2012 IEEE International Conference on Multimedia and Expo

Virtual View Reconstruction Using Temporal Information

Shujie Liu[1], Philip A. Chou[2], Cha Zhang[2], Zhengyou Zhang[2], Chang Wen Chen[1]
[1]State University of New York at Buffalo
Buffalo, New York, USA
{sl252, chencw}@buffalo.edu
[2]Microsoft Research
Redmond, Washington, USA
{pachou, zhang, chazhang}@microsoft.com

Abstract— **The most significant problem in generating virtual views from a limited number of video camera views is handling areas that have become dis-occluded by shifting the virtual view away from the camera view. We propose using temporal information to address this problem, based on the notion that dis-occluded areas may have been seen by some camera in some previous frames. We formulate the problem as one of estimating the underlying state of the object in a stochastic dynamical system, given a sequence of observations. We apply the formulation to improving the visual quality of virtual views generated from a single "color plus depth" camera, and show that our algorithm achieves better results than depth image based rendering using standard inpainting.**

Keywords-3D Video, Non-rigid Tracking, Virtual View, Temporal Correlation

I. INTRODUCTION

Reconstructing virtual (or novel) views of a scene or object from a limited number of camera views is a key problem in immersive multimedia applications such as 3D video conferencing and free viewpoint video [1], in which seamless viewpoint changes are necessary based on viewers' movements.

Image based rendering (IBR) is a common paradigm for reconstructing virtual views. One of the basic ideas in IBR is that a virtual view between two (or more) captured views can be generated by estimating the disparity between the captured views, linearly interpolating the disparity, and performing disparity compensation [2].

A form of IBR is depth image based rendering (DIBR). In essence, DIBR generates a virtual view by converting a depth map into a disparity appropriate for the virtual view, and then performing disparity compensation. The depth map is obtained either by estimating depth directly from the color camera views (e.g., using stereo matching), or is supplied externally (e.g., from a depth camera, using time-of-flight or structured light) [3][4].

Regardless of the method used for reconstructing virtual views from captured camera views, shifting the virtual viewpoint away from the viewpoint of any camera is likely to cause areas of the scene that used to be occluded (unseen by any camera) to become visible in the new viewpoint. Because these dis-occluded areas are unseen by any camera, it is difficult to reconstruct them in the virtual view using IBR or DIBR. Special techniques must be developed.

A common solution to the dis-occlusion problem is to simply capture the scene with more cameras. However, this may be costly and impractical. For example, for immersive conferencing, it is infeasible to blanket each participant with a dense array of cameras, considering the huge initial and maintenance cost.

Another solution, if the virtual viewpoint is not too far from the viewpoint of the camera, is to reconstruct the small holes caused by the dis-occlusion with inpainting, such as extrapolating into those areas from neighboring areas [5]. Such inpainting is widely used when the holes are small. However, when the virtual viewpoint varies widely, as would be typical for immersive multimedia applications such as free viewpoint video, the dis-occluded areas become holes too large for inpainting.

Another solution, which can be used when the dis-occlusions are large, is to use model-based reconstruction. For example, a model of the human head can be fitted to the camera views, and regions of the fitted model unseen by any camera can then be used to fill in dis-occlusion holes in the virtual views [6][7]. Of course model-based solutions work well only for parts of the scene that are well-modeled, as might be the case with a human head. Objects that are not yet well-modeled, including the clothed human body, cannot be handled well by this method. Hybrids are also possible, which partially compensate for this [8].

In this paper, we propose using historical information previously captured by the cameras to handle dis-occlusions. In cases of a moving scene, such as a scene of a human body moving in a 3D video conference, it is possible that dis-occluded areas may have been seen by some camera in some previous frame. For example, if the human turns his body in front of a single frontal camera, the side of his body becomes visible to the camera. That information can later be used to generate a virtual view of the side of his body at time t even if that area is not visible to the camera at time t.

Because dis-occluded areas are generally part of a non-rigid moving object (such as the human body), particular care must be taken when using historical information to fill a hole caused by dis-occlusion. Generally, data from the previous frame(s) must be warped to match the current frame before using the information. We choose to perform the warping on the underlying 3D object rather than in 2D, for maximum accuracy.

Warping a 3D object so that it registers with another 3D object is a well-studied problem. (For an overview, see [9].) A popular family of algorithms for registering 3D objects is based on the iterative closest point (ICP) matching method for rigid motion (or correspondence) [10][11]and its extension to non-rigid motion [12]. For non-rigid motion,

978-0-7695-4711-4/12 $26.00 © 2012 IEEE
DOI 10.1109/ICME.2012.194

these algorithms find a collection of locally rigid or affine transformations defined between corresponding or matching points or patches, e.g., [12][13][14]. This collection of transformations is sufficient to warp one 3D object onto another, in the regions where there are matching points.

Unfortunately, for our application, there may be large regions of one object with no matching points in the other, and moreover, the unmatched regions must be appropriately warped along with the matched regions. For example, suppose the subject is captured in 3D by a single depth camera from the front, but the virtual view is from the side, at the current time. Among the historically captured data, the front surface of the subject's body might have a good match to the data currently captured by the camera, but the back surface would not. However, in order to use the historical information to help generate the virtual view, it is important to warp the back surface along with the front so that it is consistent enough to generate the appropriate virtual view.

Therefore, in contrast to previous approaches, we develop an approach that defines the warping transformation not only near the local surfaces for which there is a match, but across the whole volume or domain of the object, so that the whole object can be warped together. Specifically, we define a class of warpings that are piecewise constant transformations over the cells of a hierarchical partition of the entire scene volume, where the transformation within each cell is rigid. We define a prior probability distribution over this class (shallow hierarchies being both more likely and more smooth), and we estimate the most likely warping consistent with the camera data.

Overall, we do not warp individual previous frames to the current frame, but rather build and maintain a single coherent non-parametric model of the object over time. We warp this model, every frame, to match the data in the new frame, before integrating it with the data in the new frame. We regard this problem as one of estimating the underlying state of the object in a stochastic dynamical system, given a sequence of observations.

We apply our approach to improving the visual quality of virtual views generated from a single "color plus depth" camera, and show that our algorithm is not only able to fill dis-occluded areas, but is also able to smooth sensor noise. Simulation results show that the algorithm achieves better results than depth image based rendering methods using standard inpainting.

The rest of the paper is organized as follows. Section 2 discusses our camera assumptions. Section 3 presents the algorithm to improve 3DV quality with temporal information integration. Section 4 includes simulation results and analysis while Section 5 concludes this paper.

II. Sensor Density And Depth Cameras

Reconstructing an arbitrary view of a scene generally requires many camera views. Even in the absence of occlusions, the minimum sampling rate of the light field in terms of camera views is proportional to the bandwidth of the scene, the focal length of the cameras, and the range of disparities $h = 1/z_{min} - 1/z_{max}$, where z_{min} and z_{max} are the minimum and maximum depths in the scene, assuming Lambertian surfaces [15]. If $z_{min} = z_{min}(x, y)$ and $z_{max} = z_{max}(x, y)$ are per-pixel depth uncertainties (e.g., as can be determined by a depth camera), the camera view sampling can be greatly reduced [16]. In the limit of perfect knowledge of depth, a single camera view can suffice to reproduce the scene from any direction, in the absence of occlusions. Indeed, single "video plus depth" (or more precisely, "color plus depth") cameras are used in this way to capture the texture and geometry of objects in order to render them from arbitrary points of view using DIBR. An example of the signal captured from a color plus depth camera is shown in Fig. 1. In reality there is occlusion, surfaces are not Lambertian, and depth measurements have noise, so multiple cameras may need to be used [17]. In this paper, we use a single color plus depth camera, the Microsoft Kinect, whose depth errors can be modeled as independent Gaussian noise with variance proportional to the fourth power of the distance from the camera [18].

Figure 1. Example of captured color plus depth.
(Left: color video frame; Right: depth map)

III. Proposed Method

In this section, we first describe the mathematical framework for our approach, and then introduce the proposed algorithm in detail.

A. Mathematical Framework

We formulate the problem of reconstructing a virtual view from the sequence of current and past "color plus depth" images as the problem of estimating the underlying state of an object from a sequence of observations in a stochastic dynamical system.

In the stochastic dynamical system, let the observation vector $\mathrm{y_t}$ be a concatenation of the color and depth images at time t (from one or more frame-synchronized cameras), and let the state vector $\mathrm{x_t}$ be any representation of the colored geometry of the object (or scene) at time t, from which it is possible to render virtual views.

Let the output distribution $P(y_t|x_t)$ be the probability that observation vector $\mathrm{y_t}$ is generated from underlying object x_t. The output distribution $P(y_t|x_t)$ is determined by rendering the object $\mathrm{x_t}$ into the appropriate camera view(s) and adding random sensor error. The random sensor error is an independent zero-mean Gaussian noise with variance proportional to the fourth power of the depth.

Let the state transition matrix $P(x_t|x_{t-1})$ be the probability that the object is in state $\mathrm{x_t}$ at time t given that it was in state x_{t-1} at time $t-1$. The conditional probability $P(x_t|x_{t-1})$ is determined by warping x_{t-1} to x_t with a random warping W_t. The random warping W_t is modeled as

a random tree with random binary splits at each interior node and random rigid transformations on each branch (described in more detail later).

The problem is, for each time t, first to find the state x_t^* maximizing the posterior distribution of x_t given all previous observations $y_t, y_{t-1}, \dots, y_1$ and the initial state x_0,

$$x_t^* \stackrel{\text{def}}{=} \arg\max P(x_t | y_t, \dots, y_1, x_0), \quad (1)$$

(i.e., the MAP estimate of x_t), and then to render x_t^* into the desired virtual view.

The MAP estimate (1) can be computed by Bayes' rule, $x_t^* = arg\,max\,\alpha_t(x_t)/P(y_t, \dots, y_1) = arg\,max\,\alpha_t(x_t)$, where:

$$\begin{aligned}\alpha_t(x_t) &\stackrel{\text{def}}{=} P(x_t, y_t, \dots, y_1 | x_0) \\ &= \textstyle\sum_{x_{t-1}} P(x_t, x_{t-1}, y_t, \dots., y_1 | x_0) \\ &= \textstyle\sum_{x_{t-1}} P(y_t|x_t) P(x_t|x_{t-1}) P(x_{t-1}, y_{t-1}, \dots., y_1 | x_0) \\ &= P(y_t|x_t) \textstyle\sum_{x_{t-1}} P(x_t|x_{t-1}) \alpha_{t-1}(x_{t-1}) \end{aligned} \quad (2)$$

Here we assume x_t depends on x_{t-1} only and use Σ rather than $\int dP$ for notational simplicity. Equation (2) is the Forward Algorithm used for inference in Hidden Markov Models [19]. It is also the combined predict/update formula for Kalman filtering when x_t is a first-order Gaussian auto regressive process and y_t is a linear function of x_t with additive Gaussian noise.

Since the number of possible values for a state x_{t-1} is very large (in fact, may be infinite), and (2) is hard to compute, we take steps to approximate it. First, we replace the summation/integration in (2) by maximization:

$$\tilde{\alpha}_t(x_t) \stackrel{\text{def}}{=} P(y_t|x_t)\, max_{x_{t-1}} P(x_t|x_{t-1}) \tilde{\alpha}_{t-1}(x_{t-1}) \quad (3)$$

which is the Viterbi Algorithm [19]. Although (3) has lower complexity than (2), it still requires storage and access to all values of $\tilde{\alpha}_{t-1}(x_{t-1})$. However, if we restrict attention to the particular $\hat{x}_{t-1}$ that maximized the previous step, then we obtain the Greedy Algorithm,

$$\hat{\alpha}_t(x_t) \stackrel{\text{def}}{=} P(y_t|x_t) P(x_t|\hat{x}_{t-1}) \hat{\alpha}_{t-1}(\hat{x}_{t-1}) \quad (4)$$

where

$$\begin{aligned}\hat{x}_t &\stackrel{\text{def}}{=} arg\,max_{x_t}\, \hat{\alpha}_t(x_t) \\ &= arg\,max_{x_t}\, P(y_t|x_t) P(x_t|\hat{x}_{t-1}) \\ &= arg\,max_{W_t}\, P\big(y_t|W_t(\hat{x}_{t-1})\big) P(W_t). \end{aligned} \quad (5)$$

Thus, having fixed the maximizing state $\hat{x}_{t-1}$ at time $t-1$, the problem of finding the maximizing state $\hat{x}_t$ at time t is simplified to finding the warping W_t that balances both the desire to match the data (i.e., to maximize $P(y_t|W_t(\hat{x}_{t-1}))$) and the desire to be smooth (i.e., to maximize $P(W_t)$).

B. Proposed Algorithm: Overview

Based on the above framework, we design the algorithm as illustrated in Fig. 2. At each time step t, first we warp our estimate $\hat{x}_{t-1}$ of the object at time $t-1$ to match the current measurement data y_t, and then we integrate y_t and the warped estimate $W_t(\hat{x}_{t-1})$ to obtain the estimate $\hat{x}_t$ of the object at time t. Finally we render the estimate $\hat{x}_t$ of the current object into the desired virtual view, before feeding it back as the estimate for the next time step.

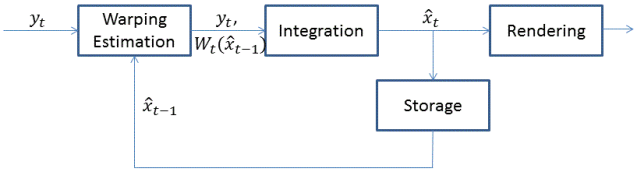

Figure 2. Algorithm overview.

Though various representations could be used (meshes, depth maps, signed distance functions on a volume, etc.), in this paper we use a colored point cloud representation for both $\hat{x}_t$ and y_t . That is, each is represented as an unstructured set of colored points $\{p\}$, where each colored point $p = (x, y, z, r, g, b)$ has both position in world coordinates and color information. (This could also be extended to contain other contextual information such as gradients, normals, and patches.) The number of points in y_t is the number of pixels in the captured depth (or color) image, while the number of points in $\hat{x}_t$ is the number of non-empty voxels in a discretization of a relevant 3D volume of the scene. (We maintain at most one point per voxel.) In addition, we maintain for each point p in $\hat{x}_t$ a weight $w_p \in [0,1]$ representing a confidence.

C. Proposed Algorithm: Warping Estimation

Given the current "depth plus color" measurements y_t and the estimate $\hat{x}_{t-1}$ of the object state at time $t-1$, we wish to find the warping W_t that maximizes $P(y_t|W_t(\hat{x}_{t-1}))P(W_t)$ as in (5), thus making the matching error as small as possible while keeping the warping as simple as possible. To be precise, the warping W_t: $\mathbb{R}^3 \to \mathbb{R}^3$ is a mapping that takes any 3D position at time $t-1$ and maps it to some 3D position at time t. $W_t(\hat{x}_{t-1})$ denotes the point cloud obtained by applying W_t to the position of each point in the point cloud of $\hat{x}_{t-1}$. Finding the optimum W_t is equivalent to minimizing $-\log P(y_t|W_t(\hat{x}_{t-1})) - \log P(W_t)$. The first term denotes the sum of the squared distances between each point in y_t and the closest point in $W_t(\hat{x}_{t-1})$, weighted by the inverse of the variance of the sensor error (where distance is measured both in Euclidean and color space). To be precise, the first term is

$$D = \textstyle\sum_{q \in y_t} \min_{p \in \hat{x}_{t-1}} d(q, W_t(p)) \quad (6)$$

where for $q = (x, y, z, r, g, b)$ and $p' = (x', y', z', r', g', b')$,

$$\begin{aligned} d(q, p') = {} & w_x \|x - x'\|^2 + w_y \|y - y'\|^2 + w_z \|z - z'\|^2 \\ & + w_r \|r - r'\|^2 + w_g \|g - g'\|^2 + w_b \|b - b'\|^2. \end{aligned} \quad (7)$$

The second term, $-\log P(W_t)$, can be considered the number of bits R needed to encode W_t. Thus finding the optimal warping is analogous to a Distortion-Rate optimization.

We assume a class of warpings each representable by a binary hierarchical partition of $\mathbb{R}^3$ with splits determined by planes in $\mathbb{R}^3$ and rigid transformations at the leaf cells. Thus a warping is analogous to a regression tree. Points in $\hat{x}_{t-1}$ are dropped down the tree, falling to the left or right of planes at

each interior node, and rigid transformations at the leaves are applied to the positions of the points to estimate y_t. A hierarchical warping tree is exemplified in Fig. 3.

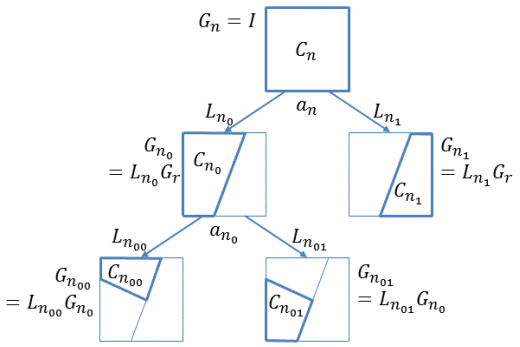

Figure 3. Example of hierarchical warping tree.

In the tree, $C_n \subseteq \mathbb{R}^3$ is the region represented by node n. L_n is the local rigid transformation applied at node n, represented by rotation matrix R_n and translation vector t_n. G_n is the composition of local transformations along the path from the root to node n. If node n is an interior node with children n_0 and n_1, then a_n is the plane that splits C_n into sub-regions C_{n_0} and C_{n_1} corresponding to n_0 and n_1.

This class of warpings is a natural fit to hierarchical, articulated motion. One interpretation of Fig. 3 is: respectively, C_n, C_{n_0}, and $C_{n_{00}}$ are the volumes occupied by the human body, arm, and foream; a_n and a_{n_0} split arm from body and forearm from arm; L_{n_0} and $L_{n_{00}}$ are local coordinate transforms from arm to body and from forearm to arm; and $G_{n_{00}}$ is the world coordinate transform for the forearm.

To optimize the warping, we use as a surrogate for $-\log P(y_t|W_t(\hat{x}_{t-1}))$ the sum of the squared distances between each point in $W_t(\hat{x}_{t-1})$ and its closest point in y_t (note the order reversal compared to (6)) weighted by the inverse of the variance of the sensor error *and* by the confidence w_p of the points p in $\hat{x}_{t-1}$, namely

$$D = \sum_{p \in \hat{x}_{t-1}} w_p \min_{q \in y_t} d(q, W_t(p)). \quad (8)$$

The confidence w_p of a point $p \in \hat{x}_{t-1}$ is set to zero or close to zero if it likely that $W_t(p)$ will not be visible in y_t, i.e., if it will be occluded or out of frame. This can be predicted by rendering $\hat{x}_{t-1}$ in the previous step.

Now the contribution to D of each leaf n in a tree is

$$D_n(G_n) = \sum_{p \in \hat{x}_{t-1} \cap C_n} w_p \min_{q \in y_t} d\big(q, G_n(p)\big), \quad (9)$$

so the warping can be optimized by growing the tree from the root as follows: Set node n to the root and apply the following recursive procedure to n:

1) Perform a robust iterative closest point (ICP) algorithm to match the points $p \in \hat{x}_{t-1} \cap C_n$ to y_t by finding G_n to minimize $D_n(G_n)$,
2) Find a candidate split a_n to split n into child nodes n_0 and n_1. (Return if no adequate split is found.)
3) Optimize $D_{n_0}(G_{n_0})$ and $D_{n_1}(G_{n_1})$ as in step (1),
4) Comparing the decrease in distortion $\Delta D = D_n(G_n) - D_{n_0}(G_{n_0}) - D_{n_1}(G_{n_1})$ to the increase in rate $\Delta R = constant$ needed to describe the split,
5) Return without splitting (making node n a leaf) if $\Delta D \leq \Delta R$; else split the node using the candidate split a_n and recursively perform the procedure on n_0 & n_1.

In Step (1), an extension of the registration algorithm in [10][11] is used to estimate the transformation between two point clouds, while a classification algorithm is developed for splitting in Step (2). These algorithms are briefly described in the following sub-sections 3.3.1 and 3.3.2.

1) Robust Iterative Closest Point Matching

With minimizing (9) as the objective, set $j = 0$, set $G_n^j = G_{parent(n)}$ (or $G_n^j = I$ if n is the root), and set $d_{thr}^j = d_0$. Then

1) $\forall p \in \hat{x}_{t-1} \cap C_n$, find $q_p = arg\,max_q\, d(q, G_n^j(p))$. Call (p, q_p) a match if $d(q_p, G_n^j(p)) < d_{thr}^j$.
2) Update d_{thr}^{j+1} based on the histogram of $d(q_p, G_n^j(p))$.
3) If (p, q_p) is not a match, set $w_p = 0$ temporarily in (9). Update G_n^{j+1} to minimize $D_n(G_n^{j+1})$ in (9) as in the usual ICP algorithm.
4) If $\|G_n^{j+1} - G_n^j\| < \epsilon$ or $j > j_0$ then stop with $G_n = G_n^{j+1}$; else set $j \leftarrow j + 1$ and go to Step 1).

2) Splitting based on Classification

For node n, after finding the transformation G_n, label each point $p \in x_{t-1} \cap C_n$ as either matched (if $d(q_p, G_n(p)) < d_{thr1}$), unmatched (if $d(q_p, G_n(p)) > d_{thr2}$), or indeter-minate (if $d_{thr1} < d(q_p, G_n(p)) < d_{thr2}$). Set $j = 0$. Let M^j and U^j be the sets of matched and unmatched points. Repeat until exiting:

1) If there are too few points in M^j or U^j, or if the points in U^j are distributed like noise around M^j, then exit with the split a_n if one has been found, else exit with no split.
2) Compute the Fisher linear discriminant a_n to separate M^j from U^j. Among the points that a_n classifies as "matched" let M^{j+1} and U^{j+1} be the sets of matched and unmatched points.
3) Set $j \leftarrow j + 1$ and go to Step 1).

D. Proposed Algorithm: Integration

The estimate $\hat{x}_t$ is obtained by integrating y_t into $W_t(\hat{x}_{t-1})$. This is important because the integration 1) brings information on parts of the object or scene not previously seen, 2) helps average out the sensor noise, and 3) corrects drift.

Since we use point clouds to represent both y_t and $W_t(\hat{x}_{t-1})$, integration is simple: take the union. However, this would allow the size of the representation to grow indefinitely. Hence we limit the number of points in $\hat{x}_t$ by allowing at most one point per voxel. If two or more points

end up in the same voxel, then we take their weighted average (in both location and color), giving unit weight to points in y_t and weights θ to points in $W_t(\hat{x}_{t-1})$, where θ is a forgetting factor. θ can be applied uniformly to points in $W_t(\hat{x}_{t-1})$ whether averaged with other data or not, and points that fall below a weight threshold can be removed to clean up ancient history.

IV. Experiments and Results

In the experiments, we set up a single Kinect camera capturing both the color video and depth from a fixed viewpoint. Two sequences of human motion, *BigMotSit* and *RotateBody*, were captured and five select color frames are shown in Fig. 4. The frame rate of the captured sequences is 30 fps while the resolution is 640×480. MeshLab [20] is used to display the colored point clouds.

Fig. 5 and Fig. 6 compares cropped point clouds of the captured data y_t (in the first row) and estimated states x_t (in the second row) for selected frames of each sequence, from various virtual viewpoints, which are significantly different from the capturing viewpoints. In the first row, large areas of dis-occlusion can be observed, both as white areas on the background and as transparent areas on the body, through which the background (white or not) can be seen. In the second row, many of the transparent areas on the body have been filled in, using historical information warped to the current frame. (The white areas in the background could also have been filled in by our technique had the background been included in the volume of interest, and had the areas been observed by the camera in some previous frame.)

Fig. 7 compares the proposed algorithm with VSRS (Virtual View Rendering Software) provided by MPEG [21], for two virtual viewpoints with sequence *BigMotSit*. VSRS generates a virtual view for each frame by transforming the captured color plus depth data y_t to the virtual view and filling any detected dis-occlusions using inpainting. In contrast, the proposed algorithm transforms the integrated historical information x_t to the virtual view, filling dis-occlusions naturally. VSRS either fails to detect the dis-occlusions or fills them with objectionable artifacts, since the areas are so large.

From the experimental results, it can be observed that the proposed tracking algorithm is able to accurately track the non-rigid motion of objects. With the integration of temporal information, the proposed method successfully recovers the dis-occluded areas not captured at the current time and thus improves the 3D video quality.

Furthermore, the tracking scheme can also be applied to 3D video compression, where a more accurate prediction of current frame will be generated based on the hierarchical warping model and historical data.

V. Conclusion

Filling dis-occlusions is one of the most important problems in virtual view generation of 3D video. Since dis-occluded areas might be captured in previous frames, in this paper, we proposed a novel temporal information integration scheme to use historical frames to handle dis-occlusions. We formulate the problem as one of estimating the underlying states of a dynamic stochastic process given a sequence of observations. Based on this formulation, a hierarchical warping model is proposed to describe the movement between states, and states are estimated by integrating information from the previous state, the warping, and the current observation. Experimental results show that the proposed method can accurately track non-rigid motion and can successfully handle dis-occlusions captured in previous frames but not captured at the current time. Further improvements include dealing with video-depth de-synchronization during the matching and merging process as well as developing a more accurate matching algorithm.

References

[1] A. Smolic, K. Müller, et al., "3D Video and Free Viewpoint Video – Technologies, Applications, and MPEG Standards", *IEEE ICME*, Jul. 2006.

[2] H.-Y. Shum, S.-C. Chan, and S. B. Kang, Image-Based Rendering, *Springer*, 2006.

[3] C. Fehn, "Depth-Image-Based-Rendering (DIBR), Compression and Transmission for a New Approach on 3D-TV", *SPIE Stereoscopic Displays and Virtual Reality Systems*, Jan. 2004.

[4] C. L. Zitnick, S. B. Kang, et al., "High-Quality Video View Interpolation Using a Layered Representation", *ACM SIGGRAPH*, Aug. 2004.

[5] Z. Tauber, Z.-N. Li, and M. S. Drew, "Review and Preview: Disocclusion by Inpainting for Image-Based Rendering," *IEEE Trans. SMC*, Jul. 2007.

[6] S. Moezzi, L.-C. Tai, and P. Gerard, "Virtual View Generation for 3D Digital Video", *IEEE Multimedia*, Vol. 4, Issue 1, pp. 18-26, 1997.

[7] B. Petit, J-D. Lesage, et al. "Multicamera Real-Time 3D Modeling for Telepresence and Remote Collaboration", *Int'l J. of Digital Multimedia Broadcasting*, 2010.

[8] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: a Hybrid Geometry- and Image-Based Approach", *ACM SIGGRAPH*, Aug. 1996.

[9] O. van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, "A Survey on Shape Correspondence," *Proc. Eurographics State-of-the-art Report,* pp.1-24, 2010.

[10] P.J. Besl and H. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. PAMI*, Vol. 14, No. 2, 1992.

[11] Z. Zhang, "Iterative Point Matching for Registration of Free-Form Curves and Surfaces", *Int'l J. Computer Vision*, Vol. 3, Issue 2, pp. 119-152, 1994.

[12] J. Feldmar and N. Ayache, "Locally Affine Registration of Free-Form Surfaces", *CVPR*, Jun. 1994.

[13] H. Li, B. Adams, L.J. Guibas, and M. Pauly, "Robust Single-View Geometry and Motion Reconstruction," *ACM Trans. Graphics*, Vol 28, No. 5, 2009.

[14] W. Chang and M. Zwicker, "Global Registration of Dynamic Range Scans for Articulated Model Reconstruction," *ACM Trans. Graphics*, Vol. 30, No. 3, 2011.

[15] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, "Plenoptic Sampling," *ACM SIGGRAPH*, Jul. 2000.

[16] K. Müller, P. Merkle, and T. Wiegand, "3-D Video Representation Using Depth Maps," *Proc. IEEE*, Apr. 2011.

[17] A. Smolic, K. Müller, P. Merkele, P. Kauff, and T. Wiegand, "An overview of available and emerging 3D video formats and depth enhanced stereo as efficient generic solution", *Picture Coding Symp.,* May 2009.

[18] Q. Cai, D. Gallup, C. Zhang, and Z. Zhang, "3D Deformable Face Tracking with Commodity Depth Cameras," *ECCV*, Sep. 2010.

[19] Bishop, Pattern Recognition and Machine Learning, *Springer*, Ch. 13, 2006.

[20] *MeshLab*, http://meshlab.sourceforge.net/.

[21] M. Tanimoto, T. Fujii , and K. Suzuki, "View synthesis algorithm in view synthesis reference software 3.0 (VSRS3.0)," *Tech. Rep. Document M16090, ISO/IEC JTC1/SC29/WG11*, Feb. 2009.

Figure 4. Selected frames of captured data from capture viewpoint (from left to right: *BitMotSit* frame 0, 4, 8; 2nd row: *RotateBody* frame 0, 20, 40)

Figure 5. Comparison of point clouds from virtual viewpoint: (1st row: captured data (y_t), 2nd row: captured data merged with warped historical data (x_t); Frame numbers from left to right: 4, 14, 24, 34, 44, 54)

Figure 6. Comparison of point clouds from virtual viewpoint: (1st row:y_t, 2nd row: x_t; Frame number from left to right: 2, 22, 42, 62, 82, 102)

(a) (b) (c) (d) (e) (f)

Figure 7. Comparison of rendering results of frame 14 in *BitMotSit* based on VSRS (a,c,e) and proposed method (b,d,f)