A Comparative Analysis of Time Coherent 3D Animation Reconstruction Methods from RGB-D Video Data

Naveed Ahmed and Mohammed Lataifeh

***Abstract*—We present a comparative analysis of Time Coherent 3D Animation Reconstruction methods from the RGB-D video data. We analyze the existing methods that can reconstruct a time coherent 3D animation, and also present two modified algorithms that extend the earlier work. We show that using all the methods it is possible to reconstruct a time-coherent 3D animation using either only the color data, color and depth data, or only the depth data. We compare all the methods using a number of error measures and analyze the strength and weaknesses of each method in terms of their accuracy and runtime performance. Our analysis demonstrates that given RGB-D video data, it is possible to select the best algorithm for time coherent 3D animation reconstruction under a number of constraints in terms of the required accuracy and runtime performance.**

***Index Terms*—3D Animation, RGB-D Video, 3D Reconstruction, Multi-view Video, Free-viewpoint Video.**

I. Introduction

THE field of time coherent 3D animation or free-viewpoint video reconstruction has been an active area of research in both computer graphics and computer vision. A number of methods [1] [2] [3] [4] [5] [6] are proposed in the last fifteen years that directly reconstruct a 3D animation from multi-view color (RGB) data. These methods can not only capture the shape of the actor, but also its appearance and motion. One of the earliest work in this area was presented by Carranza et al. [1]. They created a free-viewpoint video of a moving actor from eight synchronized RGB cameras through an optimization process that estimated the rigid body transform of each joint of a template 3D mesh. Later their work was extended by Theobalt et al. [2], who not only captured the shape and motion of the moving actor but also the surface material properties of the actors clothes.

De Aguiar et al. [4] presented a surface deformation-based optimization method that could reconstruct a time coherent 3D animation from eight synchronized RGB cameras. Similarly, Vlasic et al. [5] presented another method of skeleton-based deformation to achieve similar results. Both of these methods rely on a high quality scan of real-world actor that was used as a template model. In contrast, Ahmed et al. [6] directly reconstructed visual hulls from each frame of multi-view video RGB data and then use a shape matching approach to reconstruct a time coherent 3D animation.

In recent years, a number of new camera technologies allow to capture the depth video in addition to the RGB video data. Time of Flight (ToF) sensors provide dynamic depth data [7] [8] that can be deployed with the RGB cameras to capture RGB-D video data. A number of methods have been proposed that employ ToF sensors to reconstruct a static or dynamic representation of a 3D scene. Kim et al. [7] presented a multi-sensor fusion system comprising of RGB and ToF sensors for RGB-D static scene acquisition. Castaneda et al. [9] used two depth sensors for stereo ToF acquisition of a static scene. For the dynamic scenes, Kim et al. [10] presented a complete acquisition system comprising of high resolution RGB video cameras and low resolution ToF sensors to capture true RGB-D video data. Their work did not focus on reconstructing a time coherent 3D scene representation.

With the advent of Microsoft Kinect [11], the fusion of low cost RGB and depth sensor has been widely available. Kinect has been employed in a number of application domains, and has been widely used for dynamic scene capture and 3D scene reconstruction. Berger et al. [12] employed four Kinects for marker-less motion capture. Weiss et al. [13] used Kinect for human shape reconstruction. Baak et al. [14] employed a single depth sensor to track full body motion. The seminal work for the pose estimation using Kinect was presented by Girshick et al. [15]. Ye et al. [16] used three hand-held Kinects for marker-less motion capture. Ahmed et al. [17] employed six Kinects for 360 degree acquisition and 3D animation reconstruction of human actors. All of the above methods also did not try to reconstruct time coherent 3D animation.

Recently, Ahmed et al. [18] [19] [20] [21] presented a number of methods have been proposed to reconstruct time coherent 3D animation from RGB-D video data. All of the methods rely on a sparse matching between various frames of RGB data that is used to create a dense matching algorithm using both RGB and depth features to reconstruct a time coherent 3D animation. They employed a non-linear matching algorithm that used both RGB and depth data for 3D animation reconstruction [18]. In this method, first the sparse matching in RGB space is established, and then a non-linear matching algorithm was created using the surface orientation, RGB difference, RGB feature distance, and the distance in 3D space. Thus it relied on both RGB and depth data for the time coherent 3D animation reconstruction.

Afterward, they modified the non-linear dense matching algorithm to incorporate surface curvature instead of the RGB color difference and the 3D distance measure [21]. This method still relied on both RGB and depth features, though the use of depth features was limited. Instead of relying on the non-linear matching, they later presented a geometric matching approach [19] that enhanced the sparse

Manuscript received July 17, 2018; revised October 10, 2018.

N. Ahmed and M. Lataifeh are with the Department of Computer Science, University of Sharjah, Sharjah, 27272, UAE. E-mail: nahmed at sharjah.ac.ae and mlataifeh at sharjah.ac.ae

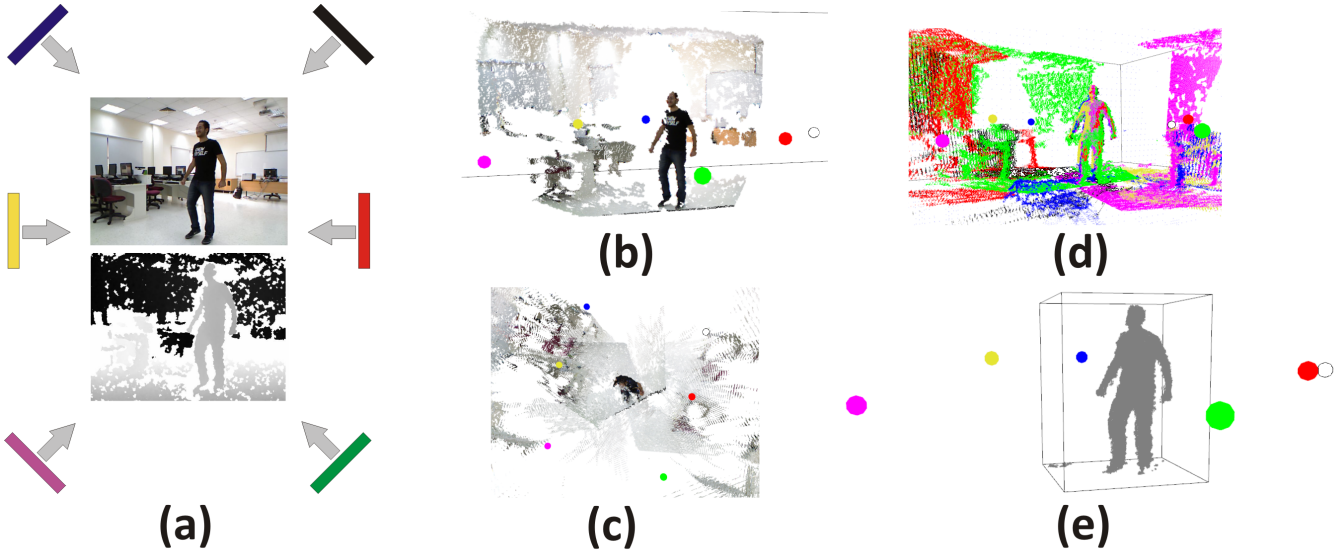

Fig. 1. Multi-view Kinect Acquisition Pipeline. (a) Six Kinects are used to acquire the RGB and depth images (only one frame from one camera is shown). (b) shows the 3D point cloud from one camera with the mapped RGB image. (c) Shows the top down view of six merged 3D point clouds. The alignment of the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

matching from RGB data and then a dense matching algorithm using the 3D motion data to reconstruct time coherent 3D animation. This work only used RGB data for the very initial sparse matching and the later algorithm did not rely on the RGB features at all. Finally, they completely removed RGB matching for the initial correspondence [20] and only used 3D features to reconstruct a time coherent 3D animation from RGB-D video data.

In all of the above presented works, each method is presented individually, and except for [18] and [19], is not analyzed for its quality in terms of its time coherence quality and runtime performance. Recently, Ahmed [22] analyzed a number of algorithms for time coherent 3D animation reconstruction. They performed the accuracy and performance analysis only for 3D and 2D features but did not consider the hybrid features. In addition, there is no method that only relies on the RGB features, instead of the depth features. Also, they have used multiple types of 3D features to reconstruct a 3D animation but there is no analysis about which type of 3D features are best suitable for a time coherent 3D animation reconstruction in terms both the quality and the runtime performance.

In this paper, we analyze the existing time coherent 3D animation reconstruction methods, and also present two modifications to the existing methods to verify how the choice of various types of RGB and depth feature affects the accuracy and runtime performance of the algorithms. We first introduce a method by only using RGB features for both coarse and dense matching for 3D animation reconstruction. Afterward, we introduce a second modification to combine both RGB and depth features for the coarse matching and reconstruct a 3D animation using this new set of coarse matches. Finally, we analyze all the methods in terms of their accuracy using three error terms and their runtime performance.

The main contributions of this paper are:

- Two proposed modifications to existing time coherent 3D animation reconstruction methods from RGB-D video data for detailed analysis and increased accuracy.
- Accuracy analysis of existing and proposed time coherent 3D animation reconstruction methods using three error measures.
- Runtime performance analysis of existing and proposed time coherent 3D animation reconstruction methods.

In the following sections we will present the work as follows: The data acquisition system employed in all the methods is detailed in Sect. II. An overview of different time coherent 3D animation reconstruction techniques is presented in Sect. III. This section discusses all the existing methods, and also presents the discussion of the two modified algorithms in Sect. III-A3 and Sect. III-B3. Results, evaluation, and a detailed analysis of each method in terms of its accuracy and runtime performance is presented in Sect. IV. Finally, the paper concludes in Sect. V.

II. Data Acquisition

All the methods discussed in this paper rely on RGB-D data acquisition using one or more Kinect cameras. At maximum, up to six Kinects are used to capture a 3D animation [17]. In general, it does not matter how many Kinects are used, because all the algorithms work on RGB-D video data registered in a global coordinate system [17] [19]. In case of multiple Kinects, if more than two cameras are used, the interference between Kinects causes the loss of depth data for one camera that is filled by the other camera [17].

Kinect captures both 640x480 pixels of RGB and depth data at 30 frames per second. Multi-view acquisition is handled through a software-based synchronization setup. In general the system is not limited to a static camera setup, though all the methods discussed are only applied to the RGB-D data acquired using a static camera setup. For a static setup there is only one-time extrinsic camera calibration in a global coordinate system, whereas a dynamic setup would require a calibration step at each frame [17]. Additionally, a dynamic camera setup would need a different solution for the

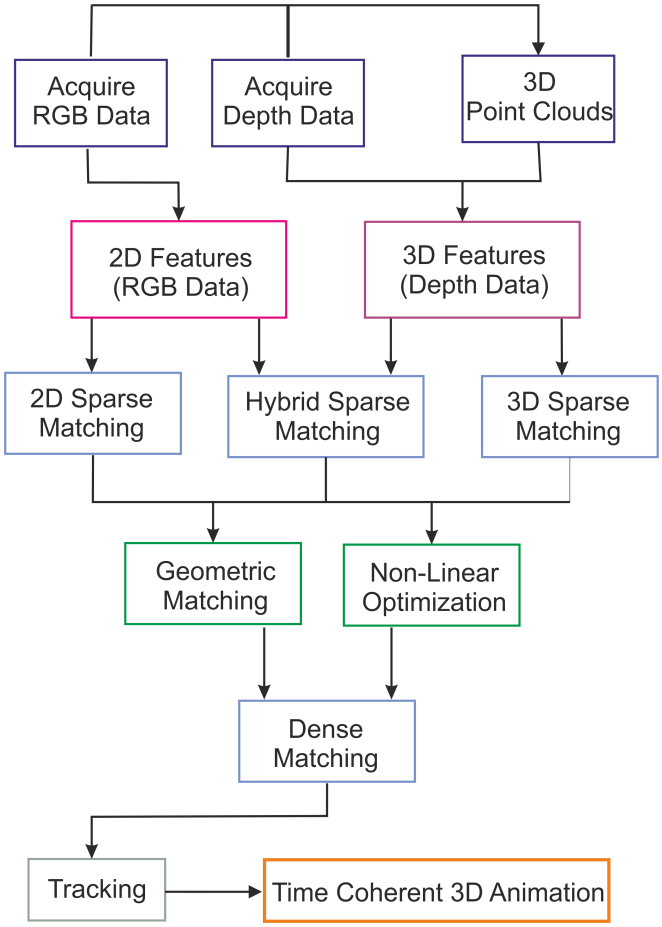

Fig. 2. Flowchart of all the methods starting from RGB-D data acquisition to temporally coherent 3D animation reconstruction. Three methods rely on sparse RGB features, whereas one method uses 3D features. Hybrid sparse matching is our introduction in this paper to improve the accuracy of the methods.

background subtraction, which is trivial for a static camera setup, where the background is recorded in advance.

Kinect's software development kit (SDK) is employed to capture all the data and find the mapping between the RGB and depth sensors. The SDK also allows the resampling of the depth data into a 3D point cloud. This allows a 360 visualization of the RGB-D data. The RGB to depth mapping then allows the 3D point cloud to be rendered with the correct color value resulting in a true 3D appearance of the human actor. There is no filtering applied on the 3D point clouds to remove the noise or outliers [23]. The extrinsic camera calibration allows all 3D point clouds to be registered in a unified global coordinate system that allows true 360 degrees 3D animation visualization [17]. Finally, a simple depth based background subtraction is performed to segment the actor from the background. Fig. 1 shows a complete acquisition pipeline for all the methods, starting from RGB-D data acquisition to the visualization of RGB mapped 3D point clouds in a unified global coordinate system.

III. Time Coherent 3D Animation Reconstruction

The acquired RGB-D video data provides us a 3D point cloud with RGB mapping at each time step. The data is not time coherent as there is no connectivity or tracking information available from one frame to the next. This type of data can be used for 360 degrees visualization, but cannot be used for any low level analysis of the video data, e.g. motion capture, action recognition, compression etc. Even in terms of the visualization, if the data is not time coherent then it doesn't look smooth in appearance. Therefore, for better video visualization and analysis, it is important to extract time coherence from the acquired RGB-D video data.

In this section, we will present a number of algorithms for time coherent 3D animation reconstruction, as presented by Ahmed et al. [18] [19] [21] [20]. We also introduce two modified algorithmic steps, one for the sparse matching, and one of the dense matching. Input to all of these methods is a sequence of 3D point clouds with RGB mapping acquired using one or more Kinect cameras (Sect. II). A 3D point cloud at each frame is independent of the other, and the number of 3D points is different in each frame. Let us denote a 3D point cloud as $\mathcal{C} = (\mathcal{V}, \mathcal{T})$, where $(\mathcal{V}, \mathcal{T})$ denotes the set of all 3D points and their corresponding RGB mapping in the point cloud. Therefore, for $(\mathcal{V}, \mathcal{T}) \in \mathcal{C}$ we will associate for each 3D position $p \in \mathcal{V}$ a 3D point (x, y, z) and its texture coordinate (u, v) to each texel (2D position in an image) $q \in \mathcal{T}$. Using $\mathcal{T}$ all 3D positions $\mathcal{V}$ obtained from the depth data are mapped to the corresponding RGB value. Since we consider a video sequence consisting of N time-frames, therefore we write the sequence of point clouds as a function of time t. Thus $\mathcal{C}(t) = (\mathcal{V}(t), \mathcal{T}(t))$, where t=0, ..., $N-1$. It is to be noted that one of the algorithm does not use RGB mapping at all and only relies on 3D features as explained in Sect. III-A2.

The aim of all algorithms is to track the $\mathcal{C}(0)$ over the complete animation sequence by mapping it iteratively to each $\mathcal{C}(t)$ in the sequence. That is, first mapping is from $\mathcal{C}(0)$ to $\mathcal{C}(1)$ which yields $\mathcal{C}_0(1)$, i.e. $\mathcal{V}(0) \in \mathcal{C}(0)$ aligned to $\mathcal{C}(1)$ with respect to its mapping. Thus $\mathcal{C}_0(t)$ will refer to $\mathcal{C}(0)$ aligned with $\mathcal{C}(t)$ after t iterations of the algorithm where t=0, ..., $N-1$.

All of these algorithms rely on sparse matching using either the RGB data, or the depth data. Then the algorithms either use both RGB and depth data to create a non-linear optimization function, or a geometric matching algorithm for the dense matching of the RGB-D data over consecutive frames resulting in the time coherent 3D animation. The final result of all of these methods is a single 3D point cloud tracked over the entire animation sequence.

We will first discuss the sparse matching step employed by all the methods, and present our modified algorithm in the following sections. Afterward, the dense matching algorithms are discussed in detail. A flowchart depicting the pipeline of all the algorithms can be seen in Fig. 2.

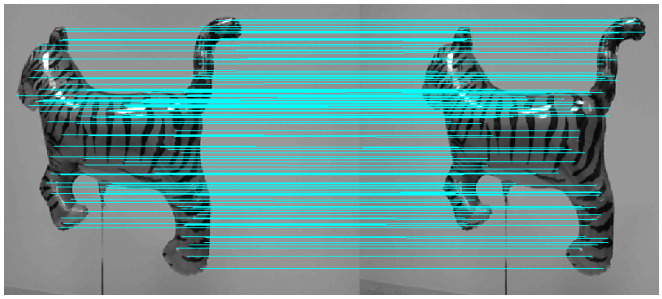

Fig. 3. A visualization of 2D sparse matching using SIFT features.

A. Sparse Matching

For all the algorithms, the first step is to establish a sparse correspondence between two consecutive frames of the RGB-D video data by means of feature matching in either the RGB or depth space. These sparse matchings are then used to establish the dense matching for time coherent 3D animation reconstruction. Three methods [18] [19] [21] use the RGB features, while the fourth method [20] only uses the depth features. The quality of sparse matching plays a very important role in reconstructing a high quality 3D animation, as discussed in the Results and Analysis section (Sect. IV). Therefore, we also introduce a hybrid approach for sparse matching using both RGB and depth data as explained in Sect. III-A3.

1) RGB Features: The first three methods [18] [19] [21], use RGB feature matching to initialize the dense matching. For every input RGB frame $I_c(t)$ for all time steps t and cameras c, these algorithms start by extracting the 2D SIFT [24] or SURF [25] feature locations. For all the RGB-D video sequences, around 200 to 300 features are obtained for each input image. Using RGB features has a number of benefits, mainly accuracy, stability and rotational and scale invariance. Each RGB feature has a location $q(t) = (u, v, t)$ in the texture space, and using the formulation $(\mathcal{V}(t), \mathcal{T}(t)) \in \mathcal{C}(t)$ each RGB feature is mapped to the corresponding $p(t) \in \mathcal{V}(t)$. All 3D points at time t that are associated with the RGB feature points are donated as the RGB feature points $\mathcal{L}(t)$.

In the next step, a mapping between $\mathcal{L}(t)$ and $\mathcal{L}(t+1)$ is established by finding the matching between the corresponding RGB features by using a simple Euclidean distance measure $\mathcal{D}$. This is a trivial step employed in many RGB based matching algorithms, where a match is established if the ratio of $\mathcal{D}$ between the nearest and second nearest feature is less than a certain threshold. This measure also helps in eliminating most of the false positives. At the end of this step, a sparse matching is established between two 3D point clouds. An example of sparse matching can be seen in Fig. 3.

2) 3D Features: The final method [20], does not rely on RGB features, rather it only uses the depth data to extract 3D features that are used for a sparse matching between consecutive depth frames. In the first step, for every input depth frame $D_c(t)$ for all time steps t and camera c, 3D SIFT [26] features are extracted. Each 3D SIFT feature has a location $q(t) = (x, y, z, t)$ in the 3D space, and for each $q(t)$, its underlying local surface curvature and normal (orientation) is calculated using 20 nearest points. All 3D points at time t that are associated with $q(t)$ are donated as the 3D feature points $\mathcal{L}(t)$.

Similar to the RGB feature matching, a mapping between $\mathcal{L}(t)$ and $\mathcal{L}(t+1)$ is established by finding the matching between the corresponding RGB features by using a simple Euclidean distance measure $\mathcal{D}$. In addition, to increase the reliability of the sparse matching the underlying curvature is also matched to eliminate the outliers. At the end of this step, a sparse matching is established between two 3D point clouds.

3) Hybrid Features: As explained in the previous sections, the proposed methods either use the RGB sparse matching, or 3D sparse matching. As sparse matching is the first step for all the algorithms, its reliability is crucial, because the dense matching algorithms depend on the quality of the sparse matching. Therefore, in order to improve the quality of the sparse matching, in this paper we propose a new hybrid algorithm that uses both RGB and 3D features for the sparse matching and then test all the algorithms using the newly established sparse matching results.

Similar to the previous sparse matching algorithms, for every input RGB frame $I_c(t)$ and depth frame $D_c(t)$ for all time steps t and cameras c, both the RGB and 3D features are extracted as explained in the previous sections. The matching algorithm for both features results in two set of matches, one for RGB, and one for 3D features. Thus, in general we get almost double the sparse features that otherwise would have been obtained by only using the RGB or 3D features. This greatly improves the initial reliability of the algorithms, and the improvements in the results using the hybrid sparse matching can be seen in the Results and Analysis section (Sect. IV).

B. Dense Matching

The sparse matching provides an initial map between around 300 points of two consecutive point clouds $\mathcal{C}(t)$ and $\mathcal{C}(t+1)$. In case of the hybrid sparse matching, the number of sparse features are around 600. A typical 3D point cloud is comprised of at least 60,000 3D points. Thus, few hundred matches are not sufficient to track the motion of the point cloud due to a number of local deformations. Therefore, a number of dense matching algorithms are proposed that either use a non-linear optimization, or geometric matching, to find the mapping for all the 3D points from $\mathcal{C}(t)$ to $\mathcal{C}(t+1)$. In the following sections we will briefly review the proposed methods, and also suggest one modified method.

1) Non-Linear Optimization using Color and Orientation: In order to determine the dense matching, Ahmed et al. [18] proposed a non-linear optimization method, where the objective function is comprised of a number of RGB and depth features at each 3D point combined with the sparse features. In the first step, for each 3D point $p(t)_i$ in $\mathcal{C}(t)$, the normal of that point $\mathbf{N}(p(t)_i)$ is estimated by fitting a plane to its 10 nearest points and finding its orientation. The RGB color associated with every $p(t)_i$ is defined as $\mathbf{C}(p(t)_i)$. Additionally, the Euclidean distance of $p(t)_i$ with some other point at $t+1$ is defined as $\mathbf{D}(p(t)_i)$. Finally, the Euclidean distance of $p(t)_i$ to its nearest sparse feature is referred as $\mathbf{F}(p(t)_i)$. Thus given these terms the non-linear matching function is defined as:

$$\begin{aligned}\mathbf{M}(p(t)_i) = \alpha(1.0 - \mathbf{N}(p(t)_i).\mathbf{N}(p(t+1)_i)) + \\ \beta(\|\mathbf{C}(p(t)_i) - \mathbf{C}(p(t+1)_i)\|) + \\ \gamma(\|\mathbf{F}(p(t)_i) - \mathbf{F}(p(t+1)_i)\|) + \delta\mathbf{D}(p(t)_i) \quad (1)\end{aligned}$$

$\mathbf{M}(p(t)_i)$ is the matching distance, $1.0 - \mathbf{N}(p(t)_i).\mathbf{N}(p(t+1)_i)$ is the angular difference in orientation, with the similar orientation resulting in a smaller value. $\|\mathbf{C}(p(t)_i) - \mathbf{C}(p(t+1)_i)\|$ is the absolute difference of color components between (R, G, B) components of two 3D points. $\|\mathbf{F}(p(t)_i) - \mathbf{F}(p(t+1)_i)\|$ is the absolute difference in the distance to the nearest sparse feature and

$\mathbf{D}(p(t)_i)$ is the 3D Euclidean distance between $p(t)_i$ and $p(t+1)_i$. The four parameters α, β, γ, and δ 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. In this method, their values are $\alpha = 0.25$,$\beta = 0.2$, $\gamma = 0.5$,$\delta = 0.05$. These values are found through experiments. Most weight is given to the difference to the nearest sparse feature because it has a higher degree of accuracy. Least weight is chosen for $\mathbf{D}(p(t)_i)$ because in principal the difference in 3D Euclidean 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 $t+1$ match the feature distance, orientation and the color. The matching point $p(t+1)_i$ is the one with the minimum value of the convex combination. If two points result in the same value of $\mathbf{M}(p(t)_i)$, then the point with smaller $\|\mathbf{F}(p(t)_i) - \mathbf{F}(p(t+1)_i)\|$ is chosen as the matching point. In the unlikely case of same values for $\mathbf{M}(p(t)_i)$ and $\|\mathbf{F}(p(t)_i) - \mathbf{F}(p(t+1)_i)\|$, $1.0 - \mathbf{N}(p(t)_i).\mathbf{N}(p(t+1)_i)$ is used to find the matching point, followed by $\|\mathbf{C}(p(t)_i) - \mathbf{C}(p(t+1)_i)\|$ and $\mathbf{D}(p(t)_i)$. The results of this method are discussed in Sect. IV.

2) Non-Linear Optimization using Curvature and Orientation: Ahmed et al. later modified [18] and removed the color and distance term and introduced a new curvature term. For each 3D point $p(t)_i$ in $\mathcal{C}(t)$, the curvature of that point $\mathbf{U}(p(t)_i)$ is estimated by fitting a second order surface to 30 nearest points. In addition, the sum of absolute difference to two nearest feature points is defined as $\mathbf{F}_2(p(t)_i)$. Finally, using the normal of each point $\mathbf{N}(p(t)_i)$ the non-linear matching function is defined as:

$$\begin{aligned}\mathbf{M}(p(t)_i) = \alpha(1.0 - \mathbf{N}(p(t)_i).\mathbf{N}(p(t+1)_i)) + \\ \beta(\|\mathbf{U}(p(t)_i) - \mathbf{U}(p(t+1)_i)\|) + \\ \gamma(\|\mathbf{F}_2(p(t)_i) - \mathbf{F}_2(p(t+1)_i)\|) \quad (2)\end{aligned}$$

Similar to the previous method, $\mathbf{M}(p(t)_i)$ is the matching distance. The values for the three weighting parameters are $\alpha = 0.25$,$\beta = 0.2$, and $\gamma = 0.5$. The distance to the feature points is given the most weight because of its accuracy. Orientation and curvature have the similar weight. All the weighting parameters are found through experiments. The results of this method are discussed in Sect. IV.

3) Non-Linear Optimization using RGB data: The two optimization functions in the previous sections, Eq. 1 and Eq. 2, mostly rely on both RGB and depth features. The optimization function in Eq. 1 uses the color term that is discarded in Eq. 2. The distance to the nearest sparse feature point is computed in the 3D space using the depth to RGB mapping. In order to truly quantify the impact of RGB features, we thus propose a modified non-linear objective function that only uses RGB features. This function only uses the RGB color, $\mathbf{C}(p(t)_i)$, and the distance to the two nearest sparse features in the RGB space $\mathbf{F}_2(p(t)_i)$:

$$\begin{aligned}\mathbf{M}(p(t)_i) = \alpha(\|\mathbf{C}(p(t)_i) - \mathbf{C}(p(t+1)_i)\|) + \\ \beta(\|\mathbf{F}_2(p(t)_i) - \mathbf{F}_2(p(t+1)_i)\|) \quad (3)\end{aligned}$$

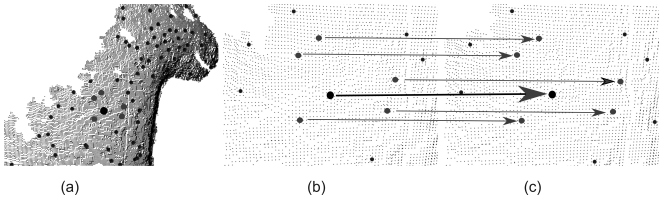

Fig. 4. Motion-based dense matching. Five nearest sparse features (middle sized, dark gray) are used to find the motion of an arbitrary point (bigger circle, black).

Similar to the previous method, $\mathbf{M}(p(t)_i)$ is the matching distance. The values for the two weighting parameters are $\alpha = 0.2$, and $\beta = 0.8$. The weights are found through experiment. The higher weight is given to the sparse feature points distance as it is more accurate. It is obvious that the quality of this dense matching will be less accurate compared to the previous methods. We have only introduced this method to do a comparative analysis and show that using both RGB and depth features result in a more accurate time coherent 3D animation reconstruction. The results of this method are discussed in Sect. IV.

4) Geometric Matching using RGB and Depth Features: The two methods [19] [20], do not use a non-linear optimization for the dense matching, rather they use a geometrical matching approach to enhance the accuracy of the sparse feature matching and then use a motion-based tracking algorithms to estimate the dense matches. The main different between the two method is that [19] relies on the sparse RGB feature matching as its starting point, whereas [20] relies on the sparse 3D feature matching as its starting point. The method [19] directly uses the matching clusters $\mathcal{L}(t)$ and $\mathcal{L}(t+1)$ using the sparse RGB matching. On the other hand, the other method [20] uses Clustered Viewpoint Feature Histogram (CVFH) [27] on the sparse 3D features to find the matching clusters $\mathcal{L}(t)$ and $\mathcal{L}(t+1)$. After the matching clusters are established both methods follow the similar algorithm of refining the sparse matches as follows:

1) Randomly choose one sparse feature $\mathbf{l}_0(t)$ from $\mathcal{L}(t)$ and the corresponding matching feature $\mathbf{l}_0(t)$ from $\mathcal{L}(t+1)$.
2) Define a plane $\mathcal{P}(t)$ using two nearest sparse features ($\mathbf{l}_1(t)$ and $\mathbf{l}_2(t)$) with respect to $\mathbf{l}_0(t)$.
3) Define a plane $\mathcal{P}(t+1)$ using two nearest sparse features ($\mathbf{l}_1(t+1)$ and $\mathbf{l}_2(t+1)$) with respect to $\mathbf{l}_0(t+1)$.
4) Project all sparse feature points $\mathcal{L}(t)$ on $\mathcal{P}(t)$ and $\mathcal{L}(t+1)$ on $\mathcal{P}(t+1)$.
5) Find the new matches of the sparse features $\mathcal{L}(t)$ and $\mathcal{L}(t+1)$ in the parametric space of the planes $\mathcal{P}(t)$ and $\mathcal{P}(t+1)$. Update the sparse matches.
6) Repeat from step 1 unless the matching stabilizes.

Once the newly refined sparse matching is defined, for each 3D point $p_i(t)$, its nearest five closest sparse features matches are used to identify its motion from $\mathcal{C}(t)$ to $\mathcal{C}(t+1)$. Thus, all the points $p_i(t)$ in $\mathcal{C}(t)$ are mapped to $\mathcal{C}(t+1)$ and consequently tracked over the whole sequence. A visualization of motion vectors-based matching can be seen in Fig. 4. The results of this method are discussed in Sect. IV.

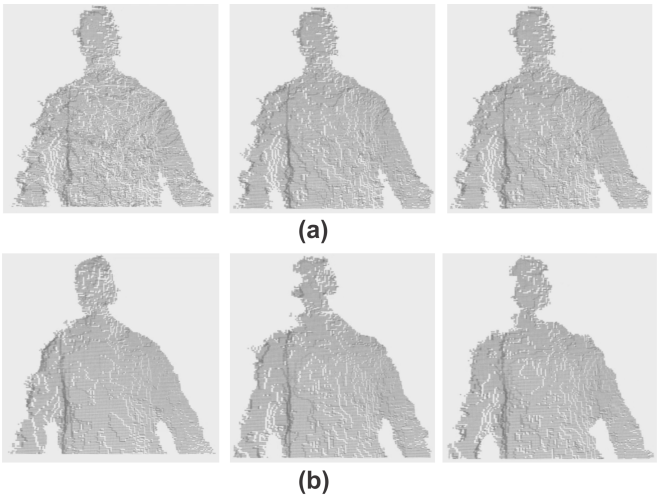

Fig. 5. (a) shows three frames from time coherent 3D animation reconstructed from a single Kinect using the geometric matching algorithm. It clearly shows a consistent point cloud. (b) shows the non-coherent 3D point clouds, where dramatic differences between the 3D point clouds are visible, especially in the head.

IV. Results and Analysis

All of the discussed methods were able to reconstruct a time coherent 3D animation from a sequence of non-coherent 3D point clouds. The non-coherent 3D point cloud data was acquired from the acquisition setup explained in Sect. II and from Ahmed et al. [6]. The data from the Kinect-based acquisition setup provides 3D point clouds with RGB mapping registered in a global coordinate system. The data from Ahmed et al. [6] is acquired using eight synchronized RGB cameras. The 3D point cloud data is obtained from visual hulls that are reconstructed at each time step. All the sequences are 100 to 200 frames long and range from slow walking motion to fast boxing or fast capoeira motion.

Fig. 5a shows different frames of the time coherent 3D animation reconstructed from the Kinect-based acquisition setup. Non-coherent frames of the 3D animation can be seen in Fig. 5b. It can be seen that the time coherent 3D animation remains consistent throughout the different frames, whereas the non-coherent data changes dramatically. Similarly, the results from the time coherent 3D animation reconstructed from the data from Ahmed et al. [6] can be seen in Fig. 6a. The non-coherent animation frames can be seen in Fig. 6b. Again the different between the two in terms of the connectivity from one frame to the next is well pronounced.

In order to quantitatively compare the methods, we have measured three different types of errors for all five methods:

1) Silhouette and Convex Hull Consistency
2) Bounding Box Containment
3) Deformation Measure

Each of these methods are tested with two sets of sparse features, i.e, RGB or 3D features and Hybrid (RGB and 3D) sparse features. Below is a general description of each of the error measure, followed by the analysis.

A. Silhouette and Convex Hull Consistency

Silhouette and Convex Hull Consistency error measure computes the average of differences in silhouette overlap and convex hull areas of the time coherent 3D animation with the input 3D point cloud rendered from one camera view. The temporally consistent point cloud is rendered 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. Both, the original non-coherent 3D point cloud and the spatio-temporally coherent 3D point cloud are rendered from the same camera view and their silhouettes are extracted. The two silhouettes are overlapped and the number of pixels that do not overlap for each frame are counted. Similarly, the area of the two convex hulls is calculated and ideally the area of both convex hulls should be equal. The final error is the average of the silhouette overlap error and the difference in the convex hull areas.

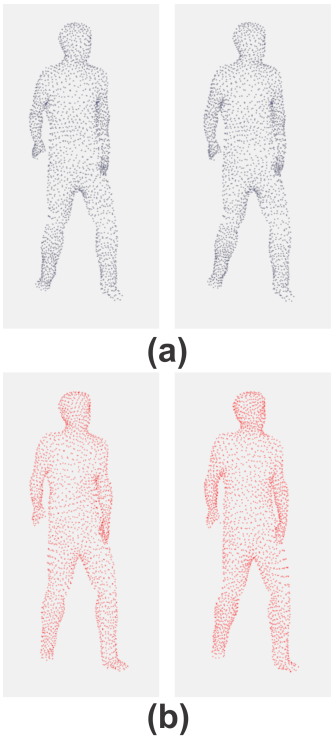

Fig. 6. (a) shows two frames of time coherent 3D point cloud, whereas (b) shows the non-coherent frames. It can be seen how the point cloud changes in (b) whereas it remains consistent in (a).

B. Bounding Box Containment

The bounding box based containment error is measured by first finding the bounding box of the non-coherent 3D point cloud at each time step. In the second step, it counts the 3D points in the temporally coherent point cloud that are not inside the corresponding bounding box. This measure also provides a good quantitative analysis in analyzing the goodness of the tracking algorithm and its temporal consistency.

C. Deformation Measure

Deformation error measure is calculated by comparing the distances between a small set of points at each frames under the assumption that dynamic object goes through a small deformation. This is achieved by sampling 200 points evenly distributed over $\mathcal{C}(0)$ and store the distance vectors between each one of them for the starting frame in a list $\mathcal{E}_i(0)$, where

TABLE I
AVERAGE ERROR MEASURES TO QUANTIFY THE ACCURACY OF EACH METHOD

Method	Average Silhouette and Convex Hull Error	Average Bounding Box Error	Average Deformation Error
Non-Linear (Sparse: RGB) (Dense: RGB+Depth)	2.84%	1.8%	2.45%
Non-Linear (Sparse: Hybrid) (Dense: RGB+Depth)	2.69%	1.7%	2.33%
Non-Linear (Sparse: RGB) (Dense: Depth)	2.95%	1.83%	2.5%
Non-Linear (Sparse: Hybrid) (Dense: Depth)	2.78%	1.75%	2.39%
Non-Linear (Sparse: RGB) (Dense: RGB)	3.46%	2.21%	3.21%
Non-Linear (Sparse: Hybrid) (Dense: RGB)	3.31%	2.13%	2.9%
Geometric Matching (Sparse: RGB) (Dense: Motion)	3.29%	2.1%	2.8%
Geometric Matching (Sparse: 3D) (Dense: Motion)	3.62%	2.35%	3.1%
Geometric Matching (Sparse: Hybrid) (Dense: Motion)	3.11%	1.93%	2.61%

TABLE II
RUNTIME PERFORMANCE OF ALL THE METHODS

Method	Frames per Minute
Non-Linear (Sparse: RGB) (Dense: RGB+Depth)	12
Non-Linear (Sparse: Hybrid) (Dense: RGB+Depth)	11
Non-Linear (Sparse: RGB) (Dense: Depth)	11
Non-Linear (Sparse: Hybrid) (Dense: Depth)	11
Non-Linear (Sparse: RGB) (Dense: RGB)	14
Non-Linear (Sparse: Hybrid) (Dense: RGB)	13
Geometric Matching (Sparse: RGB) (Dense: Motion)	20
Geometric Matching (Sparse: 3D) (Dense: Motion)	18
Geometric Matching (Sparse: Hybrid) (Dense: Motion)	17

i=0, ..., $\|\mathcal{E}\| - 1$ and $\|\mathcal{E}\|$ is the total number of vectors in $\mathcal{E}_i(0)$. After tracking, the same distance vectors $\mathcal{E}_i(t)$ are calculated for each tracked frame $\mathcal{C}_0(t)$, where t=1, ..., $N - 1$. The error measure $E_i(t)$ for one frame at time-step t is defined as:

$$E_i(t) = \frac{\sum_{i=0}^{\|\mathcal{E}\|-1} \|\mathcal{E}_i(t) - \mathcal{E}_i(0)\|}{\|\mathcal{E}\|} \tag{4}$$

whereas the average error measure E for the complete sequence is defined as:

$$E = \frac{\sum_{t=1}^{N-1} E_i(t)}{N - 1} \tag{5}$$

D. Analysis

We analyze all the methods in terms of previously defined three error measures and their runtime performance. The error measures for each of the method with different sets of sparse features can be seen in Table. I. The runtime performance of each method can be seen in Table. II.

As can be seen in Table. I, the non-linear methods in general are more accurate and result in a lower overall error. With the exception of non-linear matching that relies only the RGB data, which is comparatively much worse. Similarly, our proposed hybrid sparse matching scheme results in a better accuracy. The reason being that it results in a higher number of sparse feature points that result in a better quality of the dense matching. The downside of the non-linear methods as can be seen in Table. II is their significantly lower runtime performance compared to the geometric matching algorithms. This is due to the fact that the non-linear function is to be evaluated against all possible points in the 3D point clouds, whereas the motion-based dense matching requires only average motion of the nearest 5 motion vectors.

The geometric matching algorithms as can be seen in Table. I are lower in terms of the accuracy, but as discussed earlier they are much faster in their runtime performance as can be seen in Table. II. Same as the non-linear matching, the hybrid sparse matching results in the best overall performance at the expense of the frame rate. In general, 3D sparse matching is slower then RGB sparse matching because it needs both 3D sift, and surface curvature to eliminate the outliers. Therefore all the algorithms that are using hybrid sparse matching have a lower runtime performance.

Based on these extensive evaluations of each of the method, one can draw the following conclusions:

- If the runtime performance is not important and the best quality is required then the non-linear matching using the hybrid sparse matching and the dense matching using both RGB and depth data should be used.
- In case of the runtime performance constraints the geometric matching algorithm with the hybrid sparse matching should be employed.
- Depth data alone cannot provide very high quality of time coherent 3D animation, and same is true for only using the RGB data. This does not limit the algorithms, as shown by our analysis that even if only one type of data is available, time coherent 3D animation reconstruction is possible at the cost of accuracy.
- A hybrid approach is recommended for the sparse matching, but for the dense matching the depth information is more important than the RGB information.
- One of the major limitations of all four methods was using either the depth or the RGB data for the sparse features. This limitation can only be circumvented by means of a hybrid approach and our analysis proves it conclusively.

It is to be noted that some other limitations of all the methods are still there, e.g. no surface representation and relying on heuristics for finding the non-linear objective function coefficients. In addition, the methods are only tested on the data from Microsoft Kinect v1 or 3D point clouds obtained from visual hulls reconstructed from multi-view RGB video data. If higher quality 3D point clouds are available then dynamic surface could be generated resulting in better objective functions. Similarly, acquiring or manually creating some ground truth data would result in a better estimation of non-linear objective function coefficients. Also, some other techniques like machine learning can be employed to replace the heuristics, as none of the non-linear methods use a learning method to improve the initial estimation of the weighting factors [28] [29] [30]. These additions can improve the accuracy of the methods. In future work, as new methods are proposed we plan to further extend this analysis taking into account the limitations of these methods.

V. Conclusions

In this paper, we presented a comprehensive analysis of a number of time coherent 3D animation reconstruction methods using RGB-D video data. We compared four existing methods and also proposed two modifications of these methods to improve the accuracy of the methods and better analyze the various algorithmic components of each method. We evaluated all the methods with and without the modifications in terms of their accuracy and runtime performance. We used three different error measures, silhouette and convex hull consistency, bounding-box containment, and deformation measure, to find the goodness of each method. Based on the error measures and runtime performance we made a number of recommendations about the viability of each method. Our analysis show that it is not only possible to reliably reconstruct a time coherent 3D animation from RGB-D video data, but it is also possible to select the best method depending on the accuracy and runtime requirements and constraints.

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, pp. 569–577, 2003.

[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, pp. 45–57, 2007.

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

[4] 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.

[5] D. Vlasic, I. Baran, W. Matusik, and J. Popovic, "Articulated mesh animation from multi-view silhouettes," *ACM Trans. Graph.*, vol. 27, no. 3, 2008.

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

[7] Y. M. Kim, D. Chan, C. Theobalt, and S. Thrun, "Design and calibration of a multi-view tof sensor fusion system," in *CVPR Workshop*, 2008.

[8] T. Fujita and T. Yoshida, "3d terrain sensing by laser range finder with 4-dof sensor movable unit based on frontier-based strategies," *Engineering Letters*, vol. 24, no. 2, pp. 164–171, 2016.

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

[10] 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 *3DIM*. Kyoto, Japan: IEEE, 2009, pp. 1542–1549.

[11] MICROSOFT, "Kinect for microsoft windows and xbox 360. http://www.kinectforwindows.org/," November 2010.

[12] 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.

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

[14] 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.

[15] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *ICCV*, 2011.

[16] M. Ye, X. Wang, R. Yang, L. Ren, and M. Pollefeys, "Accurate 3d pose estimation from a single depth image," in *Proceedings of the 2011 International Conference on Computer Vision*, ser. ICCV '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 731–738.

[17] 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)*, ser. Casa'12, 2012.

[18] N. Ahmed and I. Junejo, "A system for 3d video acquisition and spatio-temporally coherent 3d animation reconstruction using multiple rgb-d cameras," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 6, no. 2, pp. 113–128, 2013.

[19] N. Ahmed and S. Khalifa, "Time-coherent 3d animation reconstruction from rgb-d video," *Signal, Image and Video Processing*, vol. 10, no. 4, pp. 783–790, 2016. [Online]. Available: http://dx.doi.org/10.1007/s11760-015-0813-1

[20] N. Ahmed, "Multi-view rgb-d synchronized video acquisition and temporally coherent 3d animation reconstruction using multiple kinects," in *Feature Detectors and Motion Detection in Video Processing*. IGI Global, 2016, ch. 7, pp. 142–163.

[21] N. Ahmed and I. Junejo, "Using multiple rgb-d cameras for 3d video acquisition and spatio-temporally coherent 3d animation reconstruction," *International Journal of Computer Theory and Engineering*, vol. 6, no. 6, pp. 447–450, 2014.

[22] N. Ahmed, "Accuracy and performance analysis of time coherent 3d animation reconstruction from rgb-d video," in *Trends and Advances in Information Systems and Technologies*, Á. Rocha, H. Adeli, L. P. Reis, and S. Costanzo, Eds. Cham: Springer International Publishing, 2018, pp. 465–480.

[23] G. Sanchez, E. Leal, and N. Leal, "A linear programming approach for 3d point cloud simplification," *IAENG International Journal of Computer Science*, vol. 44, no. 1, pp. 60–67, 2017.

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

[25] 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, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[26] P. Scovanner, S. Ali, and M. Shah, "A 3-dimensional sift descriptor and its application to action recognition," in *Proceedings of the 15th ACM International Conference on Multimedia*, ser. MM '07. New York, NY, USA: ACM, 2007, pp. 357–360.

[27] A. Aldoma, F. Tombari, R. B. Rusu, and M. Vincze, *OUR-CVFH – Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation*. Springer Berlin Heidelberg, 2012.

[28] H. Miyajima and N. Shigei, "Initial setting of effective parameters for fuzzy modeling," *IAENG International Journal of Computer Science*, vol. 44, no. 3, pp. 375–382, 2017.

[29] M. Langovoy, "Machine learning and statistical analysis for brdf data from computer graphics and multidimensional reflectometry," *IAENG International Journal of Computer Science*, vol. 42, no. 1, pp. 22–30, 2015.

[30] H. Zhuang, M. Yang, Z. C. Cui, and Q. Zheng, "A method for static hand gesture recognition based on non-negative matrix factorization and compressive sensing," *IAENG International Journal of Computer Science*, vol. 44, no. 1, pp. 52–59, 2017.

Naveed Ahmed is an assistant professor at the Department of Computer Science, University of Sharjah. He received his PhD 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, United Kingdom, for 2 years. He is currently working as an assistant professor at the Department of Computer Science, University of Sharjah. His research interests include 3-D animation, dynamic scene reconstruction, vision-based computer graphics, and multi-view video based modeling and rendering.

Mohammed Lataifeh is an assistant professor at the Department of Computer Science, University of Sharjah. He received his PhD in design and Information Technology from the De Montfort University (Institute of Creative Technologies), UK, in 2015. He worked for several institutions as solution consultant (ERP systems, E-commerce, Graphics and Multimedia). He is currently working as an assistant professor at the Department of Computer Science, University of Sharjah. His research interests include enterprise systems, graphics, and HCI within mixed reality environments.