World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:8, No:8, 2014

Temporally Coherent 3D Animation Reconstruction from RGB-D Video Data

Salam Khalifa, Naveed Ahmed

Open Science Index, Computer and Information Engineering Vol:8, No:8, 2014 publications.waset.org/9999406/pdf

Abstract—We present a new method to reconstruct a temporally coherent 3D animation from single or multi-view RGB-D video data using unbiased feature point sampling. Given RGB-D video data, in form of a 3D point cloud sequence, our method first extracts feature points using both color and depth information. In the subsequent steps, these feature points are used to match two 3D point clouds in consecutive frames independent of their resolution. Our new motion vectors based dynamic alignement method then fully reconstruct a spatio-temporally coherent 3D animation. We perform extensive quantitative validation using novel error functions to analyze the results. We show that despite the limiting factors of temporal and spatial noise associated to RGB-D data, it is possible to extract temporal coherence to faithfully reconstruct a temporally coherent 3D animation from RGB-D video data.

Keywords—3D video, 3D animation, RGB-D video, Temporally Coherent 3D Animation.

I. Introduction

In recent years, there has been a wave of interest in reconstructing 3D animation from video data. These methods can capture dynamic shape, appearance and motion of dynamic real-world objects. Traditionally, many of these methods employed multi-view video RGB data to reconstruct a 3D animation. Most of these methods capture the real-world objects faithfully and using a number of techniques ranging from shape matching to deformation they can even capture temporally coherent animation. For all the systems that use RGB data, 3D reconstruction requires two or more cameras for the depth reconstruction. Thus the quality of the reconstruction depends on the quality of underlying image correspondence algorithms. Similarly, other tasks in the pipeline, e.g. background segmentation, require working in the RGB color space.

Recently, with the arrival of high speed depth sensors e.g. ToF sensors, it is possible to capture 3D animation using only a single camera. A depth sensor can be coupled with an RGB sensor to provide both depth and color information. Microsoft Kinect is one of the RGB-D cameras that provides the color and depth information at high frame-rate. Unlike RGB camera-based high resolution acquisition systems, the resolution of depth sensors is still very low and is marred by very high spatial and temporal noise. Therefore it poses an additional challenge to analyze and process this data using standard computer graphics, vision and geometry techniques, e.g. dynamic surface reconstruction from high frame rate depth data is one of the open research problems. Nonetheless, these cameras do provide both depth and color information at high frame rate at a very low cost, allowing us to avoid using the multi-camera acquisition setup for the depth estimation. Nevertheless, one can still employ an acquisition system comprised of multiple RGB-D camera similar to a traditional multi-view RGB acquisition setups sensors for a 360° reconstruction.

Salam Khalifa (email: salam.khalifa@gmail.com), and Naveed Ahmed (email: nahmed@sharjah.ac.ae, phone: +97165053559) are with University of Sharjah, Sharjah, United Arab Emirates.

An RGB-D video representation can be obtained from a multi-view RGB camera acquisition system or from one or more RGB-D cameras. However, this representation leads to a lack of temporal coherence between the consecutive frames of the data. Temporal coherence is an important and required property for any animation, as it is also a requirement for a number of post-processing tasks, e.g. video editing, compression, scene analysis.

In this paper, we present a new method for generating a temporally coherent 3D animation from RGB-D video data. Unlike earlier works, our method does not rely on any underlying surface representation of the dynamic scene object. Rather, our algorithm is tailored to the low resolution noisy RGB-D data provided by the state-of-the-art RGB-D video cameras, e.g. Microsoft Kinect. Our test data mainly consists of RGB-D video sequences acquired from one or more Kinects. After acquisition, we extract optical feature points from RGB data that are mapped to the depth data to obtain initial sparse 3D correspondences between two frames. Thereafter, we employ an iterative geometric matching process of feature point refinement to get an unbiased matching of 3D points. The established feature pointss mapping is then used to derive the resolution-independent global mapping between any two 3D point clouds. Afterward, we use a novel motion vectors based dynamic alignment method to track a single point cloud over the entire sequence. The result of our method is a temporally coherent 3D animation, i.e. one 3D point cloud tracked over the whole sequence. We demonstrate and validate the accuracy of our methods using several RGB-D video sequences. We have developed novel methods to quantify our method with new deformation based error metrics. Our method is analyzed and validated with varying number of parameters to show the goodness of our work.

II. Related Work

3D animation or video reconstruction using multi-view video data has been an active area of research for more than a decade. In one of the earliest and pioneering works, Carranza et al. [1] presented a method to reconstruct free-viewpoint video using the synchronized multi-view video data from eight RGB cameras. Starck et al. [2] used an acquisition system comprising of high definition cameras to capture the moving

Open Science Index, Computer and Information Engineering Vol:8, No:8, 2014 publications.waset.org/9999406/pdf

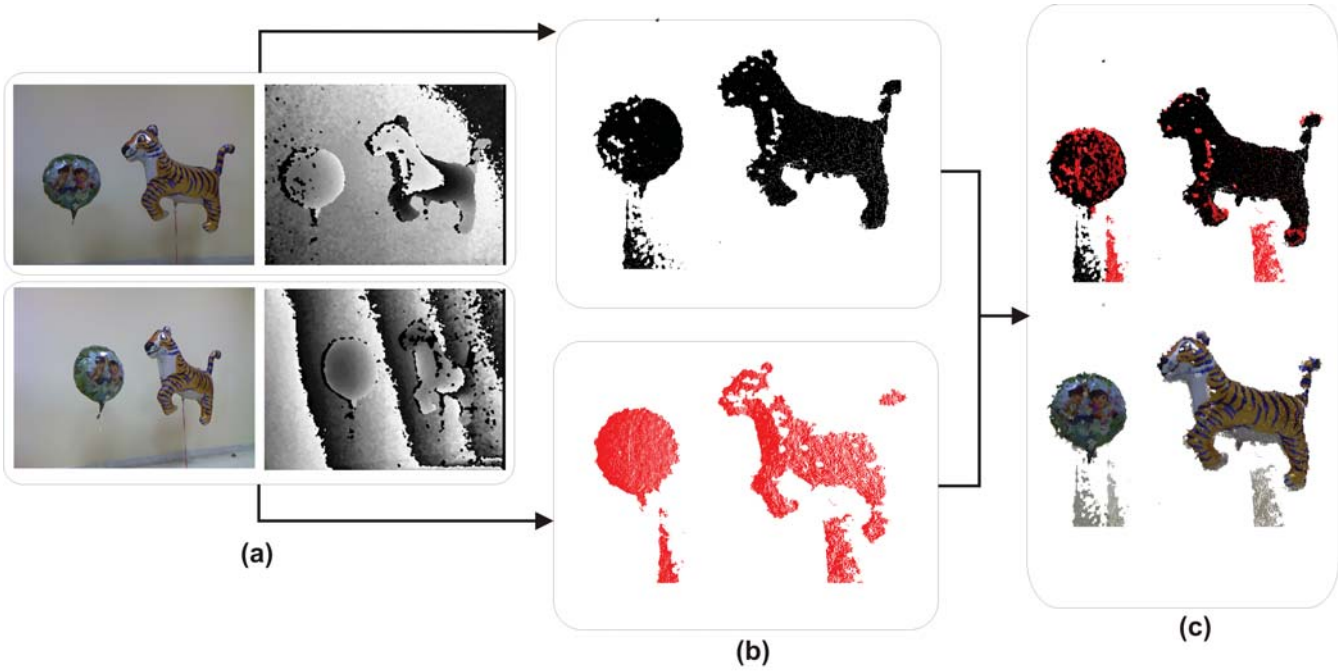


Fig. 1: (a) One frame of input RGB and Depth images from two cameras (top and bottom). RGB-D data from each camera is separately resampled in a 3D point cloud (b). (c) The point clouds are merged in a unified global coordinate system (top) with RGB mapping (bottom).

actor. They were also able to capture high frequency details of cloth deformation on the moving actor. MVV acquisition setups were extended to incorporate a large number of cameras in a number of iterations of the so-called "light-stage" by Debevec et al. [3] [4]. They captured real world subjects under a variety of static and dynamic lighting conditions. The work on free-viewpoint video by Carranza et al. [1] was extended by Theobalt et al. [5] 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 conditions 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. [6] 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. [7] where the skeleton-based deformation was employed to track the high quality template mesh over the animation. On the contrary, Ahmed et al. [8] 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. A number of other shape matching algorithm are proposed for static or dynamic 3D representations using optical or geometric features [9] [10] [11] [12] [13]. None of these methods employed depth cameras for the acquisition, and unlike these method, our work deals with noisy RGB-D video data and does not rely on any template mesh or 3D surface representation for reconstructing temporally coherent 3D animation.

With the advent of low cost depth sensors, especially Microsoft Kinect [14], there has been a wave of interest in incorporating depth sensors for acquiring static and dynamic 3D content. One of the main benefits of using Kinect is that it provides both color and depth data simultaneously at 30 frames per second, whereas the earlier works relied only on the color data where correspondences between cameras had to be used to reconstruct the depth information. Ahmed et al. [8] 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 [15].

Depth sensors have been employed in a number of applications to reconstruct a three-dimensional representation of 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. Depth data from Kinect was employed by Weiss et al. [18] 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. [19] employed a single depth camera in their pose

Open Science Index, Computer and Information Engineering Vol:8, No:8, 2014 publications.waset.org/9999406/pdf

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 work in this area was presented by Girshick et al. [20].

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. [15] 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. Berger et al. [21] employed four Kinects for unsynchronized marker-less motion capture. Recently, Ahmed et al. [22] presented an acquisition system comprising of six Kinects that can capture synchronous RGB-D data. None of these three methods [15] [21] [22] try to extract any time coherence information from the captured depth and color data.

Our main contribution is a new method to extract temporal coherence from noisy RGB-D video data using motion vector based dynamic alignment. Our method does not rely on surface representation of the depth data and being an object space method it does not requires any parametrization of the 3D point cloud. It uses both optical and geometric features for shape matching, is independent of the resolution of the 3D point cloud and can gracefully handle the difficult tracking scenarios when some part of input data is not available due to the limitations of the acquisition system. Unlike previous works, we have tested our method on sequences of multiple objects which provides unique challenges for object tracking. In addition, we have developed new error measures to validate our method and have done an extensive analysis to test the goodness of our work.

III. Data Acquisition and Calibration

Our RGB-D video acquisition system is comprised of one or more Kinects. In case of two Kinects, they are placed with an approximate angle of 90 degrees between them. While using multiple Kinects for the acquisition one has to address two issues; the synchronization and the interference between the cameras. For the synchronization, we follow the same principals as employed by Ahmed et al. [22]. Since all Kinects emit the infrared laser at the same frequency, there is bound to be interference in acquiring the depth data. Similar to [22], we do not compensate for it, rather the missing information from one camera is filled by the other or vice versa. Kinect delivers 30 fps of both RGB and depth data at 640x480 pixels per frame.

The new Kinect SDK directly provides a mapping between color and depth data, and the mapping of the depth data to real world distances. This allows us to resample each frame from of the acquired RGB-D video in the form of a 3D point cloud with the mapping of an RGB value for every 3D position. This sequence of 3D point clouds with RGB values from one or more cameras will be the main data container for all subsequent steps of our method. In practice, we use the Point Cloud Library (PCL) [23] to efficiently store the 3D point clouds and also make use of this library for the registering the point clouds from multiple cameras in a unified global coordinate system. Thus for each frame, point clouds from multiple cameras are merged into a unified single point cloud. We performed a simple depth-based segmentation for the background subtraction. Our acquisition setup, captured RGB and depth frames from Kinects, 3D point clouds from each camera and the unified 3D point cloud with an without RGB mapping can be seen in Fig. 1.

IV. Temporally Coherent 3D Animation Reconstruction

As explained in the previous section, the input to our system is a sequence of 3D point clouds with RGB mapping from one or more cameras. 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, ..., $N-1$.

The aim of our algorithm 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, ..., $N-1$. To this end, our formulation is similar to any other tracking based system. In the following sub-sections, we will describe the algorithm to obtain $\mathcal{C}_0(t)$ for any given t.

A. Estimating Optical Feature Points

For every input RGB frame $I_c(t)$ for all time steps t and cameras c, we first start by extracting the 2D SIFT feature locations [24]. For all the RGB-D video sequences that we have recorded, we obtained around 200 to 300 features for each input image. Using SIFT features has a number of benefits, mainly accuracy, stability and rotational and scale invariance. Each SIFT feature has a location $q(t) = (u, v, t)$ in the texture space, and using the formulation $(\mathcal{V}(t), \mathcal{T}(t)) \in \mathcal{C}(t)$ we can map each SIFT feature to the corresponding $p(t) \in \mathcal{V}(t)$. We denote the set of all 3D points at time t that are associated with SIFT feature points as the optical feature points $\mathcal{L}(t)$.

In the next step, we establish a mapping between $\mathcal{L}(t)$ and $\mathcal{L}(t+1)$ by finding the matching between the corresponding SIFT features using a simple Euclidean distance measure $\mathcal{D}$. This is a trivial step employed in many SIFT 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

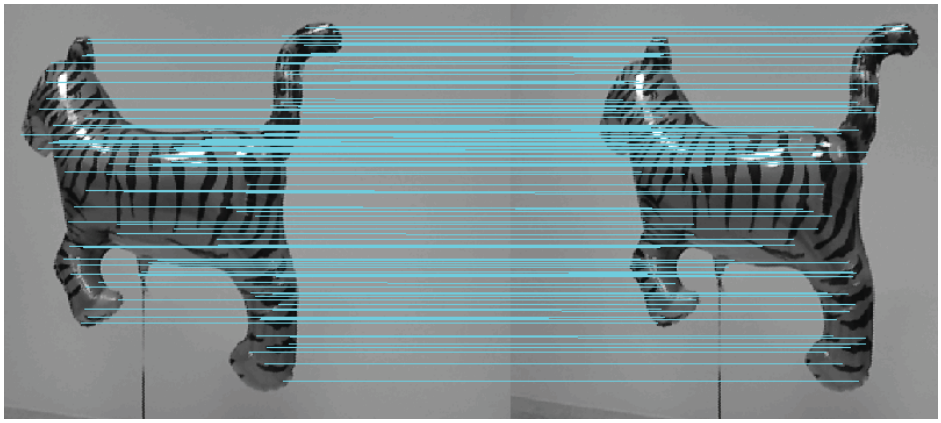


Fig. 2: Matching of optical feature points between two RGB images using SIFT.

Open Science Index, Computer and Information Engineering Vol:8, No:8, 2014 publications.waset.org/9999406/pdf

of the false positives. A visualization of the SIFT mapping can be seen in Fig. 2.

Unfortunately, the mapping from RGB to depth data is not one-to-one but one-to-many, resulting in a single $q(t)$ assigned to multiple $p(t)$ that are in a close vicinity within the 3D space. Thus, there exist multiple feature points $l(t) \in \mathcal{L}(t)$ that are associated with the same SIFT feature. If we are to match $\mathcal{C}(t)$ with $\mathcal{C}(t+1)$ using the feature point matching from $\mathcal{L}(t)$ to $\mathcal{L}(t+1)$ then this ambiguity should be resolved and one SIFT feature at t should be associated with only one $p(t)$. We resolve this ambiguity by choosing the most reliable feature point match using a novel geometrical matching algorithm explained in the next section.

B. Estimating Geometrical Feature Points Mapping

In order to resolve this ambiguity of one SIFT feature associated to multiple optical feature point matches, we propose an iterative algorithm to select the best match based on the geometrical matching. Given the optical feature points $\mathcal{L}(t)$, we define it as a set of clusters $l_s(t) \in \mathcal{L}(t)$, where all $p(t)$ in a $l_s(t)$ are associated with one SIFT feature and s=0, ..., $Number of Clusters-1$. The cluster $l_s(t)$ is matched to $l_s(t+1)$ using the distance measure $\mathcal{D}$ as explained in the previous section. Thus one-to-many mapping of a SIFT feature with $p(t)$ and $p(t+1)$ results in a many-to-many mapping between $l_s(t)$ and $l_s(t+1)$. To find the most reliable one-to-one mapping between $l_s(t)$ and $l_s(t+1)$ we use the following algorithm:

We start by selecting one $p_s(t)$ randomly from all $p(t) \in l_s(t)$ and choose its match $p_s(t+1)$ randomly from all $p(t+1) \in l_s(t+1)$. Choosing the random mapping between the clusters is not ideal, but it resolves many-to-many ambiguity and gives us an initial rough correspondence between $\mathcal{L}(t)$ and $\mathcal{L}(t+1)$ as the starting point of our algorithm. Let us denote this initial mapping as $\mathcal{M}(t)$. Given the initial correspondence $\mathcal{M}(t)$, we perform the following steps to find the best match between $l_s(t)$ and $l_s(t+1)$:

1) For the given cluster $l_s(t)$, choose three 3 space feature point positions $L_{p0}(t)$, $L_{p1}(t)$ and $L_{p2}(t)$ from $\mathcal{M}(t)$ such that $L_{p0}(t) = p_s(t)$, whereas $L_{p1}(t)$ and $L_{p2}(t)$ are chosen to be the nearest feature pont position in terms of Euclidean distance to $L_{p1}(t)$ given $L_{p0}(t)$, $L_{p1}(t)$ and $L_{p2}(t)$ are non-collinear.
2) For the cluster $l_s(t+1)$ choose three 3 space feature point positions $L_{p0}(t+1)$, $L_{p1}(t+1)$ and $L_{p2}(t+1)$ from $\mathcal{M}(t)$ under the same conditions outlined in step number 1 for $t+1$.
3) Define a plane $P(t)$ with the normal $n(t)$ using the positions $L_{p0}(t)$, $L_{p1}(t)$ and $L_{p2}(t)$.
4) Define a plane $P(t+1)$ with the normal $n(t+1)$ using the positions $L_{p0}(t+1)$, $L_{p1}(t+1)$ and $L_{p2}(t+1)$.
5) Project all $p(t) \in l_s(t)$ on $P(t)$ and obtain their parametric coordinates (a, b, t) on $P(t)$. Root point of the plane is chosen randomly from $L_{p0}(t)$, $L_{p1}(t)$ and $L_{p2}(t)$.
6) Project all $p(t+1) \in l_s(t+1)$ on $P(t+1)$ and obtain their parametric coordinates $(a, b, t+1)$ on $P(t+1)$. Root point of the plane is chosen randomly from $L_{p0}(t+1)$, $L_{p1}(t+1)$ and $L_{p2}(t+1)$.
7) A new match between $l_s(t)$ and $l_s(t+1)$ is now defined to be between the two points $p_s(t)$ and $p_s(t+1)$ that have the least distance in terms of their parametric coordinates (a, b, t) and $(a, b, t+1)$.
8) Update the mapping $p_s(t)$ and $p_s(t+1)$ in $\mathcal{M}(t)$.
9) Repeat for all clusters until the matches stabilize.

The result of the geometric matching algorithm is a one-to-one correspondence $\mathcal{M}(t)$ between $\mathcal{L}(t)$ and $\mathcal{L}(t+1)$ and subsequently this gives a direct correspondence between $\mathcal{C}(t)$ with $\mathcal{C}(t+1)$. Our algorithm is inspired by the work of Tevs et al. [9]. Thus we obtain a correct sparse matching of two frames using a geometric based mapping algorithm which uses color based matching as the starting point. We validated our geometric matching algorithm by estimating temporal coherence with and without the algorithm. More discussion can be seen in the Results and Validation section (Sect. V).

Even though we have obtained a feature points based correspondence between $\mathcal{C}(t)$ with $\mathcal{C}(t+1)$, it is still not enough to align the two point clouds, because we only get 200 to 300 feature points matches, whereas the number of points in the point cloud is more than 60,000. In the next section, we will explain our method for global alignment of point clouds using the feature points based correspondence.

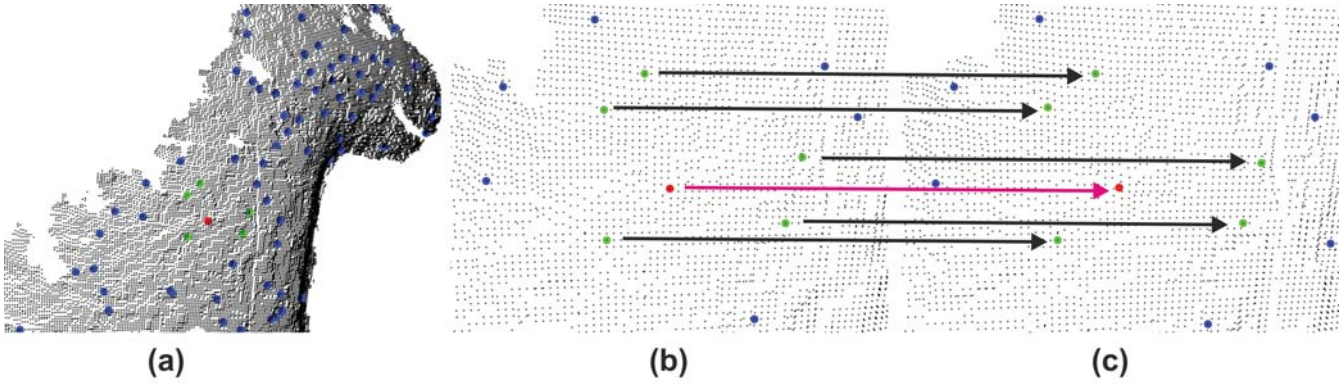


Fig. 3: (a) Zoomed-out point cloud with feature points shown in blue and green. Red point at time-step t is to be matched, and green points are the five nearest feature points. (b) Shows the zoomed-in point cloud at t. Motion vectors are calculated with respect to the 5 nearest feature points. These motion vectors are used to calculate the matching point at $t+1$ as shown in (c) and explained in (Sect. IV-C). Note that the matching point (red) in (c) is not centered on any point because the matching is resolution independent.

Open Science Index, Computer and Information Engineering Vol:8, No:8, 2014 publications.waset.org/9999406/pdf

C. Alignment using Motion Vectors

In order to align $\mathcal{C}(t)$ with $\mathcal{C}(t+1)$ we need to find the mapping for $\mathcal{V}(t) \in \mathcal{C}(t)$, whereas the featuer points based correspondence gives us a sparse matching $\mathcal{M}(t)$. To establish the mapping for all $p(t) \in \mathcal{V}(t)$ that are not associated with any feature point in $\mathcal{M}(t)$ we use the following algorithm:

1) Find $\mathcal{N}$ nearest feature point positions $L_n(t)$ in terms of Euclidean distance to $p(t)$, where n=0, ..., $\mathcal{N}-1$.
2) Find the mapping of $L_n(t)$ to $t+1$, which is $L_n(t+1)$ using $\mathcal{M}(t)$.
3) Find motion vectors $V_n(t)$ from $L_n(t)$ to each 3 corresponding 3 space position in $L_n(t)$, i.e. $V_n(t)$ = $L_n(t+1)$-$L_n(t)$.
4) Find the average motion vector $V_p(t)$ for $p(t)$ by summing up all $V_n(t)$ and dividing by $\mathcal{N}$.
5) The match $p(t+1)$ of $p(t)$ is found using $p(t+1)$=$p(t)$+ $V_p(t)$, i.e. the matching point lies at the same distance with respect to the average motion of $L_n(t+1)$ to $L_n(t)$.

A visualization of feature points and the alignment algorithm can be seen in Fig. 3. We can justify our global alignment algorithm under the assumption that the given an arbitrary motion of the dynamic object, the deformations will be largely isometric. Obviously for extreme non-isometric deformations, where points collapse on each other, our algorithm will not hold but the same is true for any temporal shape matching algorithm. On the other hand, we validated our algorithm on a number of data sets and were able to extract time coherence with remarkable accuracy. Even in the extreme cases where the depth data has holes in some areas due to noise or the limitation of the depth sensor and no motion information can be inferred for that area, our method is able to track the motion using the nearest feature points gracefully. The value of $\mathcal{N}$ is found through experiments and more discussion can be seen in the Results and Validation section (Sect. V).

Given the establishment of the alignment between $\mathcal{C}(t)$ and $\mathcal{C}(t+1)$, our tracking algorithm starts from t=0 and map $\mathcal{C}(0)$ to $\mathcal{C}(1)$ yielding $\mathcal{C}_0(1)$ that is $\mathcal{C}(0)$ aligned with $\mathcal{C}_0(1)$ using the feature point matching $\mathcal{M}(0)$ between $\mathcal{L}(0)$ and $\mathcal{L}(1)$. At the next step $\mathcal{C}_0(1)$ is aligned with $\mathcal{C}(2)$ yielding $\mathcal{C}_0(2)$ using the feature point matching $\mathcal{M}(1)$ between $\mathcal{L}(1)$ and $\mathcal{L}(2)$. Thus at every subsequent step of the algorithm the tracked point cloud $\mathcal{C}_0(t)$ is aligned with $\mathcal{C}(t+1)$ yielding $\mathcal{C}_0(t+1)$ using the feature point matching $\mathcal{M}(t)$ between $\mathcal{L}(t)$ and $\mathcal{L}(t+1)$. Thus our method tracks $\mathcal{C}(0)$ over the whole sequence, resulting in a temporally coherent 3D animation.

V. Results and Validation

We apply our temporally coherent 3D animation reconstruction method on three real-world data sets. The data sets are acquired from our RGB-D video acquisition system (Sect. III). The first two of the three RGB-D data sets are recorded using 2 Kinects, while the third was recorded using just one Kinect. The first sequence shows a floating tiger balloon in the air depicting interesting motion without significant deformations. Second data set is similar to the first but this time with two dynamic objects, tiger and Dora. While the third data set is a floating Pooh balloon. All sequences are 100 frames long. It should be noted that the RGB data from Kinect is very low resolution but we are still able to correctly match the feature points from RGB to depth data. On average for the first two data sets we had on average 300 feature point matches between two consecutive frames, whereas for the third data set there were only 50 feature point matches for each frame. Average number of points in each camera for the first two sequences is 60,000, while for the third sequence is 25,000. It can be seen in Fig. 4 and the accompanying video that our method can convincingly reconstruct temporally coherent 3D animation from noisy RGB-D data. It is evident from the non-coherent data that our method is able to extract temporal coherence even in the presence of large spatial and temporal noise, which is especially evident by the missing depth data in the RGB-D video streams causing holes in the point clouds. More results can be seen in the accompanying video. It should be noted that due to the space limitations of the paper and the size limitation of the video we cannot show all of the sequences.



Fig. 4: (a) Two frames (zoomed-in) of the non-coherent tiger balloon RGB-D sequence. It can be seen that the point cloud changes dramatically from one frame to other, e.g. face. (b) Shows the same two frames tracked using our temporally coherent 3D animation reconstruction method, in which the point cloud is tracked without significant distortions. More results can be seen in the accompanying video.

Open Science Index, Computer and Information Engineering Vol:8, No:8, 2014 publications.waset.org/9999406/pdf

Even though the visual analysis of our sequences provides a good evidence of the robustness of our method, we also perform a quantitative analysis on the quality of the temporally coherent 3D animation to validate different steps of our algorithm. We wanted to test if the geometric feature point mapping step is improving the tracking, and similarly how does the number of nearest feature points $\mathcal{N}$ impact the tracking results and what would be the good value for $\mathcal{N}$. For the quantitative analysis, since we do not have any ground truth data to compare against, we have developed a distortion measure to check the quality of reconstructed 3D animation under different initial conditions.

The main idea is to measure the tangential distortion by comparing the distances between a small set of points at each frames under the assumption that dynamic object goes through low deformation. We achieve this 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 i=0, ..., $\|\mathcal{E}\| - 1$ and $\|\mathcal{E}\|$ is the total number of vectors in $\mathcal{E}_i(0)$. After tracking, we calculate the same distance vectors $\mathcal{E}_i(t)$ 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:

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

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

$$E = \frac{\sum_{t=1}^{N-1} E_i(t)}{N-1} \quad (2)$$

We use the average error measure E to find out the optimal value of $\mathcal{N}$ in terms of low distortion and tracking quality. It is also used to validate our geometric feature point mapping algorithm. As explained in Sect. IV-C the alignment algorithm looks for $\mathcal{N}$ nearest features to construct the vector field for each $p(t)$. The value of $\mathcal{N}$ depends a lot on the type of motion and the shape of the object. For example, if the object is animated by a global transformation then increasing the value of $\mathcal{N}$ will not induce significant errors rather it will normalize the motion resulting in the reduced average error for each match. On the other hand if in addition to some global motion, individual areas of the object also experience local motion, e.g. parts of the body moving independently, then increasing the value of $\mathcal{N}$ beyond a certain threshold will result in the incorrect animation.

TABLE I: Average error comparison

$\mathcal{N}$	Geom. Map	No Geom. Map
1	4.73%	6.46%
3	3.91%	4.16%
5	2.93%	3.33%
10	2.55%	2.76%
15	2.14%	2.25%
20	1.90%	1.97%
30	1.80%	1.86%

For the balloon sequences, due to less local motion, we observe that the increasing value of $\mathcal{N}$ results in smaller average error for the whole sequence. Table I shows the average error for different values of $\mathcal{N}$ for this first sequence with one tiger balloon. As can be seen in the table that the average error for 10 nearest feature points is around 2.55%, whereas for 20 nearest feature points it is around 1.90%. It can be seen that the error does not reduce linearly with respect to the increasing value of $\mathcal{N}$. Although the higher value of $\mathcal{N}$ reduces the overall error, it also results in the normalization of the motion. Thus all the results in the video for balloon sequences are generated with $\mathcal{N}$=10. We address this issue below in the discussion of the limitations of our method.

Finally we validated the geometric feature point mapping by looking at the average error per frame and for the complete sequence with and without the geometric feature point mapping. Table I shows that on average geometrical feature point mapping results in reducing the error by 0.2% to 0.4%. Fig. 5 shows the average error plot for each frame of the RGB-D sequence with one tiger balloon with and without geometrical feature point mapping for $\mathcal{N}$=5.

Our method is computationally very efficient, on average we can reconstruct a temporally coherent animation at the rate of 20 frames per minute. Thus it takes around 5 minutes to process a 100 frames sequence on a dual core 2.4 Ghz Core i5 system.

Our method is subject to a couple of limitations. One of the limitations is that we have to rely on Euclidean distance

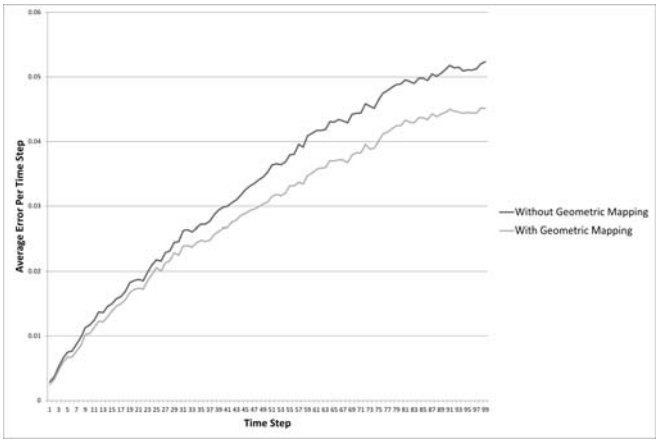


Fig. 5: Average error per time-step with and without geometric feature point mapping algorithm for $\mathcal{N}$=5.

measures in all the steps of our algorithm because we do not try to reconstruct any surface representation from the point cloud data. This is partially a hardware induced limitation as the depth data from Kinect is extremely noisy, and even though there has been recent work in using Kinect as a laser scanner for static objects, estimating surface of dynamic objects remains a challenging problem. The other limitation, as we mentioned above is not limiting the search for nearest feature points within a local region of similar motion. A data set with very fast motion and high local motion or deformations will be very challenging to handle without a local search. This is not a principal limitation of our method because a local search can be integrated independently without modifying the actual algorithm but is an area that we would definitely like to address in future. Additionally we would like to add more quantitative analysis for the validation, e.g. a silhouette-based error measure or a bounding box error measure. These error measures are challenging for Kinect based RGB-D data because of very high spatial and temporal random noise, which results in a very large variation of either measurement at each time step. Therefore it is not straightforward to make a direct comparison between input and tracked sequences using these error measures.

Despite the limitations we have presented an efficient method for reconstructing a temporally coherent 3D animation from single or multi-view RGB-D video.

VI. Conclusions

We presented a new method to reconstruct a temporally coherent 3D animation from single or multi-view RGB-D video data using unbiased feature point sampling. Starting from RBG-D video data resampled in the form of 3D point clouds with the RGB color mapping, we first find optical feature points that map two consecutive frames of the sequence. The feature points mapping is refined by a new geometrical mapping algorithm which is then used to derive the resolution-independent global mapping of all the points in a point cloud to the next frame. The result of our work is a single point cloud tracked over the complete animation sequence. We tested our method on data recorded from one or more Kinects. The resulting temporally coherent 3D animation can be used in a number of tasks, e.g. video editing, compressions or dynamic scene analysis.

References

[1] Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 22(3):569–577, 2003.
[2] Jonathan Starck and Adrian Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.
[3] Paul E. Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *SIGGRAPH*, pages 145–156, 2000.
[4] Tim Hawkins, Per Einarsson, and Paul E. Debevec. A dual light stage. In *EGSR*, pages 91–98, 2005.
[5] Christian Theobalt, Naveed Ahmed, Gernot Ziegler, and Hans-Peter Seidel. High-quality reconstruction of virtual actors from multi-view video streams. *IEEE Signal Processing Magazine*, 24(6):45–57, 2007.
[6] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, 27(3), 2008.
[7] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popovic. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27(3), 2008.
[8] Naveed Ahmed, Christian Theobalt, Christian Rössl, Sebastian Thrun, and Hans-Peter Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video. In *CVPR*, 2008.
[9] Art Tevs, Alexander Berner, Michael Wand, Ivo Ihrke, and Hans-Peter Seidel. Intrinsic shape matching by planned landmark sampling. In *Eurographics*, 2011.
[10] Peng Huang, Adrian Hilton, and Jonathan Starck. Shape similarity for 3d video sequences of people. *International Journal of Computer Vision*, 89(2-3):362–381, 2010.
[11] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Tosiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01*, pages 203–212, New York, NY, USA, 2001. ACM.
[12] Cedric Cagniart, Edmond Boyer, and Slobdodan Ilic. Iterative mesh deformation for dense surface tracking. In *ICCV Workshops*, ICCV'09, 2009.
[13] Kiran Varanasi, Andrei Zaharescu, Edmond Boyer, and Radu Horaud. Temporal surface tracking using mesh evolution. In *ECCV'08*, pages 30–43, Berlin, Heidelberg, 2008.
[14] MICROSOFT. Kinect for microsoft windows and xbox 360. http://www.kinectforwindows.org/, November 2010.
[15] Y. M. Kim, D. Chan, Christian Theobalt, and S. Thrun. Design and calibration of a multi-view tof sensor fusion system. In *CVPR Workshop*, 2008.
[16] Y. M. Kim, Christian Theobalt, J. Diebel, J. Kosecka, B. Micusik, and S. Thrun. Multi-view image and tof sensor fusion for dense 3d reconstruction. In *3DIM*, pages 1542–1549, Kyoto, Japan, 2009. IEEE.
[17] Victor Castaneda, Diana Mateus, and Nassir Navab. Stereo time-of-flight. In *ICCV*, 2011.
[18] Alexander Weiss, David Hirshberg, and Michael J. Black. Home 3d body scans from noisy image and range data. In *ICCV*, 2011.
[19] Andreas Baak, Meinard Muller, Gaurav Bharaj, Hans-Peter Seidel, and Christian Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *ICCV*, 2011.
[20] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *ICCV*, 2011.
[21] Kai Berger, Kai Ruhl, Yannic Schroeder, Christian Bruemmer, Alexander Scholz, and Marcus A. Magnor. Markerless motion capture using multiple color-depth sensors. In *VMV*, pages 317–324, 2011.
[22] Naveed 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.
[23] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *ICRA*, 2011.
[24] David G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, pages 1150–1157, 1999.

Open Science Index, Computer and Information Engineering Vol:8, No:8, 2014 publications.waset.org/9999406/pdf