A System for 3D Video Acquisition and Spatio-Temporally Coherent 3D Animation Reconstruction using Multiple RGB-D Cameras

Naveed Ahmed and Imran Nazir Junejo

University of Sharjah
nahmed@sharjah.ac.ae, ijunejo@sharjah.ac.ae

Abstract

We present a system for acquiring synchronized multi-view color and depth (RGB-D) data using multiple off-the-shelf Microsoft Kinect and a new method for reconstructing spatio-temporally coherent 3D animation from time-varying dynamic RGB-D data. Our acquisition system is independent of any specific hardware component for the synchronization of the camera system. We show that the data acquired by our framework can be synchronously registered in a global coordinate system and then can be used to reconstruct the 3D animation of a dynamic scene. The main benefit of our work is that instead of relying on expensive multi-view video capture setups, multiple low cost Microsoft Kinect sensors can capture both the image and the depth data to do a 360^o reconstruction of a dynamic scene. We also present a new algorithm for tracking dynamic three-dimensional point cloud data that can be used to reconstruct a time-coherent representation of a 3D animation without using any template model or a-prior assumption about the underlying surface. We show that despite some limitations imposed by the hardware for the synchronous acquisition of the data we can get reasonably good reconstruction of the animated 3D geometry, which can be used in a number of applications.

Keywords: *Multi-view video acquisition, 3D and Free-viewpoint video, Dynamic scene reconstruction, RGB-D data acquisition*

1. Introduction

Spatio-Temporally coherent time-varying dynamic scene geometry has been employed in a number of applications. It can be used for 3D animation in digital entertainment productions, electronic games, 3D television, motion analysis, gesture recognition, *etc.* 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 has no 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. This synchronization is typically achieved by means of a hardware-based camera trigger, which acts as an external synchronizer. 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, 2, 3, 4, 5]. 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.

One of the earlier works in this area was presented by Carranza, *et al.*, [1], who used eight multi-view recordings to reconstruct the motion and shape of a moving subject and applied it in the area of free-viewpoint video reconstruction. Theobalt et al. [2] extended this work so that in addition to capturing the shape and motion they also captured surface reflectance properties of a dynamic object. Starck, *et al.*, [5] presented a high quality surface reconstruction method that could capture detailed moving geometry from multi-view video recordings. Later de Aguiar, *et al.*, [3] and Vlasic, *et al.*, [4] presented new method for reconstructing really high quality of dynamic scene using multi-view video recordings. Both of their methods first obtained the shape of the real world object using a laser scanner and then deformed the shape to reconstruct the 3D animation. Ahmed, *et al.*, [6] presented a method of dynamic scene reconstruction with time coherent information without the use of any template geometry, but unlike our method they did not explicitly include multiple matching criteria for extracting time coherence in their method.

With the arrival of depth sensors, especially low cost Microsoft Kinect [7], there has been a wave of interest in incorporating them in a number of research areas including the reconstruction of dynamic scene geometry. For example, one or more depth sensors are employed in 3D shape scanning and dense 3D reconstruction of static objects; pose, motion and 3D shape estimation [8, 9, 10]. 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. Recently, two methods have been presented to capture data using multiple depth sensors. Kim, *et al.*, [11] presented a system for the fusion and calibration of RGB and depth sensors. Their system uses a dedicated hardware setup for the synchronization of the color and depth cameras. More recently, Berger, *et al.*, [8] presented a method to capture motion using four Kinects but without any active synchronization between the sensors. These methods do not try to extract any spatial or temporal coherence information from the acquired dynamic data.

The goal of our work is to present a unified system comprising of multiple Kinects to perform synchronous capture using a software-only acquisition setup, and to 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. Our work is an extension of the acquisition system presented by Ahmed, *et al.*, [12]. In principle, any type and any 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 30 frames per second. Our acquisition system can acquire synchronous streams of RGB-D data from multiple Microsoft Kinects.

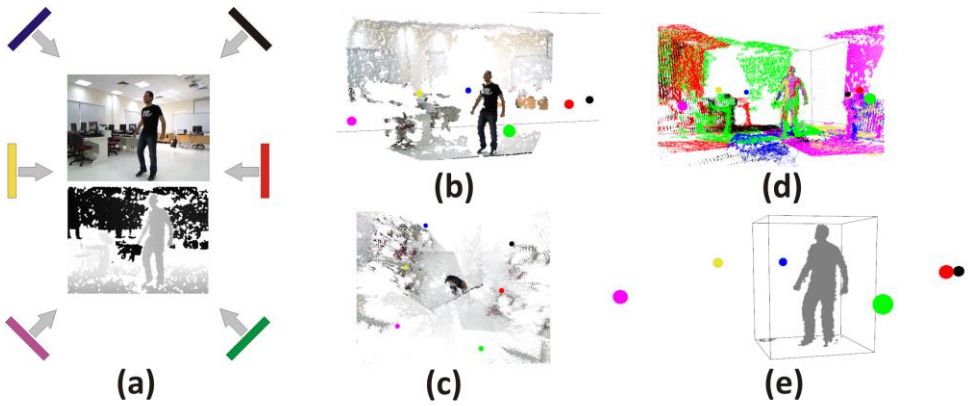


Figure 1. Our system pipeline: (a) Six Kinects are used to acquire the RGB and depth images (only one frame from one camera is shown). (b) Shows the 3D point cloud from one camera with the mapped RGB image. (c) Shows the top down view of six merged 3D point clouds. The alignment of the cameras after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

The acquired multi-view RGB and depth data is not temporally coherent as each frame is independent of the other. We present a system, which can use both depth and color information and extract 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. Main benefit of using Microsoft Kinect cameras is that unlike the requirement of employing eight or more color cameras, only one Microsoft Kinect camera can be employed to get meaningful depth and color information of a real world dynamic scene. The main contributions of our work are:

1. Acquisition of data using one or more Microsoft Kinect camera and organize it in a form of a three-dimensional dynamic point cloud with the color information.

2. A new method of finding time coherent information from three-dimensional point cloud data using both color and depth information. This data can either be acquired from Microsoft Kinect cameras as described in step 1 or a traditional setup of multi-view video acquisition using color cameras.

2. Related Work

We capitalize on previous research in a number of areas, but primarily our work derives from the areas of multi-view video (MVV) acquisition, 3D and free-viewpoint video, static and dynamic surface reconstruction from color and depth cameras.

MVV data has been used in a number of applications. In one of pioneering works on free-viewpoint video, Carranza, *et al.*, [1] used eight color cameras to capture the shape, appearance and motion of a real-world actor. Starck, *et al.*, [5] also used a high-definition acquisition system comprising of eight cameras to capture the moving actor. They were also able to capture the high-level cloth deformations on the actor. MVV acquisition is not limited to low number of cameras, rather Debevec, *et al.*, [13, 14, 15] reconstructed a number of iterations of the so-called "light-stage" which used a large number of cameras to capture the static and dynamic objects under static and varying lighting conditions. The work on free-viewpoint video by Carranza, *et al.*, was extended by Theobalt, *et al.*, [2] where, in addition to eight high resolution color cameras, they used to calibrated spot lights to not only acquire the shape, motion and appearance but also the surface reflectance properties of a moving person. The estimation of dynamic surface reflectance allowed rendering the reconstructed 3D animation in a virtual environment having starkly different lighting condition compared to the recording environment.

A number of methods have been proposed to reconstruct spatio-temporally consistent 3D animation from MVV data. De Aguiar, *et al.*, [3] presented a method to reconstruct high quality spatio-temporal reconstruction of dynamic objects by means of a deformation based method. They first obtained a high quality template scan of the real-world person that was deformed over the course of the animation by means of an optimization method that ensured that the deformed model is consistent with the input MVV data. Similar approach was adopted by Vlasic, *et al.*, [4] where the skeleton-based deformation was employed to track the high quality template mesh over the animation. On the contrary, Ahmed, *et al.*, [6] first reconstructed spatio-temporally incoherent visual hulls from MVV data for each frame of MVV data. They tracked the first visual hull over the whole sequence by means of a dense correspondence finding method that maps one visual hull to the next. None of these methods employed depth cameras for the acquisition, and unlike this method, our work does not rely on any template data or 3D surface representation for reconstructing spatio-temporally coherent 3D animation.

With the advent of low cost depth sensors, especially Microsoft Kinect [7], there has been a wave of interest in incorporating depth sensors for the acquiring 3D static and dynamic content. One of the main benefits of using Kinect is that provides both color and depth data simultaneously at 30 frames per second. Earlier works relied only on the color data where correspondences between cameras had to be used to reconstruct the depth information, now directly provided by Kinect. Ahmed, *et al.*, [6] reconstructed time-varying visual hulls by similar means. It is not necessary to use Kinect for acquiring the depth information as it can also be obtained from other types of sensors, *e.g.*, Time of Flight (ToF) sensors [11].

One or more depth sensors have been employed in a number of applications to reconstruct a three-dimensional representation or static and dynamic objects. Kim, *et al.*, [16] presented a multi-view image and depth sensor fusion system to reconstruct 3D scene geometry. Castaneda, *et al.*, [17] used two depth sensors for stereo-ToF acquisition of a static scene. Microsoft Kinect camera was employed by Weiss, *et al.*, [9] for human shape reconstruction. Their method combines low-resolution image silhouettes with coarse range data to estimate a parametric model of the body. Similarly, Baak, *et al.*, [10] employed a single depth camera in their pose estimation framework for tracking full-body motions. Pose estimation from a single depth sensor has been a

hallmark of Kinect as an input device, and one of the seminal works in this area was presented by Girshick, *et al.*, [18].

The low cost of Microsoft Kinect, coupled with the benefits of acquiring depth information directly from the sensor, has led to the use of multiple depth sensors in an acquisition system. In one of the pioneering works, Kim, *et al.*, [11] 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.*, [8] 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.*, [19]. 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 [11, 8] do not try to extract any time coherence information from the captured depth and color data.

In this paper we present a software-synchronized multi-view acquisition system using multiple Microsoft Kinects. We show that using just of-the-shelf equipment, it is possible to create a highly scalable low-cost multi-view acquisition system. We also present a new method that can reconstruct spatio-temporally coherent 3D video from a dynamic 3D representation that can be acquired by our or any comparable MVV acquisition system.

3. Data Acquisition

Our multi-view recording setup is comprised of multiple Kinect cameras. We tested our acquisition, using two, four and six cameras. For the acquisition with six cameras, three cameras are placed on each side of the room (Figure 1(a)). The four corner cameras are placed with the angle of 90 degrees between them. In between on each side, two additional cameras are placed that make the angle of 45 degrees with their two adjacent cameras (shown in red and yellow in Figure 1(a). 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. Ideal angle between two Kinects would be 180 degrees for simultaneous acquisition without any interference. For our work, we deliberately ignore the interference issue because our aim was to test 360 acquisitions and observe how much problem is caused by 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. The placement of our cameras allows us to capture a dynamic object within an area of around 2m x 3m.

Figure 2. One Microsoft Kinect (circled) as used in our acquisition system

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 (Figure 2). We believe that this is not a big limitation as all comparable acquisition systems use a similar setup. We make use of OpenKinect freenect Kinect drivers and library for data acquisition [20]. This library is chosen because it provides a wrapper to query for the depth and RGB data using a synchronous interface [21]. This facility is not even 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 provides the depth or RGB data on query 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.

The Kinect camera provides 640x480 pixels of RGB and depth data at the frame rate of 30 fps. Unfortunately this data is not hardware synchronized and there is a lag of 16 ms between depth and RGB data. If data is acquired from each Kinect without any consideration to the acquisition from other cameras and assumed to be approximately synchronous, the video streams will start to drift temporally very quickly. We circumvent this issue by employing the following steps: First, our hardware setup is identical for each Kinect that assures that the data transfer and processing will take place at the same speed. Before the acquisition step, each machine is independently synchronized to a web-server (http://www.time.is) multiple times so that their internal clocks are synchronized. This web-server also indicates if the computer clock is running faster or slower than the web-server. The machines are synchronized till they all report the exact time. Thereafter all machines are programmed to start recording at the exact same time. This is done using our software interface, which accepts hour, minute and second as the starting time. We also provide the number of frames to record. Since we know the frame rate of a Kinect, total number of frames to record, starting time of the recording and given that we need to query both RGB and depth frames, exact clock ticks are calculated for querying each frame in advance. Each machine then uses the synchronous interface at the pre-calculated time to query for depth and RGB data alternatively. To minimize the I/O overhead we store both RGB and depth data in a buffer and once the recording is finished, data is written to the disk. We currently do not use the time stamp information for any post-recording synchronization but after comparing the timestamps for the 12 images (6 RGB and 6 depth)

obtained for each frame, we did not observe any noticeable differences. Since all machines are querying the data at the exact time, the acquisition drift is kept under check.

4. System Geometry & Calibration

The projection of a 3D scene point $\mathbf{X} \sim [X\ Y\ Z\ 1]^T$ onto a point in the image plane $\mathbf{x} \sim [x\ y\ z\ 1]^T$, for a perspective camera can be modeled by the central projection equation:

$$s\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & t \end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & t \end{bmatrix}\underbrace{\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}}_{\hat{\mathbf{x}}} \quad (1)$$

where ~ indicates equality up to a non-zero scale factor. Here $\mathbf{R} = \mathbf{R}_x\,\mathbf{R}_y\,\mathbf{R}_z = [r_1\ r_2\ r_3]$ is the rotation matrix. The upper triangular 3x3 matrix **K** encodes the five intrinsic camera parameters: focal length *f*, aspect ratio $= f_y/f_x$, skew γ and the principal point at (u_o, u_1). As argued by [22, 23], it is safe to assume γ= 0.

The aim of camera calibration is to determine the calibration matrix **K**. Instead of directly determining **K**, it is common practice (for *e.g.*, [24]) to compute the symmetric matrix $\omega = \mathbf{K}^{-T}\mathbf{K}^{-1}$ referred to as Image of the Absolute Conic (**IAC**). **IAC** is then decomposed uniquely using the Cholesky Decomposition to obtain **K**.

For our system, we take multiples pictures of a planar checkerboard pattern, moved to different location in each frame in order to avoid the degenerate configurations, as shown in Figure 3. The motion for the checkerboard need not be known. The model plane contains a pattern of 11x8 squares, so there are 352 corners. The size of the pattern is 32cm x 25cm.

Assuming the model plane as on Z = 0 of the world coordinate system, eq. (1) can be arranged as:

$$\mathbf{x} \sim \underbrace{\mathbf{K}\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}}_{\mathbf{P}}\mathbf{X}, \quad \mathbf{K} = \begin{bmatrix} f_x & \gamma & u_o \\ 0 & f_y & v_o \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

or

$$s\tilde{\mathbf{x}} = \mathbf{H}\tilde{\mathbf{X}}$$

where *s* is the arbitrary scale factor with $\mathbf{H} = \mathbf{K}[\mathbf{r}_1\ \mathbf{r}_2\ t]$ which can be easily obtained from the images of the model plane. Since, **H** is a 3x3 matrix, defined up to a scale factor, we have from eq. (2):

$$[\mathbf{h}_1\ \mathbf{h}_2\ \mathbf{h}_3] = s'\mathbf{K}[\mathbf{r}_1\ \mathbf{r}_2\ t] \quad (3)$$

where s' is an arbitrary scalar, and by using the orthogonality of $\mathbf{r}_1$ and $\mathbf{r}_2$, we get the two basic constraints on the intrinsic parameters:

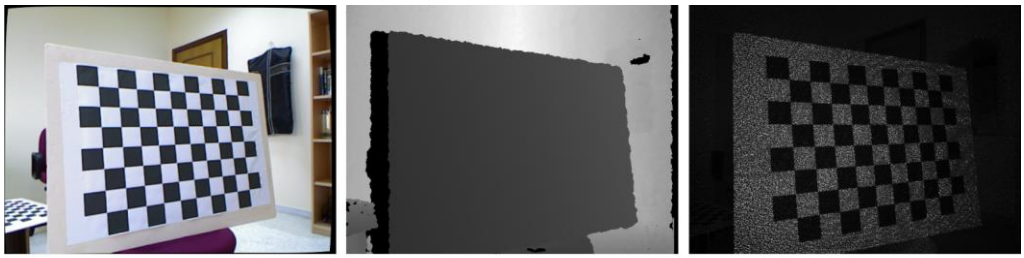


Figure 3. Intrinsic camera calibration - Checkerboard as recorded from the (a) color camera, (b) depth camera, and (c) infrared sensor

$$\mathbf{h}_1^T \omega \mathbf{h}_2 = 0 \quad (4)$$

$$\mathbf{h}_1^T \omega \mathbf{h}_1 = \mathbf{h}_2^T \omega \mathbf{h}_2 \quad (5)$$

A homography has 8 degrees of freedom, and there are six 6 extrinsic parameters (3 for rotation and 3 for translation), we can only obtain two constraints for the intrinsic parameters. We refer the reader to [25], which describes how equations (4) and (5) can be used to extract both the intrinsic and the extrinsic parameters. The parameters obtained at this stage are used as initializing values, along with the radial distortion parameters [24], to compute the Maximum Likelihood Estimate (MSE) of all the parameters.

The parameters for the depth camera are estimate in a similar fashion. Since the pattern on the checkerboard is not discernible by the depth sensor, we selected only the corners of the checkerboard manually. Standard stereo calibration technique is used to learn the transformation between the depth sensor and the RGB camera [24] (Figure 3).

5. Global Registration and Segmentation

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 find out the correspondences between the different cameras. This is achieved by recording the checkerboard pattern at different locations for each pair of adjacent camera as shown in Figure 4. The corners of the checkerboard provide the correspondences between two cameras are obtained using OpenCV, and in addition we also find the correspondences using SIFT [26]. The correspondences are estimated in RGB space and from depth to RGB mapping we obtain the correspondences between the point clouds. Once the correspondences between all adjacent cameras are found, 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, which is 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) [27] because of its flexible data format (PCD) for storing point clouds.

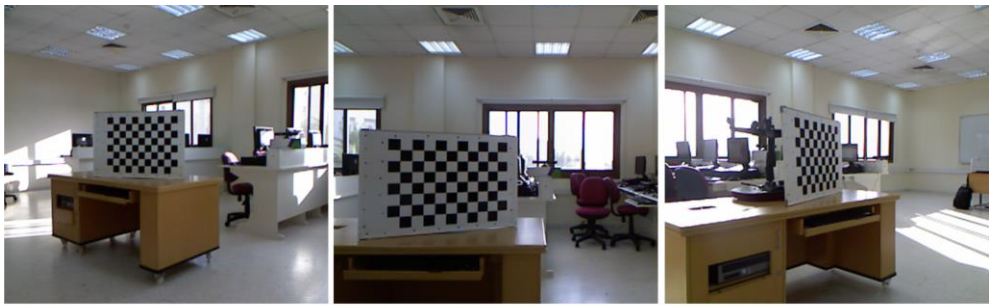


Figure 4. Extrinsic calibration for Global Registration - Checkerboard as recorded from three cameras. Corners from the checkerboard are used as some of the initial correspondences for the Iterative Closest Point method for the Global Registration

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 frame of the animation. Additionally, using the mapping between color and depth cameras, we also associate the color value with each point. Thus we obtain a dynamic 3D representation of a real world scene. This representation is not time coherent because each frame is independent of the other. Examples of a 3D point cloud from one of the cameras can be seen in Figure 1(b) and 5(a).

Final step before getting a dynamic 3D point cloud is to segment the scene so that the real-world actor can be separated from the background. We do the background subtraction 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 Figure 1 and Figure 5.

We also use data from Ahmed, *et al.*, [6], which have 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.

6. 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. 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. Even it is not 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.

Figure 5. 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

The first step in this method is to extract the orientation or the normal at every point of all 3D point clouds. To approximate the normal, we find the normal to the plane that is tangent to the surface at that point. Since we have a point cloud, there is no actual surface, rather we choose 10 nearest points to fit a plane and then find normal to that plane to find the orientation for each point. We treat normal at each point as one of the feature of that point. Henceforth, for a 3D point x at frame i, the normal of that point will be referred as $\mathbf{N}(x_i)$. One way to match one point cloud to the next would be to just match the two points that have similar normals. This will mostly hold true as our animation is not very fast and at each frame we do not have a strong motion. But this relies on the assumption that each point has a completely unique orientation and therefore mapping from one frame to the next is trivial. In practice this is a false assumption because for every three dimensional object there are planar regions where the orientation of all the points are the same and only using the orientation information to match two 3D point clouds can never work because of the ambiguity in one to many mapping from one frame to the next.

To circumvent this ambiguity we propose to use another set of features using the color information. Since we know the mapping of the depth data to the color data, we can extract the color info at each point in the 3D point cloud. Thus, when matching two point clouds, in addition to the orientation, the color information is also matched to ensure that there is no obvious incorrect mapping of the point clouds. Henceforth, for a 3D point x at frame i, the color of that point will be referred as $\mathbf{C}(x_i)$.

The color and orientation information can give us partial matching but it is still ambiguous as the actor can wear clothes of a uniform color. Therefore we introduce two more criteria to make sure that a point is not mapped to another point of the same color and the same orientation but at farther distance. Our data has two notions of distance: one obvious notion is the three-space distance between two three-dimensional points. This can be trivially found by finding the 3D Euclidean distance between two points. For a 3D point x at frame i the Euclidean distance of that point with the other 3D point at frame $i + 1$ will be referred as $\mathbf{D}(x_i)$. The second novel notion of the distance which is one of the major contributions of our algorithm is to use SIFT [26] to find out the feature points in the 3D point cloud. SIFT is one of most well knows feature descriptors in the image space which is invariant under affine transformation and varying lighting conditions. Using SIFT we find feature points in the color image and consequently using the mapping between depth and RGB images, we find the feature 3D points for

each frame. The SIFT matching is then used to find out the correspondences between the feature points at each frame. Once we have feature point matches identified, we associate a distance to each point with respect to its nearest feature point. For a 3D point *x* at frame *i* the distance to its nearest feature point will be referred as $\mathbf{F}(x_i)$.

Assuming we are finding a match between two consecutive frames, *e.g.*, *i* and *i + 1*, we find the match for every 3D point *x* at the frame *i* with the 3D points at the frame *i + 1* using the following matching function:

$$\mathbf{M}(x_i) = \alpha(1.0 - \mathbf{N}(x_i).\mathbf{N}(x_{i+1})) + \beta(\|\mathbf{C}(x_i) - \mathbf{C}(x_{i+1})\|) + \gamma(\|\mathbf{F}(x_i) - \mathbf{F}(x_{i+1})\|) + \delta\mathbf{D}(x_i) \quad (6)$$

Where x_i+1 is the 3D point at the frame $i+1$, which is used to evaluate the equation 6. $\mathbf{M}(x_i)$ is the matching distance, 1.0-$\mathbf{N}(x_i)$-$\mathbf{N}(x_{i+1})$ is the angular difference in orientation, with the similar orientation resulting in a smaller value. $\|\mathbf{C}(x_i)$-$\mathbf{C}(x_{i+1})\|$ is the absolute difference of color components between (R, G, B) components of two 3D points. $\|\mathbf{F}(x_i)$-$\mathbf{F}(x_{i+1})\|$ is the absolute difference in the distance to the nearest feature point and $\mathbf{D}(x_i)$ is the 3D Euclidean distance between x_i and x_{i+1}. The four parameters $\alpha, \beta, \gamma, \delta$ are weighting parameters resulting in a convex combination of four terms, i.e. their sum is equal to 1 and their value is between 0 and 1. For our method we set $\alpha = 0.25, \beta = 0.2, \gamma = 0.5, \delta = 0.05$. These values are chosen based on the reliability criteria for each of the term. We give most weight to the difference to the nearest feature point because it is directly derived from SIFT and has a higher degree of accuracy. Least weight is chosen for $\mathbf{D}(x_i)$ because in principal we cannot penalize the difference in 3D Euclidean position because the change in position is a fundamental property of an animation. This term is only used to preserve the drift and avoid the local minima in case multiple points at frame *i + 1* match the feature distance, orientation and the color. We choose the matching point x_{i+1} as the one with the minimum value of the convex combination. If two points result in the same value of $\mathbf{M}(x_i)$, then the point with smaller $\|\mathbf{F}(x_i)$-$\mathbf{F}(x_{i+1})\|$ is chosen as the matching point. In the unlikely case of same values for $\mathbf{M}(x_i)$ and $\|\mathbf{F}(x_i)$-$\mathbf{F}(x_{i+1})\|$, 1.0-$\mathbf{N}(x_i)$-$\mathbf{N}(x_{i+1})$ is used to find the matching point, followed by $\|\mathbf{C}(x_i)$-$\mathbf{C}(x_{i+1})\|$ and $\mathbf{D}(x_i)$.

7. Results

To test our software-synchronized multi-view RGB acquisition system, we record a number of sequences using different number of cameras. Each sequence is between 100 – 200 frames long. The sequences range from a simple walking motion to the fast boxing motion. Results from our acquisition system can be seen in Figure 1, 6(right) and also in the accompanying video [28]. 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 shown in the figure and the video, our method is able to reconstruct a full 360^o 3D animation of even the faster motion to a great degree of accuracy. We do not observe any drift between the cameras even though the acquisition setup is not hardware synchronized. We validated our synchronous capture approach by deliberately recording a sequence in which one camera was set to record with the standard asynchronous acquisition interface. This resulted in a temporal misalignment as can be seen in Figure 6(left). Our results and validation show that our system is capable of synchronous capture of RGB-D data from multiple Kinects, which can be used for 3D animation reconstruction.

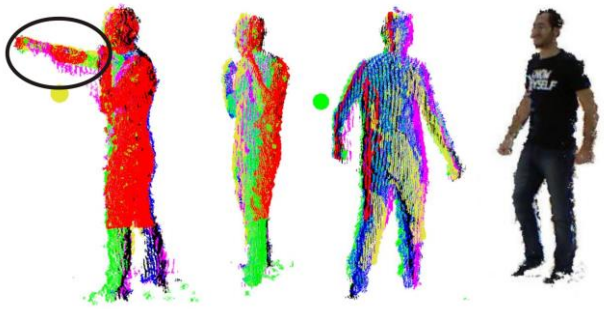


Figure 6. Image on the left shows the acquisition drift (circled) if the software synchronization is not employed. Next two images show the result of synchronous acquisition with the alignment of the dynamic point clouds. Last image shows the fusion of both RGB and depth data

Our acquisition system currently requires a dedicated machine for each Kinect. Our acquisition uses machines with Intel Core i5 2.4 GHz with 4 GB of RAM running Windows 7 64 bit. We believe that this is not a significant limitation of our acquisition system because most of the comparable acquisition systems use similar arrangement. Using Microsoft's current SDK for Kinect it is possible to connect multiple Kinects to a single machine albeit using different USB hubs, but we have not yet tested this setup. If multiple Kinects are attached to a single machine then a new study would be required to study the impact of processing cost and other input / output overheads. We believe our acquisition system provides a good solution in terms of robustness, efficiency and scalability, as a new camera can easily be added to the system without compromising the acquisition as long as it is connected to a dedicated machine.

To test our spatio-temporal 3D animation reconstruction method we use two types of data sets. The first data set is recorded through our software-synchronized acquisition system. We also use data from Ahmed, *et al.*, [6], which is captured using eight color cameras with an acquisition system synchronized by the dedicated hardware. Figure 7(a), qualitatively shows two consecutive frames without time coherence, whereas Figure 7(b) shows the same two frames with time coherence. As can be seen in Figure 7(a) there is no connectivity between the two frames, *e.g.*, feet of the actor have different shape. Using time coherence we can visualize the animation with a single 3D point cloud tracked over the sequence, which can be seen in Figure 7(b). More results can be seen in the accompanying video [28]. 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.

In order to quantitatively validated the proposed method, we perform 3D animation reconstruction while measuring the silhouette-based overlap and also by comparing the bounding box between the spatio-temporally consistent animation and the non-coherent animation. For the silhouette based overlap, we render the spatio-temporally consistent point cloud from the viewpoint of one of the input cameras. Once a 3D point cloud is projected onto a 2D image plane, calculating the silhouette of projected 2D points is trivially limited to finding their convex hull. We render both the original non-coherent 3D point cloud and the spatio-temporally coherent 3D point cloud from the same camera view and extract their silhouettes. We overlap the two silhouettes and count the number of pixels that do not overlap for each frame. On average we found that our spatio-temporally coherent 3D animation differs only by 2.8% in terms of silhouette overlap comparison. The graph of silhouette based

overlap error can be seen in Figure 8. It can be seen that even though the error increases with time, which is expected for every tracking based algorithm it remains really low and does not deter the quality of our 3D animation that can be seen in the accompanying video [28]. For the bounding box based comparison, we compute the bounding box for the non-coherent 3D point cloud and count the 3D points in the spatio-temporally coherent point cloud that are not inside this bounding box. This measure also provides a good quantitative analysis in analyzing the goodness of our tracking algorithm and its spatio- temporal consistency. It can be seen in Figure 9, that on average the bounding box consistency remains less than 2% of the non-coherent animation.

Our method is subject to some limitations. Most notably, we only employ one feature point in our matching function (eq. 6). This feature point is the nearest in terms of 3D Euclidean distance under whereas ideally one should look for more than one nearest feature points relative to the geodesic distance, as shown by Ahmed, *et al.*, [6]. We circumvent this issue by only using only one nearest feature point in terms of 3D Euclidean distance, which given the high number of feature points does not pose any issues for our method. Other approach would be to find the body segments and use multiple feature points from the nearest segments. Additionally our method relies on empirical justification for the steps that require heuristics, e.g. the choice of the values of coefficients in the objective function (eq. 6). Employing machine learning techniques to estimate proper values of parameters may well circumvent these heuristics. In the future work, we would like to explore this area of research.

Despite the limitations, we show that using multiple Microsoft Kinect cameras it is possible to capture synchronized dynamic 3D point clouds with the color information. We also show that given a dynamic three-dimensional content representation in the form of dynamic 3D point clouds with color information, it is possible to reconstruct spatio-temporally coherent 3D animation of a real-world object.

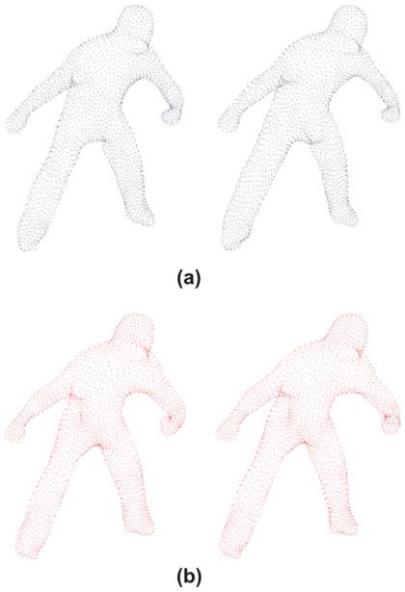


Figure 7. (a) Shows two consecutive frames from a dynamic 3D point cloud without any time coherence. (b) Shows same two frames tracked using the time coherence. For example, at the feet, the point cloud changes dramatically from one frame to the next without the time coherence, whereas in (b) the point cloud remains consistent

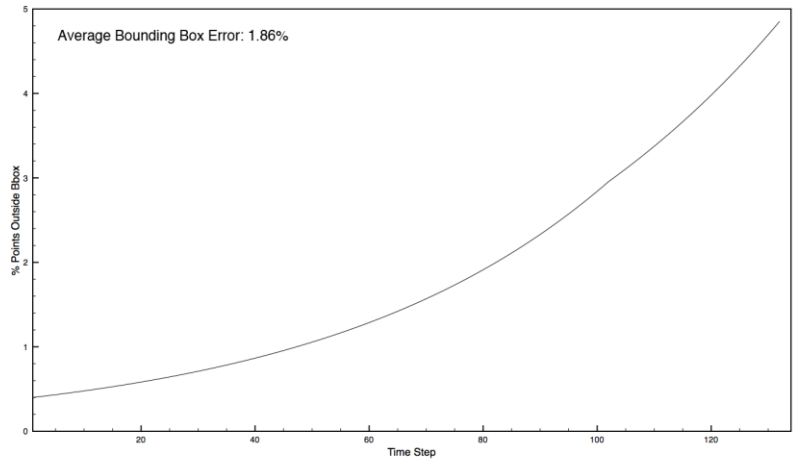


Figure 8. The silhouettes overlap error measure for each time step of the spatiotemporal coherent animation

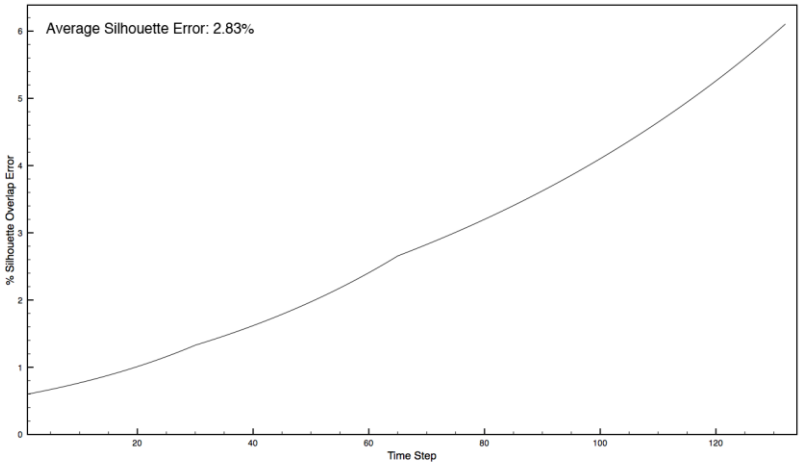


Figure 9. The bounding box error measure for each time step of the spatio-temporal coherent animation

8. Conclusion

We presented (1) a system for software-based synchronized 3D video acquisition and (2) a method for spatio-temporally coherent 3D animation reconstruction of a real-world scene from the dynamic RGB-D data. We showed that such a representation could be reconstructed using one or more Microsoft Kinect cameras. 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 can be used with data obtained from any number of cameras. Our spatio-temporally coherent reconstruction 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.

Acknowledgements

This work is supported by the Seed Grant for the new researchers courtesy of Department of Graduate Studies, University of Sharjah, United Arab Emirates.

References

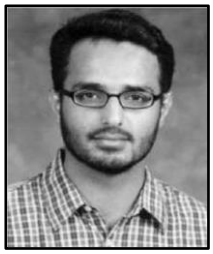
[1] J. Carranza, C. Theobalt, M. A. Magnor and H. -P. Seidel, "Free-viewpoint video of human actors", ACM Trans. Graph, vol. 22, no. 3, **(2003)**, pp. 569-577.
[2] C. Theobalt, N. Ahmed, G. Ziegler and H. -P. Seidel, "High-quality reconstruction of virtual actors from multi-view video streams", IEEE Signal Processing Magazine, vol. 24, no. 6, **(2007)**, pp. 45-57.
[3] 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.
[4] D. Vlasic, I. Baran, W. Matusik and J. Popovic, "Articulated mesh animation from multi-view silhouettes", ACM Trans. Graph, vol. 27, no. 3.
[5] J. Starck and A. Hilton, "Surface capture for performance-based animation", IEEE Computer Graphics and Applications, vol. 27, no. 3, **(2007)**, pp. 21-31.
[6] N. Ahmed, C. Theobalt, C. Rossl, S. Thrun and H. -P. Seidel, "Dense correspondence finding for parametrization-free animation reconstruction from video", in: CVPR, **(2008)**.
[7] MICROSOFT, "Kinect for Microsoft windows and Xbox 360", http://www.kinectforwindows.org/, **(2010)** November.
[8] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz and M. A. Magnor, "Markerless motion capture using multiple color-depth sensors", in: VMV, **(2011)**, pp. 317-324.
[9] A. Weiss, D. Hirshberg and M. J. Black, "Home 3d body scans from noisy image and range data", in ICCV, **(2011)**.
[10] 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", in ICCV, **(2011)**.
[11] Y. M. Kim, D. Chan, C. Theobalt and S. Thrun, "Design and calibration of a multi-view tof sensor fusion system", in: IEEE CVPR Workshop on Time-of-flight Computer Vision, **(2008)**.
[12] N. Ahmed, "A system for 360 degree acquisition and 3d animation reconstruction using multiple RGB-D cameras", in: Proceedings of the 25th International Conference on Computer Animation and Social Agents (CASA), Casa'12, **(2012)**.
[13] P. E. Debevec, T. Hawkins, C. Tchou, H. -P. Duiker, W. Sarokin and M. Sagar, "Acquiring the reflectance field of a human face", in: SIGGRAPH, **(2000)**, pp. 145-156.
[14] T. Hawkins, P. Einarsson and P. E. Debevec, "A dual light stage", in: O. Deussen, A. Keller, K. Bala, P. Dutre, D. W. Fellner, S. N. Spencer (Eds.), Rendering Techniques, Eurographics Association, **(2005)**, pp. 91-98.
[15] P. Einarsson, C. -F. Chabert, A. Jones, W. -C. Ma, B. Lamond, T. Hawkins, M. T. Bolas, S. Sylwan and P. E. Debevec, "Relighting human locomotion with flowed reflectance fields", in T.Akenine-Moller, W. Heidrich (Eds.), Rendering Techniques, Eurographics Association, **(2006)**, pp. 183-194.
[16] 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: A. Hilton, T. Masuda, C. Shu (Eds.), IEEE Workshop on 3-D Digital Imaging and Modeling (3DIM), IEEE, Kyoto, Japan, **(2009)**, pp. 1542-1549.
[17] V. Castaneda, D. Mateus and N. Navab, "Stereo time-of-flight", in ICCV, **(2011)**.
[18] R. Girshick, J. Shotton, P. Kohli, A. Criminisi and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images", in: ICCV, **(2011)**.
[19] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall and H. -P. Seidel, "Markerless motion capture with unsynchronized moving cameras", in: CVPR, **(2009)**, IEEE.
[20] OpenKinect, Open source libraries for Microsoft Kinect, http://www.openkinect.org/.
[21] OpenKinect, Openkinect C-Sync wrapper, http://www.openkinect.org/wiki/c sync wrapper.
[22] L. D. Agapito, E. Hayman and I. Reid, "Self-calibration of rotating and zooming cameras", IJCV, vol. 45, no. 2, **(2001)**, pp. 107-127.
[23] M. Pollefeys, R. Koch and L. V. Gool, "Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters", IJCV, vol. 32, no. 1, **(1999)**, pp. 7-25.

[24] R. I. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision", 2nd Edition, Cambridge University Press, ISBN: 0521540518, **(2004)**.
[25] Z. Zhang, "A flexible new technique for camera calibration", PAMI, vol. 22, no. 11, **(2000)**, pp. 1330-1334.
[26] D. G. Lowe, "Object recognition from local scale-invariant features", in: ICCV, **(1999)**, pp. 1150-1157.
[27] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)", in: ICRA, **(2011)**.
[28] Accompanying video (46 MB), http://dl.dropbox.com/u/7563218/sersc.mp4.

Authors

Naveed Ahmed

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 Engineers 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 Nazir Junejo

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 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.