Using Multiple RGB-D Cameras for 3D Video Acquisition and Spatio-Temporally Coherent 3D Animation Reconstruction

Naveed Ahmed and Imran Junejo

***Abstract*—We present a system for spatio-temporally coherent 3D animation reconstruction from multi-view RGB-D images. Our system captures multi-view synchronous RGB-D images from six RGB-D cameras and we show that by using the unique properties of both depth and color images, it is possible to reconstruct a spatio-temporally consistent 3D animation from a non-coherent time-varying data. The reconstructed spatio-temporally coherent 3D animation can be used in a number of applications that require time-coherent data, e.g. motion analysis, gesture recognition, compression, free-viewpoint video and CG animations.**

***Index Terms*—Multi-view video acquisition, 3D and free-viewpoint video, dynamic scene reconstruction, RGB-D data acquisition.**

I. Introduction

First step in obtaining spatio-temporally coherent 3D video is to capture the shape, appearance and motion of a dynamic real-world object. One or more video cameras are employed for this acquisition, but unfortunately, data obtained by these video cameras lacks temporal consistency, as there is no relationship between the consecutive frames of a video stream. In addition, for a multi-view video, all the cameras have to be synchronized to extract temporal correspondences at each frame of the video, which is typically achieved by means of a hardware-based camera trigger. From the acquired synchronized data, in order to reconstruct a spatio-temporally coherent 3D animation, a spatial structure between cameras has to be established along with the temporal matching over the complete video data. In this paper we present a system for acquiring synchronized dynamic 3D data using multiple RGB-D cameras along with a new method for capturing spatio-temporal coherence between RGB-D images captured from multiple RGB-D video cameras.

Synchronized multi-view video (MVV) data is used in a number of applications, e.g. motion capture, dynamic scene reconstruction, free-viewpoint video etc. Traditionally, the MVV recordings are acquired using synchronized color (RGB) cameras, which are later processed for use in a number of applications [1]. The acquisition setups used for these earlier works comprised of a dedicated system for capturing synchronous high quality RGB MVV recordings, which were then used to reconstruct dynamic 3D scene representation. Previously, a number of methods have been proposed toreconstruct spatio-temporally consistent 3D animation from MVV data [2], [3]. However, the arrival of Microsoft Kinect [4] has generated a new wave of interest in this area. One or more depth sensors are employed in 3D shape scanning and dense 3D reconstruction of static objects; pose, motion and 3D shape estimation [5]-[7]. These works show that despite the limitation of depth sensors, i.e. low resolution and high noise, it is possible to employ them to get high quality results.

One or more depth sensors have been employed in a number of applications to reconstruct a three-dimensional representation or static and dynamic objects [8], [9]. Using multiple Kinects, Kim *et al.* [10] presented the design and calibration of a system that enables simultaneous recording of dynamic scenes with multiple high-resolution video and low-resolution ToF depth cameras. Unlike our system, their system relied on hardware trigger for the explicit synchronization of color and depth cameras. Berger *et al.* [5] employed four Kinects for marker-less motion capture. Since their area of application was silhouette-based motion capture, they did not explore the use of multiple Kinects in generating dynamic scene geometry. They also assume that Kinects are synchronous and did not actively try to create a setup for the synchronous capture. For motion capture, it can be assumed that synchronization is not a primary requirement as shown by Hasler *et al.* [11]. Nevertheless, for a dynamic scene reconstruction setup, which merges the data from multiple cameras, a higher degree of synchronization is required to produce a correct 3D animation. Both of the methods [5], [10] do not try to extract any time coherence information from the captured depth and color data.

The goal of our work is to present a unified system comprising of multiple Kinects to synchronously capture using a software-only acquisition setup and reconstruct spatio-temporally coherent dynamic 3D scene geometry from dynamic RGB-D data. Our system is highly scalable and can be extended to any number of cameras. We show that the data from our acquisition setup can be merged to reconstruct the dynamic 3D scene to a very good approximation of its real world counterpart. Our system is very low cost, and we only use easily available open source software solutions to acquire, register, and process the data. To our knowledge this is the first system, which shows that acquisition, and time-coherent 3D animation reconstruction is possible using multiple Kinects. In principle, any type and combination of RGB and depth cameras can be used for the acquisition. We chose Microsoft Kinect because it is a hybrid color (RGB) and depth camera system which provides both the color and depth information at the rate of 30 frames per second. Our

Manuscript received January 9, 2014; revised March 26, 2014.

The authors are with University of Sharjah, United Arab Emirates (e-mail: nahmed@sharjah.ac.ae, ijunejo@sharjah.ac.ae).

acquisition system can acquire synchronous streams of RGB-D data from multiple Microsoft Kinects.

Our system uses both depth and color information and extracts time coherence information from the dynamic three-dimensional content. The dynamic three-dimensional content is assumed to be in the form of a three-dimensional point cloud with color information associated with every point at every frame. We will show that we can obtain this information very easily from our acquisition setup that provides us not only the depth information of real world scene but also its color information. Our work is not limited to the data obtained by the Microsoft Kinect cameras but we will also show that our work is equally suitable for the three-dimensional content obtained using a traditional acquisition setup of multi-view color cameras.

II. Data Acquisition and Calibration

Our multi-view recording setup comprises of multiple Kinect cameras. We test our acquisition, using two, four and six cameras. For acquisition with six cameras, three cameras are placed on each side of the room (Fig. 1a). The four corner cameras are placed at an angle of 90 between them. In between on each side, two additional cameras are placed that make an angle of 45 degrees with their two adjacent cameras (shown in red and yellow in Fig. 1a). In principle, all Kinects emit the infrared laser at the same frequency, which is a potential source of problem when using multiple Kinects for simultaneous acquisition. The ideal angle between two Kinects should be 180 degrees for simultaneous acquisition without any interference. For our work, we deliberately ignore this interference issue as our aim is to test 360° acquisition and study the effect of the interference. Our intuition that missing information from one camera will be filled by one of the other cameras turned out to be correct, as shown by our results. Our particular placement of the cameras allows for capturing a dynamic object within an area of around 2m × 3m.

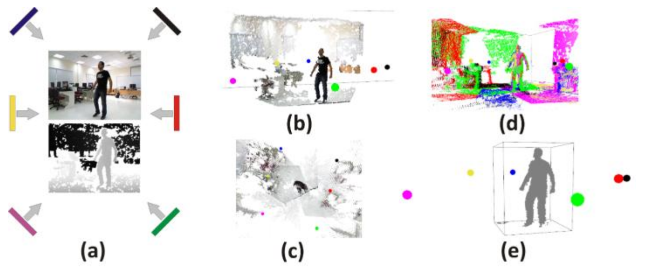

Fig. 1. Our system pipeline. (a) Six Kinects are used to acquire the RGB and depth images (only one frame from one camera is shown). (b) shows the 3D point cloud from one camera with the mapped RGB image. (c) Shows the top down view of six merged 3D point clouds. The alignment of the cameras after the global registration is shown in (d) using the color-coded points. The final segmented and filtered point cloud is shown in (e).

Each Kinect is connected to a dedicated machine comprising Intel Core i5 2.4 GHz with 4 GB of RAM running Windows 7 64 bit. We believe that this is not a big limitation as all comparable acquisition systems use a similar setup. We make use of *Open Kinect freenect Kinect* drivers and library for data acquisition [12], as it provides a wrapper to query for the depth and the RGB data using a synchronous interface [13] - a feature not provided by Microsoft's current SDK for Kinect. In general, all current Kinect SDKs provide an asynchronous interface where callback functions are invoked when sensor data is available. The synchronous interface manages a buffer where it holds the data and on query provides the depth or RGB data with their respective time stamp. This procedure introduces some gaps in the data, but for a multi-view synchronous capture, these gaps are desirable if all Kinects can query the data at the same time. All machines are clock-synchronized and record the data at same time, resulting in a sequence of depth and color images from all six cameras.

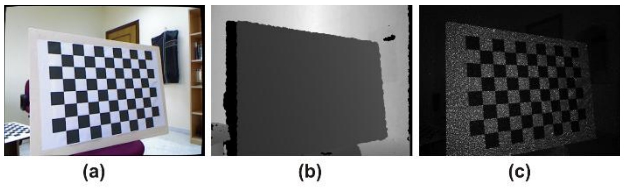

Fig. 2. Intrinsic camera calibration - Checkerboard as recorded from the (a) color camera, (b) depth camera, and (c) infrared sensor.

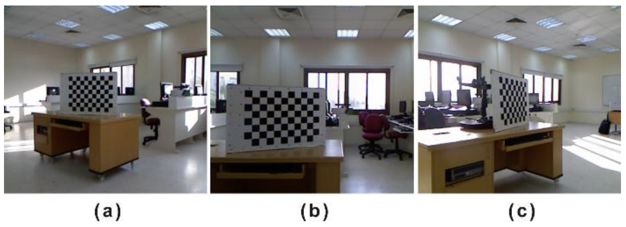

Fig. 3. Extrinsic calibration for global registration - checkerboard as recorded from three cameras. Corners from the checkerboard are used as some of the initial correspondences for the iterative closest point method for the global registration.

A multi-view acquisition system requires both local and global calibration. Local calibration provides camera specific parameters, or intrinsic parameters. On the other hand, the global calibration or extrinsic parameters provide the spatial mapping between the cameras. For a Microsoft Kinect, which has two sensors, there is an additional level of local calibration. In the first step, both the depth and color sensors have to be calibrated to estimate their intrinsic parameters, as shown in Fig. 2. Secondly, a mapping should be established between the depth and color sensors so that color data can be projected on the depth data. Finally, depth values are mapped to real-world distances in order to get 3D positions in a global coordinate system. The intrinsic parameters are obtained using Matlab Camera Calibration toolkit. We record a checkerboard from both color and infrared sensors to facilitate this calibration. To convert the depth data to meters we employ the method proposed by Nicolas Burrus. We use the Kinect RGB Demo software to do the full internal calibration. Using the internal calibration we obtain a 3D point cloud for each camera along with its mapping to the color data. An example of the 3D point cloud with depth to color mapping can be seen in Fig. 1.

The final step for getting a dynamic 3D point cloud is to merge all the cameras together in a global unified coordinate system. This global registration is an important step because without it each point cloud would be in its own coordinate frame. To achieve this global registration we first need to find out correspondences between different cameras. This is achieved by recording the checkerboard pattern at different locations for each pair of adjacent cameras, as shown in Fig.

3. The corners of the checkerboard seen in different views provide the correspondences between two cameras. These corners of the checkerboard are estimated using *OpenCV*, and in addition, we also find the correspondences using SIFT [14]. The correspondences are estimated in RGB space. The depth to RGB mapping is also obtain using correspondences between the point clouds. Once the correspondences between all adjacent cameras are found, any one camera is selected as a reference camera and the correspondences are used as the starting point for the iterative closest point algorithm to find the rotation and translation transformations that maps one point cloud to the other. This transformation is found for each of two adjacent pairs and all cameras are mapped to a unified global coordinate system, i.e. the coordinate frame of the reference camera. The global registration is a standard process and any relevant method can be applied for this step. In our work we used the Point Cloud Library (PCL) [15] because of its flexible data format (PCD) for storing point clouds.

Global registration gives a unique pair of rotation and translation transformations for each camera and it is applied to the corresponding depth data. The final result of the global registration is a 3D point cloud for each animation frame. Additionally, using the mapping between color and depth cameras, we also associate the color value to each depth point. Thus we obtain a dynamic 3D representation of a real world scene. However, this representation is not time coherent as each frame is independent of the other. Examples of a 3D point cloud from one of the cameras can be seen in Fig. 1b and 4a.

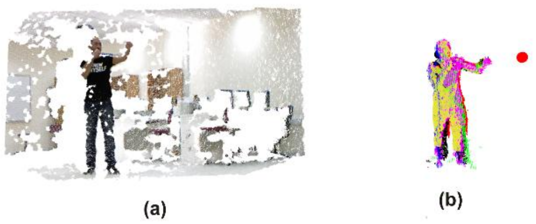

Fig. 4. One frame of the dynamic 3D point cloud with RGB mapping can be seen in (a). (b) shows the merged point clouds from all cameras after global registration and segmentation.

Before getting a dynamic 3D point cloud, we have to segment the scene so that a real-world actor can be separated from the background. This background subtraction is performed using the depth data. First the acquisition room is recorded without the human actor and later the depth information of the background is used to subtract the real-world actor from the background. The result of Global registration and segmentation can be seen in Fig. 1 and Fig. 4.

We use data provided by Ahmed *et al.* [3]. The data has a 3D visual hull representation at every time step and a corresponding color information. We extract the point cloud from the visual hull representation and also extract the time-coherent representation of dynamic 3D content from the data, as explained in the next section.

III. Spatio-Temporally Coherent 3D Animation

As explained in the previous section, the dynamic three-dimensional content obtained through either one or more Microsoft Kinects or a traditional multi-view video acquisition system completely lacks any temporal coherence. That is, there is no connectivity from one point cloud to the next for each consecutive frame of the video. Thus the data is not very useful in extracting any meaningful information about the scene other than simple visualization. It is not even visually pleasing, as the position of the points change so quickly from frame to frame that it distracts the viewer from the actual animation. We therefore propose a new method to extract spatio-temporal coherence information from this dynamic 3D point clouds using both geometric and color information. This coherence info will be found between two consecutive frames over the course the animation. Using the coherence information we aim at tracking a 3D point cloud throughout the entire animation.

In the first step, we reconstruct a coarse surface representation of the all 3D point clouds by fitting a plane to every 3D point x at frame i. This course surface representation allows us to use the first order normal of that point $N(xi)$, and the second order curvature $C(xi)$. In the second step we estimate SURF features [16] on the color data. Using the intrinsic mapping we find key 3D points that are associated with the SURF features in the color images $S(xi)$:

Finally, using these three local descriptors for each $x + i$ we define a matching function to find the mapping of each x_i to x_{i+1}:

$$M(x_i) = \alpha\left(1.0 - N\left(x_i.N(x_{i+1})\right) + \beta(\|C(x_i) - C(x_{i+1})\|) + \gamma(\|S(x_i) - S(x_{i+1})\|)\right) \tag{1}$$

where x_{i+1} is the 3D point at the frame $i + 1$, which is used to evaluate the eq (1). $M(x_i)$ is the matching distance, 1:0 $N(x_i)$:$N(x_{i+1})$ is the angular difference in orientation, with the similar orientation resulting in a smaller value. $\|C(x_i)\ C(x_{i+1})\|$ is the absolute difference of curvature between the points. $\|S(x_i)\ S(x_{i+1})\|$ is the absolute difference in the distance to the two nearest SURF feature. The three parameters α, β, and γ are weighting parameters. For our method we set $\alpha = 0{:}3$, $\beta = 0{:}25$, and $\gamma = 0{:}45$. These values are chosen according the weight of each feature in terms of its influence. The SURF feature localizes the position; therefore it gets the maximum weight. Under the assumption that the motion of the object is small over two consecutive frame, and the deformations are isometric, the local orientation of the normal should not change. The orientation therefore gets the second highest weight. Finally, curvature is an important property for the matching, but due to inherent noise in Kinect depth data, it is not as reliable compared to SURF and orientation, therefore it is assigned a lower weight. We use this matching function to find the mapping of each x_i to a point at $i + 1$. A map of x_i is the point that minimizes the matching function $M(x_i)$.

IV. Results

To test our method, we record a number of sequences using different number of cameras. Each sequence is between 100 200 frames long. Additionally, we use data from Ahmed *et al.* [3], which is captured using eight color cameras with an acquisition system synchronized by the dedicated hardware. The sequences range from a simple walking motion to the fast boxing motion. Results from our acquisition system can be seen in Fig. 1 and Fig. 5. It can be seen that the dynamic depth maps are well aligned and the RGB image are also mapped accurately to the point cloud. As can be seen in Fig. 5a, there is no connectivity between the two frames, e.g. feet of the actor have different shape. Using the estimated time coherence, we can visualize the animation with a single 3D point cloud tracked over the sequence, which can be seen in Fig. 5b. It can be observed that our method can reliably track the point cloud from one frame to the next and consequently over the course of the animation. This results in generating a 3D animation that is temporally smooth.

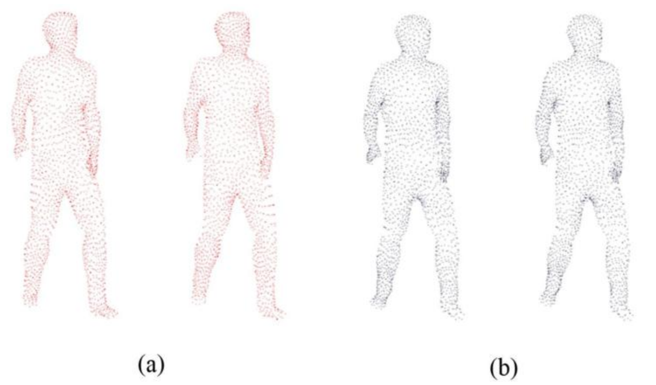

Fig. 5(a) Shows two consecutive frames from a dynamic 3D point cloud without any time coherence. (b) Show same two frames tracked using the time coherence. For example, at the feet, the point cloud changes dramatically from one frame to the next without the time coherence, whereas in (b) the point cloud remains consistent.

Our method is subjected to some limitations. Most notably, we only employ two nearest SURF features in our matching function (cf. eq. (1)). We cannot use more SURF features, because the nearest function can map points at different body parts, which will result in the incorrect animation. Additionally, due to high noise in Kinect's depth data the curvature is not the most reliable descriptor. One can improve it by first smoothing the surface and then estimating the curvature. We would like to explore this in the future work.

Despite the limitations, we show that it is possible to reconstruct spatio-temporally coherent 3D animation of a real-world object from RGB-D data from multiple Kinects.

V. Conclusion

We presented a method to acquire synchronized RGB-D data from multiple Kinects and reconstruct spatio-temporally coherent animation from that data. Microsoft Kinect provides both color and depth information of a scene. We combine multiple Kinect cameras and capture a complete three-dimensional dynamic scene. Our system is scalable and our spatio-temporally coherent re-construction method can be applied to any three-dimensional representation of the data, as long it is comprised of 3D point clouds with color information. We demonstrated this by applying our method on the data obtained using multiple acquisition setups, and in future we would like to extend our work to increase the robustness of our tracking method and explore the possibilities in the area of scene analysis and dynamic surface reconstruction.

References

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

[2] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun, "Performance capture from sparse multi-view video," *ACM Trans. Graph.*, vol. 27, no. 3, 2008.

[3] N. Ahmed, C. Theobalt, C. Rossl, S. Thrun, and H.-P. Seidel, "Dense correspondence finding for parametrization-free animation reconstruction from video," *CVPR*, 2008.

[4] MICROSOFT. (November 2010). Kinect for microsoft windows and xbox 360. [Online]. Available: http://www.kinectforwindows.org/

[5] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz, and M. A. Magnor, "Markerless motion capture using multiple color-depth sensors," *VMV*, pp. 317–324, 2011.

[6] A. Weiss, D. Hirshberg, and M. J. Black, "Home 3d body scans from noisy image and range data," *ICCV*, 2011.

[7] A. Baak, M. Muller, G. Bharaj, H.-P. Seidel, and C. Theobalt, "A data-driven approach for real-time full body pose reconstruction from a depth camera," *ICCV*, 2011.

[8] Y. M. Kim, C. Theobalt, J. Diebel, J. Kosecka, B. Micusik, and S. Thrun, "Multi-view image and tof sensor fusion for dense 3d reconstruction," in *Proc. IEEE Workshop on 3-D Digital Imaging and Modeling (3DIM)*, Kyoto, Japan, 2009, pp. 1542–1549.

[9] V. Castaneda, D. Mateus, and N. Navab, "Stereo time-of-flight," *ICCV*, 2011.

[10] Y. M. Kim, D. Chan, C. Theobalt, and S. Thrun, "Design and calibration of a multi-view tof sensor fusion system," in *Proc. IEEE CVPR Workshop on Time-of-flight Computer Vision*, 2008.

[11] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall, and H.-P. Seidel, "Markerless motion capture with unsynchronized moving cameras," *CVPR*, 2009.

[12] Open Kinect. Open source libraries for Microsoft kinect. [Online]. Available: http://www.openkinect.org/."

[13] Openkinect C Sync Wrapper. [Online]. Available: http://www.openkinect.org/wiki/c sync wrapper

[14] D. G. Lowe, "Object recognition from local scale-invariant features," *ICCV*, pp. 1150–1157, 1999.

[15] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," *ICRA*, 2011.

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

Naveed Ahmed received his Ph.D. in computer science from the University of Saarland (Max-Planck-Institute for Informatics), Germany in 2009. He worked as a research and development engineer at Autodesk in Cambridge, UK for two years. He is currently working as an assistant professor at the Department of Computer Science, University of Sharjah. His research interests include 3D animation and dynamic scene reconstruction and multi-view video based modeling and rendering.

Imran N. Junejo received his Ph.D. in computer science from University of Central Florida, U.S.A in 2007. After a post-doc at INRIA-Rennes, he joined the Department of Computer Sciences, University of Sharjah where he is currently working as an assistant professor. His current focus of research is human action recognition from arbitrary views. Other areas of research interests include: camera calibration, metrology, path modeling, video surveillance, scene understanding and event detection.