other direction. This combined with the camera motion will result in more groups of matches for each type of motion.

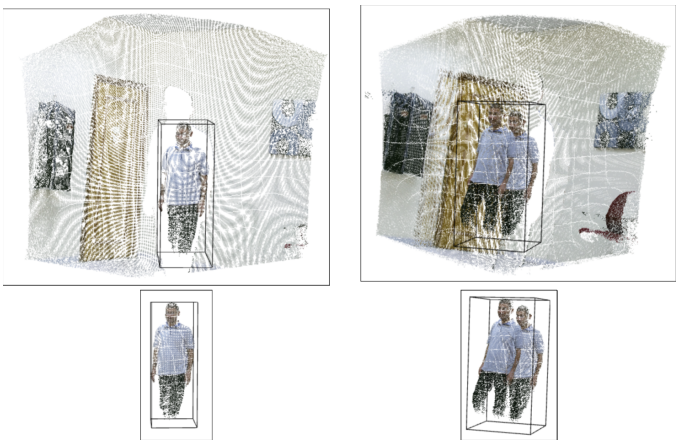

Figure 3. (left) shows the 3D bounding box localizing the foreground at one particular frame. (right) shows the dynamic bounding box spanning over the period of 10 frames that is used to segment the foreground over that interval.

In the end, once the groups are formed, our main assumption is that the group with the maximum matches will belong to the background and the rest of the groups will be classified as the foreground matches. This assumption is valid in our case, as most our scene is made up of the static background that appears to move because of the camera motion and there is only one dynamic object. If there are more than one dynamic objects in the scene then a different approach would be required to further classify the foreground into different segments, as discussed in Section IV.

Now that the feature points that belong to the foreground are identified, they are used to estimate a dynamic 3D bounding box over the interval of 10 frames. To get this bounding

Figure 4. (left) the original RGB frame, (middle) segmented background, and (right) segmented foreground. The motion of the object and the camera is evident from these images.

box, the calibration results (Section II) are used to first find the mapping of the foreground feature points to the depth image. The depth to three-space mapping provides the 3D points corresponding to each feature point. Based on the depth data of all the feature points that belong to the foreground, we find the minimum and maximum depth value in these feature points. This provides us a 3D bounding box in the depth image at both frames. This bounding box is expanded by 20% on each side to get a conservative estimate of the true foreground. If the grown bounding box incorporates a background matching feature then its size is reduced unless the background feature point is removed from the bounding box. All the pixels in the depth image inside this bounding box at each frame are then classified as the foreground. The depth to three-space mapping also provides us with the bounding box that can segment the 3D point cloud at each frame, Figure 3(left). The minimum and maximum values of the two bounding boxes at 10 frame interval provide a stretched 3D dynamic bounding box that encompasses the motion of the object over these frames, Figure 3(right).

The stretched 3D dynamic bounding is then used to segment all the in-between frames. The approach works in both depth image, or in the three-space. All points that lie within this stretched 3D dynamic bounding box from frame 2 to 9 are classified as the foreground and the background points are discarded. The algorithm is then trivially extended iteratively over the next 10 frames till the end of the sequence. Some more results of the video segmentation can also be seen in Figure 4.

IV. RESULTS

We recorded a data set to test and validate our method. The data set is 200 frames long and was acquired using Kinect v2 via our acquisition system (Section II). In the recorded RGB-D sequence, the object depicts a walking motion towards the camera, while the camera rotates and moves freely. Our method was able to segment the foreground from the static background completely, even with the camera motion and the classification of the foreground and background was excellent based on the visual analysis. It can be seen in the results, Figure 3 and Figure 4 that our method is able to successfully segment the foreground even in the case of high camera motion. As our method depends on the depth data for the segmentation, the slight artifacts in the results are due to the missing depth data that is a limitation of Kinect, rather than our method.

Our method is very efficient. We consider calculating SURF features and matching as a pre-processing step. In terms

 ISBN: 978-1-61208-464-0

of actual run-time of the algorithm, it takes around a second to estimate the dynamic 3D bounding box over 10 frames and segment the remaining 8 frames. Thus, a sequence of 200 frames is processed within 20 seconds. The main bottleneck is the file output that can take additional 40 seconds for all the frames for the segmented foreground and background.

Our method is subject to a couple of limitations. The method relies on the SURF matching that depends on the quality of RGB data. In general, it works fine because Kinect v2 captures full HD video at 30 fps that results in the sharp high quality video that is suitable for finding a good number of feature points and matches under the assumption that the motion is not very fast. For a very fast motion, a higher frame-rate camera will be required. Our choice of 10 frames interval seems arbitrary but it works well in practice. The number of frames in the interval depend on the type of motion. If the motion is fast then the interval should be short and vice versa. One can quantify the interval by creating a heuristic over the velocity of feature points. If the speed is very high over 10 frames then the interval can be reduced to bring the speed within a certain limit. In future, we would like to implement this method to further automate the segmentation process. Additionally, our method works fine for a scene comprising of a single dynamic object, but at the moment we do not provide any method to further segment the foreground in case of multiple dynamic objects. It is a challenging problem that we are considering for the future work. Finally, we do not validate the goodness of our method quantitatively but rather through the visual analysis. It is to be noted that as the camera is moving there is no ground truth available for us to compare the background to the foreground. So far, we have resorted to the visual analysis of the results for the validation, but for the future work we will also generate synthetic data for the ground truth validation.

Despite the limitations, our method shows that it is possible to do segmentation of the video data from a moving RGB-D camera using both RGB and depth data.

V. Conclusions

We presented a new method to segment video data from a moving RGB-D camera. Our method uses Kinect v2 to acquire both the RGB and depth video data together with intrinsic and extrinsic parameters of both RGB and depth cameras. Initially, SURF algorithm is used to find feature points in RGB images that are matched over an interval of 10 frames to get a meaningful classification of feature points belonging to foreground and background based on their velocities. The feature points are segmented into two clusters, and the size of the clusters is used to identify the foreground feature points. The foreground feature points are then mapped to depth coordinates that are used to find a bounding box of the foreground in each frame based on the minimum and maximum depth value. The two 3D bounding boxes at 10 frames interval is then used to estimate a dynamic 3D bounding box over the 10 frames interval that is used to segment the foreground over the in-between frames. The method is then iteratively extended over the next 10 frames till the end, to segment the complete sequence. The method allows a general purpose video segmentation of dynamic objects from a non-stationary RGB-D cameras. It can be used in a number of scenarios where an RGB-D camera can be arbitrarily deployed to capture a scene without any constraints in terms of positioning the camera. In future, we plan to extend our work to include validation using the synthetic data and further segment the foreground to identify multiple dynamic objects.

References

[1] S. H. Shaikh, K. Saeed, and N. Chaki, Moving Object Detection Using Background Subtraction. Springer, 2014.

[2] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," ACM Trans. Graph., vol. 22, no. 3, 2003, pp. 569–577.

[3] J. Starck and A. Hilton, "Surface capture for performance-based animation," IEEE Computer Graphics and Applications, vol. 27, no. 3, 2007, pp. 21–31.

[4] E. de Aguiar et. al., "Performance capture from sparse multi-view video," ACM Trans. Graph., vol. 27, no. 3, 2008, pp. 98:1–98:10.

[5] MICROSOFT, "Kinect for microsoft windows. http://www.kinectforwindows.org/," November 2010, retreived: Jan, 2016.

[6] F. Liu and M. Gleicher, "Learning color and locality cues for moving object detection and segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 320–327.

[7] G. Zhang, J. Jia, W. Hua, and H. Bao, "Robust bilayer segmentation and motion/depth estimation with a handheld camera," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 33, no. 3, 2011, pp. 603–617.

[8] T. Lim, B. Han, and J. H. Han, "Modeling and segmentation of floating foreground and background in videos," Pattern Recogn., vol. 45, no. 4, Apr. 2012, pp. 1696–1706.

[9] S. Kwak, T. Lim, W. Nam, B. Han, and J. H. Han, "Generalized background subtraction based on hybrid inference by belief propagation and bayesian filtering." in ICCV, D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, Eds. IEEE, 2011, pp. 2174–2181.

[10] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in Computer Vision, 2009 IEEE 12th International Conference on. IEEE, 2009, pp. 1219–1225.

[11] X. Cui, J. Huang, S. Zhang, and D. N. Metaxas, "Background subtraction using low rank and group sparsity constraints," in Computer Vision–ECCV 2012. Springer, 2012, pp. 612–625.

[12] M. Narayana, A. Hanson, and E. Learned-Miller, "Coherent motion segmentation in moving camera videos using optical flow orientations," in Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE, 2013, pp. 1577–1584.

[13] K. M. Yi, K. Yun, S. W. Kim, H. J. Chang, and J. Y. Choi, "Detection of moving objects with non-stationary cameras in 5.8ms: Bringing motion detection to your mobile device," in IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2013, Portland, OR, USA, June 23-28, 2013, 2013, pp. 27–34.

[14] K. Greff, A. Brandão, S. Krauß, D. Stricker, and E. Clua, "A comparison between background subtraction algorithms using a consumer depth camera." in VISAPP (1), 2012, pp. 431–436.

[15] P. Koutlemanis, X. Zabulis, A. Ntelidakis, and A. A. Argyros, "Foreground detection with a moving rgbd camera." in ISVC (1), ser. Lecture Notes in Computer Science, G. Bebis, R. Boyle, B. Parvin, D. Koracin, B. Li, F. Porikli, V. B. Zordan, J. T. Klosowski, S. Coquillart, X. Luo, M. Chen, and D. Gotz, Eds., vol. 8033. Springer, 2013, pp. 216–227.

[16] D. Zamalieva, A. Yilmaz, and J. W. Davis, "Exploiting temporal geometry for moving camera background subtraction," in Pattern Recognition (ICPR), 2014 22nd International Conference on. IEEE, 2014, pp. 1200–1205.

[17] A. Yilmaz and M. Shah, "Matching actions in presence of camera motion," Comput. Vis. Image Underst., vol. 104, no. 2, Nov. 2006, pp. 221–231.

[18] N. Ahmed, "A system for 360 acquisition and 3d animation reconstruction using multiple rgb-d cameras," in Computer Animation and Social Agents (CASA), 2012, pp. 9–12.

[19] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," Comput. Vis. Image Underst., vol. 110, no. 3, Jun. 2008, pp. 346–359.

 ISBN: 978-1-61208-464-0