·**Article**·

View Synthesis from multi-view RGB data using multi-layered representation and volumetric estimation

Zhaoqi SU[1], Tiansong ZHOU[2], Kun LI[2], David BRADY[3], Yebin LIU[1*]

1. *Department of Automation, Tsinghua University, Beijing* 100086, *China*

2. *Tianjin University, Tianjin* 300072, *China*

3. *Duke Kunshan University, Kunshan* 215316, *China*

* **Corresponding author,** liuyebin@mail.tsinghua.edu.cn

Received: 10 July 2019 **Revised:** 15 December 2019 **Accepted:** 2 December 2019

Citation: Zhaoqi SU, Tiansong ZHOU, Kun LI, David BRADY, Yebin LIU. View Synthesis from multi-view RGB data using multi-layered representation and volumetric estimation. Virtual Reality & Intelligent Hardware, 2020, 2(1): 43—55

DOI: 10.1016/j.vrih.2019.12.001

Abstract **Background** Aiming at free-view exploration of complicated scenes, this paper presents a method for interpolating views among multi RGB cameras. **Methods** In this study, we combine the idea of cost volume, which represent 3D information, and 2D semantic segmentation of the scene, to accomplish view synthesis of complicated scenes. We use the idea of cost volume to estimate the depth and confidence map of the scene, and use a multi-layer representation and resolution of the data to optimize the view synthesis of the main object. **Results/Conclusions** By applying different treatment methods on different layers of the volume, we can handle complicated scenes containing multiple persons and plentiful occlusions. We also propose the view-interpolation→multi-view reconstruction→view interpolation pipeline to iteratively optimize the result. We test our method on varying data of multi-view scenes and generate decent results.

Keywords View interpolation; Cost volume; Multi-layer processing; Multi-view reconstruction; Iterative optimization

1 Introduction

With the rapid development of AR/VR devices, there is now a demand for immersive experience in digital settings, such as boxing games or virtual tours, that mimic real-world scenes. Most AR/VR materials for such scenes are produced via the 3D reconstruction and texturing of the whole scene. However, with the massive data representing these 3D models and the unavoidable holes on the borders of the models, such a method is impractical for representing dynamic scenes. On the other hand, some studies have focused on directly interpolating views from among known views[1,2]. The methods proposed in these studies can interpolate novel views among cameras and directly generate images of certain views. However, these methods either cannot generate subtle background scenes[1] or might not perform well in complicated scenes with massive occlusions and multiple persons[2].

Furthermore, with the development of techniques for image segmentation, such as in [3] and [4], the input image can be segmented into different layers or labels according to semantic information. The main objects (which are, most of the time, the main persons) in the pictures can be accurately segmented as the foreground object. Thus, we perform segmentation via these methods to generate a multi-layer representation of the data and apply different treatment methods on different layers so that the foreground and background regions can be separately handled by the algorithm. We first segment the input RGB multi-view data using the method in [5] and use the input image as the RGB information to optimize the segmentation. Next, we define different resolution volume spaces for different layers, and estimate the depth and the confidence map using the idea of cost volume, as in [2], with the help of prior segmentation. Four nearby views are then used to generate a novel view image, using the confidence map estimated in the previous step. We also propose the view-interpolation→multi-view reconstruction→view interpolation pipeline to iteratively optimize our view interpolation result and to generate better 3D models, compared to models created via 3D reconstruction, using only the input RGB data.

The main contributions of our work are as follows:

(1) We combine the idea of cost volumes from [2] and semantic segmentation to deal with the view interpolation of complicated scenes with much occlusion and many objects, a procedure that has not been done before.

(2) We propose the view-interpolation→multi-view reconstruction→view interpolation pipeline to iteratively optimize both the 3D reconstruction of the scene and the view synthesis results.

The rest of the paper is divided as follows: Section 2 introduces related work on 3D reconstruction and view synthesis, Section 3 describes our detailed method for view synthesis using the view-interpolation→ multi-view reconstruction→view interpolation pipeline, and Section 4 reveals and discusses the experiment results of our method.

2 Related work

This section presents related research on view synthesis and stereo depth estimation and reconstruction. Previous studies on view synthesis are divided mainly into two methods: the first is 3D reconstruction and rendering, and the second is the direct use of input views to generate novel views.

The first method leverages 3D reconstruction and texture mapping to generate new views[6,7,8]. For example, Liu et al. used an optical-flow-based method to optimize the initial visual-hull model, captured and calculated by an indoor multi-camera capture system, and renders it with texture[8]. This method is designed for generating free-view viewpoints of a particular object. The method in [6] tends to generate more general scenes, like buildings and landscapes, by using several photos, taken by consumer-level cameras, to reconstruct a model of the scene. Guo et al. used a fusion-based method to generate 3D models of human motion with albedo and texture information[7]. With this system, when the human body is completely captured, a free-view scene of the person can be created. Mustafa et al. used a segment-based method to generate 4D dynamic models of indoor scenes[9]. This method has the potential for creating scenes of random views with texture information, but the use of texture is not implemented in the study. These aforementioned methods can be used to generate free views of a scene, but often with holes on the background area and with distortion artifacts of the 3D model[5], because some background areas of the input pictures have relatively low correspondence, or can generate free views only of particular objects or humans[7,8].

The second method is to render the novel view directly using the input pictures. Some studies on view interpolation used traditional ways of estimating novel views, while others use deep-learning-based methods. Among the studies on traditional methods, Chen et al. is the first paper proposing image-based view synthesis, which integrates image morphing techniques into an interactive view-interpolation pipeline[10]. Zitnick et al. used a layered representation of a scene to form continuous view changes inside the view angle of eight cameras[11]. Ballan et al. used a plane approximation of the foreground object and a 3D model of the background to handle the view changes of a human in a large scenario[1]. Penner et al. used the idea of confidence volume to estimate the depth, and then the confidence map of a 3D scene, a method that can generate excellent results with some types of scenes[2]. Li et al. proposed a new view-synthesis framework and makes good use of other neighboring complementary views to implement hole filling in view-synthesis scenarios[12]. These methods tend to either have good results with relatively simple scenes with few occlusions[2,11], compared with complicated scenes, such as live boxing games, or generate blurred backgrounds[1]. Our method is based on [2], and with semantic segmentation and 3D-reconstruction-based iterative optimization, we can handle the view interpolation of more complicated scenes.

Deep-learning-based methods are used mostly for the view interpolation of static scenes. For example, Kalantari et al. used a CNN method to estimate a disparity map for 2×2 light field cameras, which is then used to warp a source image into novel views[13]. Flaynn et al. used an end-to-end network to predict pixels in novel views; this method utilizes nearby views to estimate the color tower and depth tower, and then uses this information to synthesize novel views[14]. Zhou et al. used a layered representation to generate multi-plane images (MPI); this representation contains both the RGB and alpha information of the scene, and the method uses the MPI layers to blend the novel views[15]. Hedman et al. used different warped views according to estimated depth as view mosaics and trained a network to learn the blending weights of the neighbor views[16]. Mildenhall et al. trained MPIs for each input view and used existing views as ground truth for training, and then used nearby MPIs to warp the novel views[17]. Srinivasan et al. also estimated the MPI representation of a scene by leveraging flow information, to perform view extrapolation. Because deep-learning-based methods rely mostly on the diversity of the dataset, and because there are few multi-view datasets suitable for complex dynamic scenes, these methods focus mainly on static scenes[18]. Our method uses semantic information and traditional 3D reconstruction methods and can therefore tackle the problem of view synthesis for dynamic scenes.

3 Overview

Because the method in [2] performs well in the view interpolation of several scenes, we use their pipeline as the backbone and apply some adjustments for our data. However, the method in [2] has some drawbacks: it considers only the color and edge information, although with consistency estimation of neighbor views, and is still difficult to be used in complicated scenes where the objects may have various occlusions and where the baseline is wide and random between views. Therefore, we propose our method, which is based on [2] but with modification of the algorithm and uses semantic segmentation to recognize particular objects as our specified layer, and use the view-interpolation→multi-view reconstruction→view interpolation pipeline for special cases to leverage global 3D information.

Our algorithm basically includes three stages. Stage 1 is data segmentation and optimization, described in Section 3.1; Stage 2 is depth and confidence volume estimation, described in Section 3.2; Stage 3 is color blending and view interpolation, described in Section 3.3; Stage 4 is optional, which is the multi-view reconstruction→view interpolation iteration for the optimization of results, described in Section 3.4.

3.1 Segmentation

The multi-view RGB data are the input of our algorithm. In the first step, we segment the RGB pictures into different layers, thus forming a multi-layer representation of the scene. The reason for performing the segmentation are as follows:

(1) When the scene is complicated with abundant occlusion, especially when the focus objects of the scene (such as the players of a boxing game) are quite small in the picture, it may be difficult for the traditional view-interpolation method to handle the data. Therefore, with segmentation, the relationship between different layers, such as the foreground and background objects, can be clearer.

(2) Segmentation of the scene can also give us access to using different volume resolutions in different layers, which will make the depth estimation of the foreground object more subtle and optimize the quality of the synthesized foreground objects.

Our segmentation method is based on Mask-RCNN[3], wherein each image from every view in every frame is fed to a pre-trained network, as in [5], resulting in the segmentation of layers. For each object or person we want to set as the h^{th} layer $\mathcal{L}_h$, we specify the corresponding segment for one view in one frame, for example, the m^{th} segment of the 0^{th} view in the 0^{th} frame, denoted as $\mathcal{S}_{0,0}^m$, and then use rough depth estimation (which will be described in Section 3.2) to automatically determine the corresponding segment of the same objects in other views and in other frames.

To be more specific, if the m^{th} segment is determined in view i in the j^{th} frame as $\mathcal{S}_{i,j}^m$, and the segmented parts generated by the method from [5] for view $i+1$ as $\mathcal{S}_{i+1,j}^{m'}, m' \in segments$, we want to determine m' for the object. First, we estimate the rough depth as described in Section 3.2, and then for every point (x,y) inside $\mathcal{S}_{i,j}^m$, we use the i^{th} camera parameters to map it into the 3D space, and project it into view $i+1$:

$$(x',y') = C_{i+1}(C_i^{-1}(x,y)), \tag{1}$$

where C_i and C_{i+1} are the camera matrices for the two views.

After every point inside $\mathcal{S}_{i,j}^m$ is projected into view $i+1$, we assign the segments which contain the most projected points as the correct segments for layer $\mathcal{L}_h$. The process is shown in Figure 1.

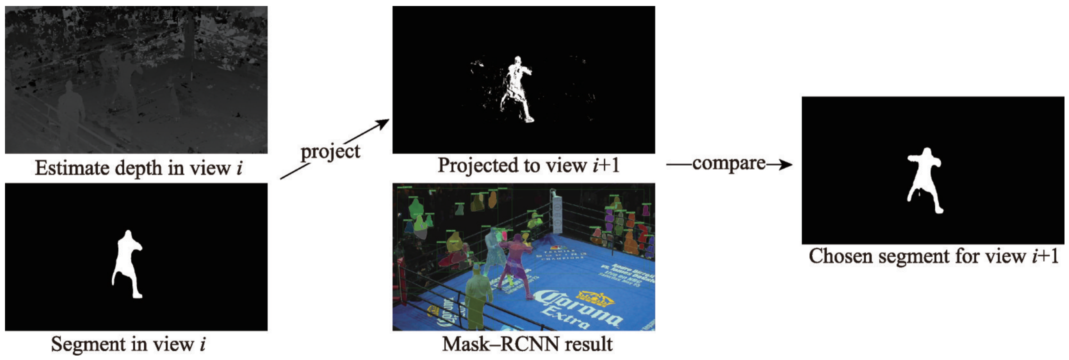

Figure 1 Procedure for choosing the segment for a particular layer.

After each object to be selected as foreground is automatically determined, basic segmentation is made for every view in every frame. Because the segments from Mask-RCNN [3] sometimes fail to capture the limbs of the person when the person in the picture is occluded by another person, as shown in Figure 2, we multi-view information to iteratively optimize the segmenting of all views altogether. For view i, we choose four nearby views and view i itself to estimate the segment-confidence map $Seg\text{–}conf_i$:

$$Seg\text{–}conf_i(x,y) = Seg_i(x,y) + \epsilon(\Delta d) \sum_{j \in \mathcal{N}_i} Seg_j(C_j(C_i^{-1}(x,y))), \tag{2}$$

where $\mathcal{N}_i$ denotes the neighboring views of view i, $Seg_i(x,y)$ denotes as a 1 or 0 value whether (x,y) belongs to the foreground region, and Δd is the absolute difference between the depth estimated at the projected pixel and the depth calculated in view j with $C_i^{-1}(x,y)$. $\epsilon(x)$ is a gaussian function which gives

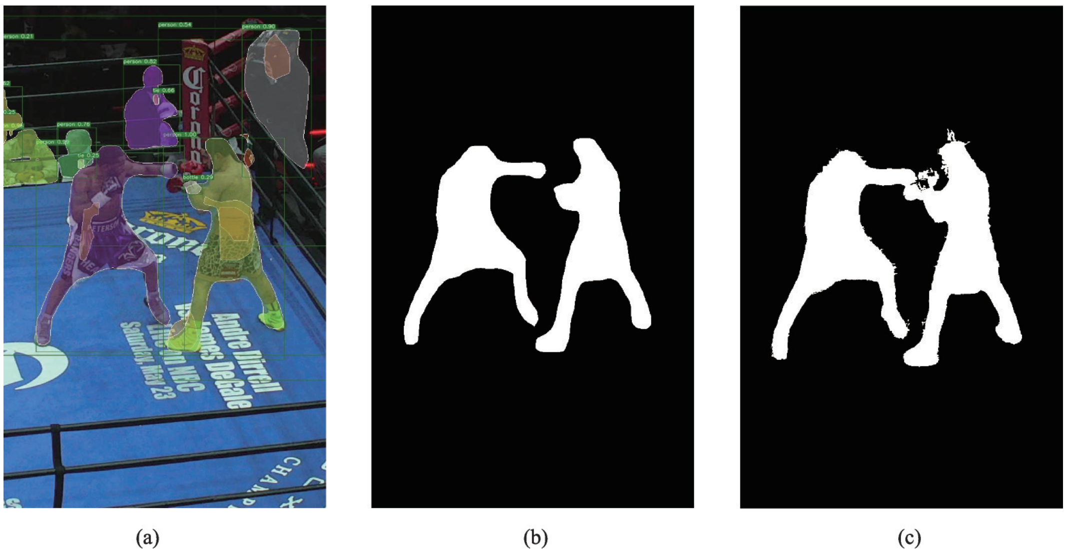

Figure 2 Illustration of the role of using multi-view information to optimize the segmentation. (a)(b) Segmentation using Mask-RCNN[3]; (c) Segmentation after optimization.

more weight with more consistent depth estimation.

After the segment-confidence map $Seg\text{-}conf_i$ is estimated, the segment of view i is determined by this map: when $Seg\text{-}conf_i(x,y)$ is above a threshold ϵ_{sil}, the point is regarded as a foreground point; otherwise, it is classified as a background point.

3.2 Volumetric estimation

The volumetric estimation basically includes two parts: depth estimation and confidence volume estimation. The basic idea follows [2], but with multi-layer representation of the scene: the estimation optimizes the multi-layer information and generates more robust results for complicated scenes.

Like in [2], for each view in every frame, we make plane sweeps and generate a discrete ray for every pixel, thus formulating a volume $V(x,y,z)$, where (x,y) represents the pixel, and z denotes the depth. We estimate the depth estimation volume of every view as in [2]:

$$E_{raw}(x,y,z) = \frac{\sum_{k \in N} E_k(x,y,z)\mathcal{L}_k(x,y,z)}{\sum_{k \in N} \mathcal{L}_k(x,y,z)}$$
$$E(x,y,z) = \sum_{(\hat{x},\hat{y}) \in W} w(x,y,\hat{x},\hat{y})E_{raw}(\hat{x},\hat{y},z), \tag{3}$$

where N represents the nearby views of the current view, $E_k(x,y,z)$ denotes the absolute color difference between the pixel and the projected point in view k with coordinate (x,y,z). $\mathcal{L}_k(x,y,z)$ denotes whether (x,y) in the current view belongs to the same layer with the projected point. W is the perception domain of the guided filter at pixel (x,y). $w(x,y,\hat{x},\hat{y})$ is the guided filter kernel[19], which also returns 0 when (x,y) and $(\hat{x},\hat{y})$ are labeled in different layers.

With the depth estimation volume calculated, the raw depth image of each view is decided as follows:

$$D_{raw}(x,y) = \arg\min_z E(x,y,z). \tag{4}$$

Because the depth estimated using this method may be limited to pixel-level calculation and by a lack of global information, we use the multi-view optical-flow-based method in [8], which proposed the multiple starting scales (MSS) framework and utilized the coarse-to-fine image pyramid to catch global patch information. This method can thus be used to optimize the quality of the depth image:

$$\begin{aligned} Flow^d &= DepthToFlow(D_{raw}) \\ Flow^i &= Upsample\&Optimize(Flow^{i+1}), \\ D &= FlowToDepth(Flow^0) \end{aligned} \tag{5}$$

where $Flow^i(i = d - 1, d - 2, \cdots, 0)$ denotes the i^{th} layer pyramid optical flow. In each pyramid layer, the flow will be optimized by minimizing the optical-flow based energy function, as in [8]:

$$\begin{aligned} E(w) &= E_D(w) + \alpha E_S(w) \\ E_D(w) &= \int_\Omega \psi_D(|I_r(p + d(w)) - I_t(p)|^2 \\ &\quad + \gamma|\Delta I_r(p + d(w)) - \Delta I_t(p)|^2)dxdy, \\ E_S(w) &= \int_\Omega \psi_S(|\Delta w|^2)dxdy \end{aligned} \tag{6}$$

where $E_D(w)$ and $E_S(w)$ denote the data term and smooth term, respectively, of the energy function, w represents the flow, and $d(w)$ denotes the flow projected to the epipolar line of the target image. ψ_D and ψ_S are the robust functions used in [20]. The final layer of the pyramid is then transferred to the depth image D.

After the depth image D_i is calculated for each view i, as adjusted from [2], the surface consensus volume is estimated as follows:

$$\begin{aligned} Con_{raw} &= \frac{\sum_{k \in N} w_k e^{|D_k(x',y') - z'|}}{\sum_{k \in N} w_k}, \\ Con(x,y,z) &= \sum_{(\hat{x},\hat{y}) \in W} w(x,y,\hat{x},\hat{y})Con_{raw}(\hat{x},\hat{y},z) \end{aligned} \tag{7}$$

where N and W are defined in the same way as in equation (3). (x',y',z') denotes the point (x,y,z) transformed from view i to view j using camera parameters, and

$$w_k = (D_k(x',y') - z' < \epsilon_{depth} \&\& \mathcal{L}_i(x,y) = \mathcal{L}_k(x',y')) \tag{8}$$

indicates whether the point (x',y',z') can be seen in view k, and whether the points have the same layer label with the corresponding point in view i. The Con volume is also calculated using the layer-labelled guided filter, as in equation (3).

To increase robustness, the above procedures are iteratively operated, because the results for the surface consensus volume can be used to optimize the depth estimation, because it increases the information for the consistence of different views. Therefore, we recalculate the depth estimation volume as follows:

$$\begin{aligned} E^{iter}_{raw}(x,y,z) &= \frac{\sum_{k \in N} E_k(x,y,z)Con(x',y',z')\mathcal{L}_k(x,y,z)}{\sum_{k \in N} Con(x',y',z')\mathcal{L}_k(x,y,z)} \\ E(x,y,z) &= \sum_{(\hat{x},\hat{y}) \in W} w(x,y,\hat{x},\hat{y})E_{raw}(\hat{x},\hat{y},z) \end{aligned} \tag{9}$$

The rest of the steps will still be operated to iteratively increase the quality of the depth image and the robustness of the surface consensus volume.

3.3 View interpolation

For a novel view which we want to synthesize from the scene, we use the nearby 4 views to interpolate the final view. To consider both the visibility and the confidence volume, we generate the scene representation volume for view i as follows:

$$scene_i(x,y,z) = \max(0, \min(Con_i(x,y,z), 1 - \sum_{\hat{z} < z} Con_i(x,y,\hat{z}))). \tag{10}$$

The scene representation and the consensus volume of each of the nearby views can then be separately used to generate the color volume and the consensus volume of the novel view N with layer k:

$$\begin{aligned} color_{N,k}(x,y,z) &= \frac{\sum\limits_{\substack{i \in \mathcal{N}_N \\ \mathcal{L}_i(x_{i'},y_{i'}) = k}} scene_i(x_{i'},y_{i'},z_{i'}) I_i(x_{i'},y_{i'})}{\sum\limits_{\substack{i \in \mathcal{N}_N \\ \mathcal{L}_i(x_{i'},y_{i'}) = k}} scene_i(x_{i'},y_{i'},z_{i'})}, \\ Con_{N,k}(x,y,z) &= \sum_{\substack{i \in \mathcal{N}_N \\ \mathcal{L}_i(x_{i'},y_{i'}) = k}} Con_i(x_{i'},y_{i'},z_{i'}) \end{aligned} \tag{11}$$

where $\mathcal{N}_N$ represents the nearby views of view N. $\mathcal{L}_i(x_{i'},y_{i'}) = k$ denotes the pixel $(x_{i'},y_{i'})$ in the i^{th} image that is labeled with semantic layer k. Each pixel (x,y) in the novel view labeled as each layer $\mathcal{L}_k$ will give values for the color and the confidence, which utilize the segmentation information as in Section 3.1. The advantage is that through this procedure, the color blending will have less tendency to blend the colors of the different objects in the scene, which benefits the view interpolation of a complicated scene.

Then, we calculate the label volume and the final consensus volume as follows:

$$\begin{aligned} Con_N(x,y,z) &= \max Con_{N,k}(x,y,z) \\ \mathcal{LV}_N(x,y,z) &= \arg\max_k Con_{N,k}(x,y,z) \end{aligned} \tag{12}$$

To synthesize the novel view, we search for z for each pixel (x,y) to get the best-matched depth. To consider the occlusion, the search range of z is given as follows:

$$\mathcal{R}_N(x,y) = \{ z | \sum_{\hat{z} < z} Con_N(x,y,z) < \epsilon_{novel} \} \tag{13}$$

We first mark the layer label for each pixel (x,y) using the search range calculated above:

$$\mathcal{L}_N(x,y) = \mathcal{LV}_N(x,y,\arg\max_{z \in \mathcal{R}_N(x,y)} Con_N(x,y,z)) \tag{14}$$

Because there might be salt-and-pepper noise in the layer label, we propagate the label value to a pixel if most of its neighbor labels are consistent. After that, we can finally calculate the RGB value of that pixel:

$$\begin{aligned} RGB_N(x,y) &= color_{N,\mathcal{L}_N(x,y)}(x,y,z_0) \\ z_0 &= \arg\max_{z \in \mathcal{R}_N(x,y)} Con_{N,\mathcal{L}_N(x,y)}(x,y,z)) \end{aligned} \tag{15}$$

3.4 Optimizing from reconstruction

To optimize the robustness of the algorithm, we propose the view-interpolation→multi-view reconstruction →view interpolation pipeline to iteratively optimize the result, especially when a major area of the scene is still (i.e., not moving). We first generate the view interpolation result of the scene, and then take the output images and use the multi-view reconstruction method MVE [6] to generate a 3D model, which can then be used to optimize the depth estimation in our pipeline. As shown in Figure 3, after performing optimization from the multi-view reconstruction, we obtain a better estimated depth of the scene with fewer outliers, which benefits our final view-synthesis results.

In practice, because such a 3D reconstruction method is very time-consuming, we tend to generate only the static background utilizing the segmentation we did before. Then, we use the generated 3D model to project to each camera and get the background depth image. We then use this depth image to renew the depth estimated in the first step of Section 3.2 in every frame. The reason for using the 3D reconstruction to optimize the results is that the 3D reconstruction tends to eliminate the outliers of the 3D scene, which can give a more robust depth for the initialization of our pipeline. Furthermore, the 3D reconstruction method takes advantage of using global information instead of local information, a characteristic that

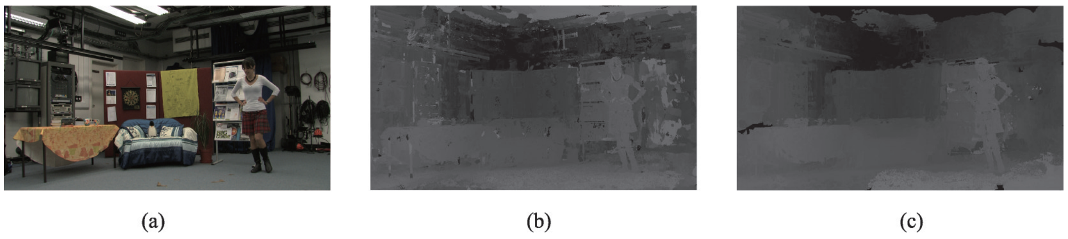

Figure 3 Depth estimation before and after optimization from reconstruction. (a) Input image; (b) Depth estimation without optimization; (c) Depth estimation with optimization.

forms complementarity with our current pipeline.

One more thing to be pointed out in this pipeline is, we do not use the origin input image as the 3D reconstruction input. The reason is that because of the wide baseline of the images, the method from [6] tends to generate a scene with large holes. To illustrate that, we use one of the datasets from [9], which contains 6 still camera frames, to generate 40 output views, and we respectively take 6 input images and 40 output images as the MVE input; the output 3D model is shown in Figure 4. The MVE model generated from the output 40 views apparently has a better quality than the one generated from only the input 6 views.

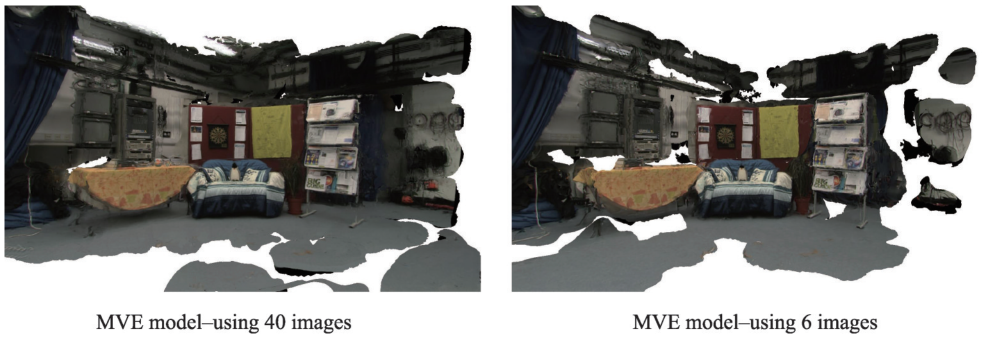

Figure 4 Comparison of models created using different MVE inputs.

After the model is generated and projected to each input view, the depth image is acquired and can be used to iteratively optimize the results by feeding it into the first step.

4 Results

In this section, we show some of the results for the generated novel view of the multi-view input data. Because our work focuses mainly on dealing with complicated scenes, which have hardly been used for view interpolation before, there may not be much information from previous studies with which to compare our results. The following experiments are conducted on three datasets: the boxing dataset, which contains a scene with 500 frames with 32 views; the dataset from [8], which contains a 20-view scene inside a green-background "cage", and the dataset from [9], which has just 6 views of a girl inside a room.

Figure 5 shows the multi-view data of a boxing game and its corresponding view-generating result. The first and the third columns are the nearby input views of the multi-view data, and the second column is the generated scene. The scene is complicated because there are occlusions between the players in the boxing ring, and also between the boxing ring and the audiences and the referee. Furthermore, there are so many people and objects in the input pictures. Figure 5 shows that we can handle this kind of scene and generate good results.

We also generate views from the input data from [8], which contains multi-view data of a single person inside a "cage" surrounded by a green screen. With this setup, the human can be easily segmented, and we

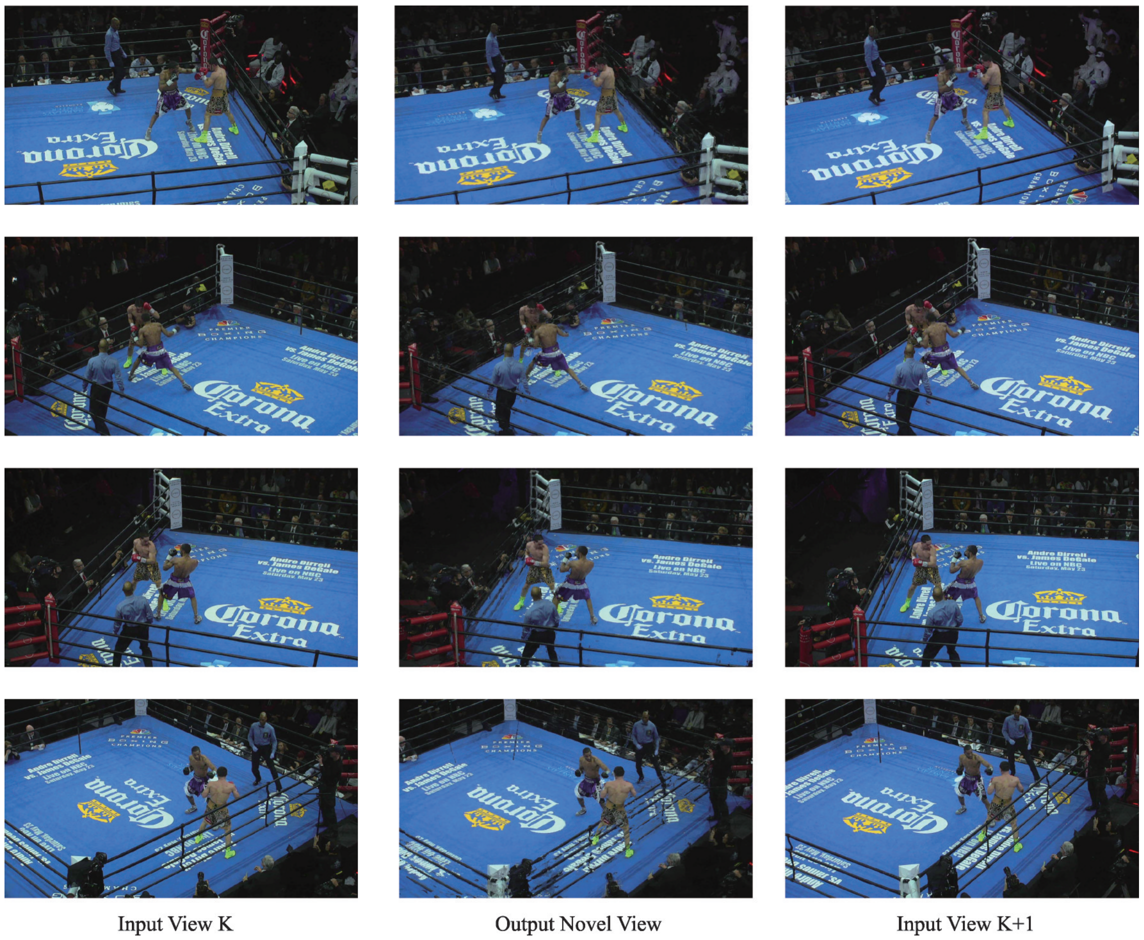

Figure 5 Generated novel view of boxing data.

can generate a view of only the human. As shown in Figure 6, we can generate a high-quality free view of a human.

The data from [9] is also used for testing our view interpolation result. In this dataset, the scene, except for the human, is static, which is well-suited for our view-interpolation→multi-view reconstruction→view interpolation pipeline. We first use the method from [5] to segment the human in the scene, and then we use only the background part to generate 40 views between the input 6 views using our method described in Section 3. Afterward, we use the 40 output views as input to generate a 3D model using [6]. The generated model is shown in Figure 4. The model is then projected to each input view to generate more robust depth estimation of the background part, which we use to replace the depth estimated in the first step. The result is shown in Figure 7.

The comparison between using this pipeline and not using image segmentation or reconstruction for optimizing this dataset is shown in Figure 8. As is shown, without image segmentation, because of the inaccurate depth estimation of the foreground object, the result shows artifacts in the boundary area of the human. Furthermore, without optimization from reconstruction, there are some artifacts in the background area because of the complicated occlusion between objects and the subtle depth estimation result without such optimization.

We compare our method with that from [2]. As shown in Figure 9, the method from [2] generates blurring results, particularly in the boundary area of the human, whereas by contrast, our method generates clearer boundaries benefiting from our semantic segmentation strategy. In addition, the method from [2] cannot estimate accurate depth, especially in areas between two persons, but with our view-

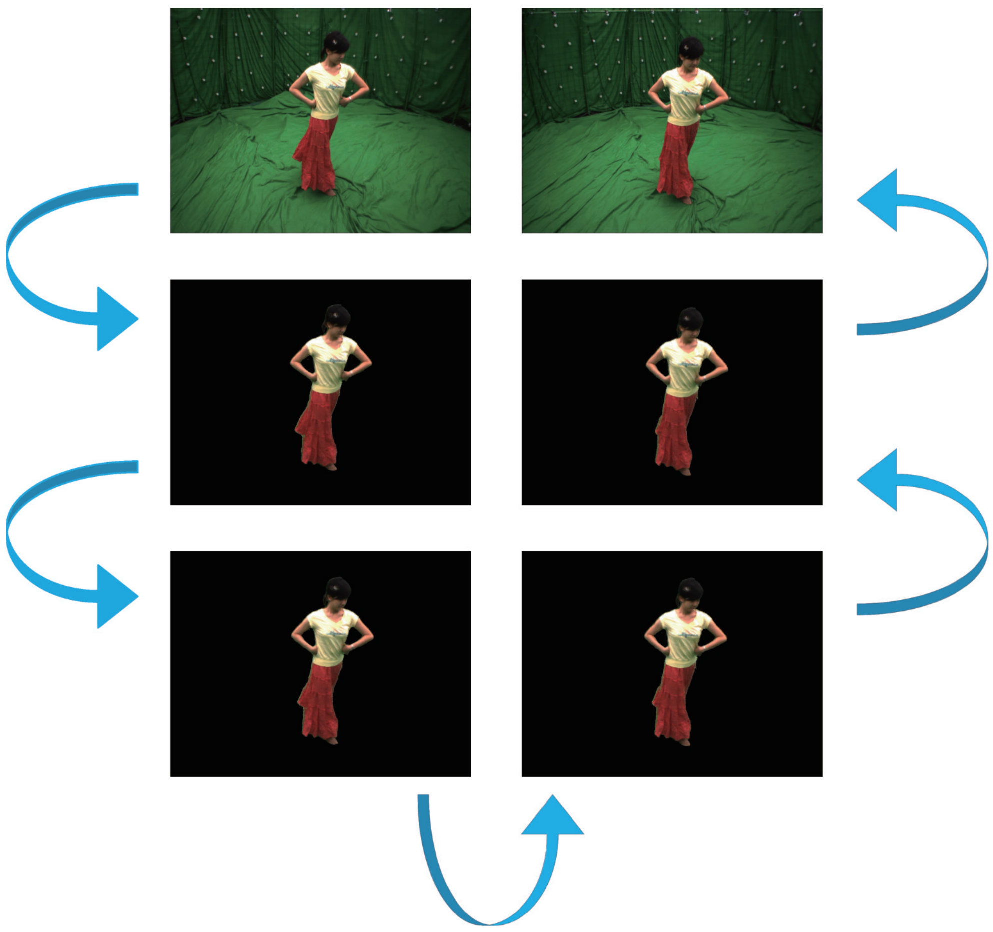

Figure 6 View interpolation results of data from [8]; the top row are the two nearby input views, and other four pictures are the generated novel view. Notice that the nearby cameras have a relatively small baseline compared to those of other datasets; the video demo should have a clearer demonstration of this result.

interpolation→multi-view reconstruction→view interpolation pipeline and semantic layer representation, we can avoid the ghosting effects of the floor area between the two boxers. We also compare our method with that from [17]. As shown in Figure 9, because of the complexity of our scene, the method from [17] cannot generate a clear view-interpolation result because it focuses mainly on a simple scenario with a few objects. On the other hand, our method can handle plausible view interpolation of this kind of the scene.

In short, we propose a view-interpolation method leveraging both volumetric information and semantic information of the input images and use the view-interpolation→multi-view reconstruction→view interpolation pipeline to add robustness to the view interpolation of dynamic scenes. Our work still has some limitations; for instance, scenes without clear semantic layers may not be used to generate plausible results, according to our pipeline. Furthermore, the quality of the result relies heavily on the quality of the segmentation method, and if some body parts are not accurately segmented (e.g., hands, legs), the result may show some blurring effect in these areas. Because our pipeline focuses mainly on dynamic scenes, and because there are few corresponding methods that are open-source, it is difficult to make comparisons with current studies on view synthesis.

For future research, a more robust depth estimation can be integrated into our pipeline, and some 3D deep-learning approach for volumetric estimation may also be explored. Furthermore, the continuousness of the interpolated scene between different frames can also be considered.

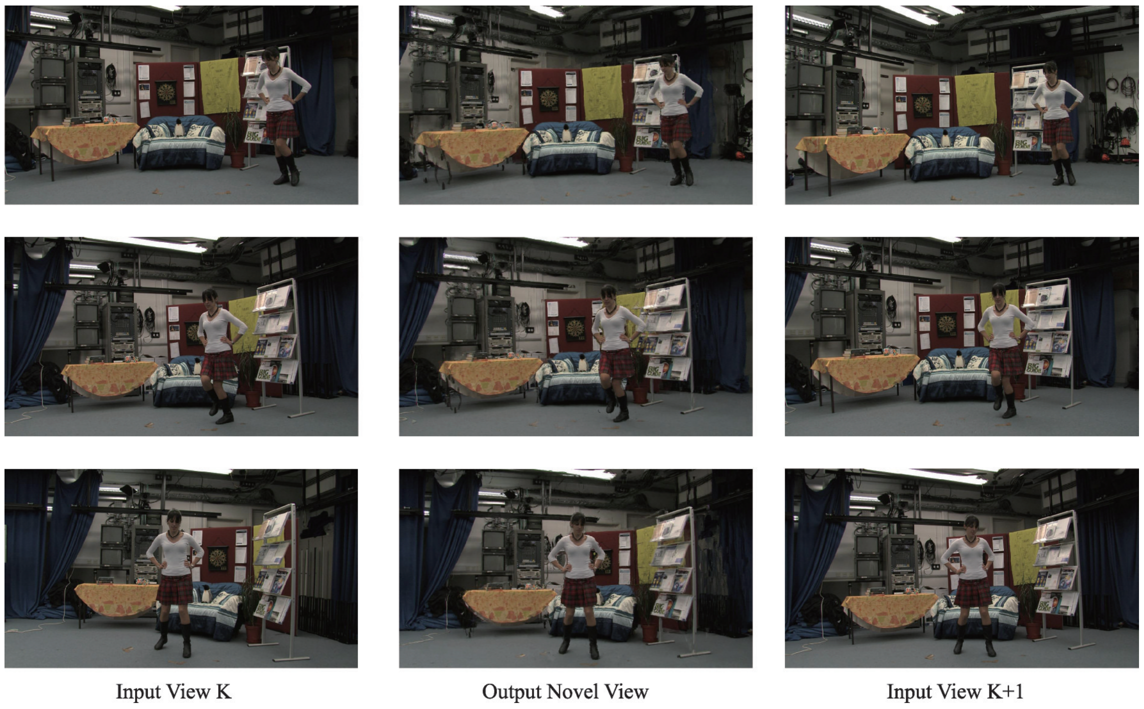

Figure 7 Generated novel view of dance data from [9].

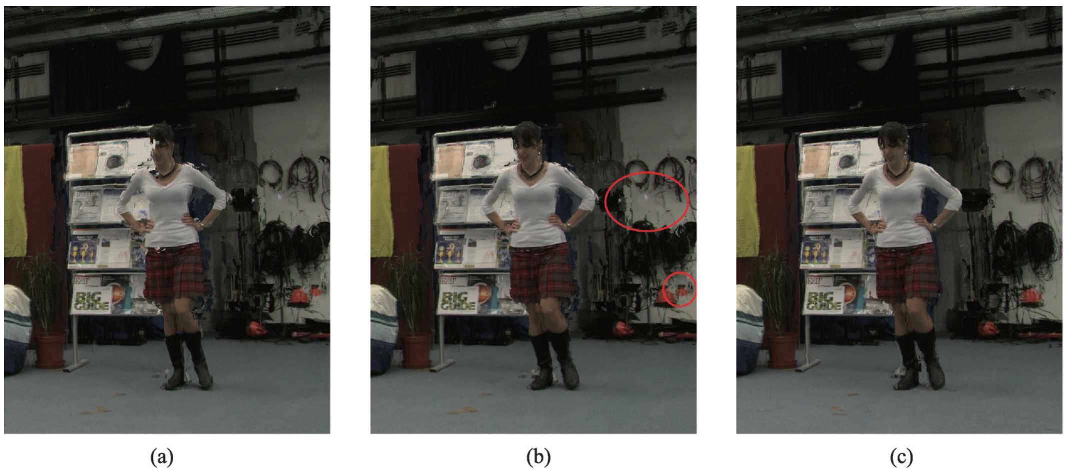

Figure 8 Illustration of the role of segmentation and view-interpolation→multi-view reconstruction→view interpolation pipeline. (a) Result without image segmentation procedure; (b) Result without optimization from reconstruction; (c) Result with optimization.

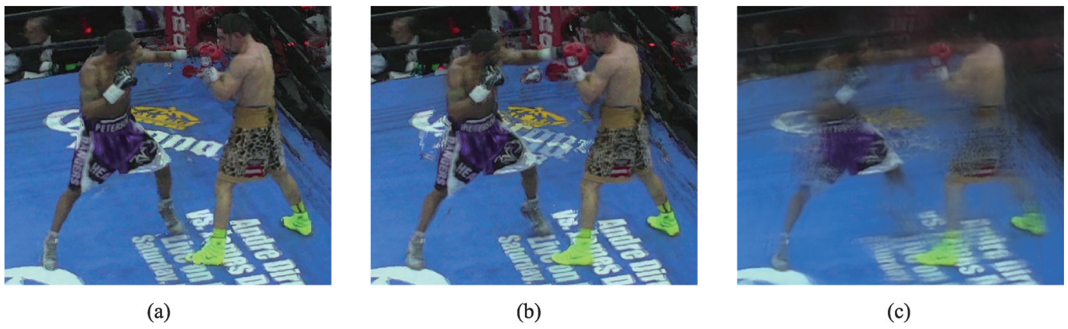

Figure 9 Comparison between our method, soft3d[2], and local light-field fusion[17]. (a) Result generated by our method; (b) Result generated by method from [2]; (c) Result generated by method from [17].

Acknowledgements: The authors would like to thank Tsinghua Univesity, Tianjin University, and Duke Kunshan University for supporting the research.

References

1 Ballan L, Brostow G J, Puwein J, Pollefeys M. Unstructured video-based rendering: interactive exploration of casually captured videos. In: ACM SIGGRAPH 2010 papers. Los Angeles, California, ACM, 2010, 1–11
DOI:10.1145/1833349.1778824

2 Penner E, Zhang L. Soft 3D reconstruction for view synthesis. ACM Transactions on Graphics, 2017, 36(6): 1–11
DOI:10.1145/3130800.3130855

3 He K M, Gkioxari G, Dollar P, Girshick R. Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV). Venice, IEEE, 2017
DOI:10.1109/iccv.2017.322

4 Wang S H, Sun J D, Phillips P, Zhao G H, Zhang Y D. Polarimetric synthetic aperture radar image segmentation by convolutional neural network using graphical processing units. Journal of Real-Time Image Processing, 2018, 15(3): 631–642
DOI:10.1007/s11554-017-0717-0

5 Girshick R, Radosavovic I, Gkioxari G, Dollár P, He K. Detectron. https://github.com/facebookresearch/detectron, 2018

6 Fuhrmann S, Langguth F, Goesele M. MVE: a multi-view reconstruction environment. In: Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage. Darmstadt, Germany, Eurographics Association, 2014, 11–18
DOI:10.2312/gch.20141299

7 Guo K W, Xu F, Yu T, Liu X Y, Dai Q H, Liu Y B. Real-time geometry, albedo and motion reconstruction using a single RGBD camera. ACM Transactions on Graphics, 2017, 36(4): 1
DOI:10.1145/3072959.3126786

8 Liu Y B, Cao X, Dai Q H, Xu W L. Continuous depth estimation for multi-view stereo. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. Miami, FL, IEEE, 2009
DOI:10.1109/cvpr.2009.5206712

9 Mustafa A, Kim H, Guillemaut J Y, Hilton A. Temporally coherent 4D reconstruction of complex dynamic scenes. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA, IEEE, 2016
DOI:10.1109/cvpr.2016.504

10 Chen S E, Williams L. View interpolation for image synthesis. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques-SIGGRAPH '93. New York, USA, ACM Press, 1993
DOI:10.1145/166117.166153

11 Zitnick C L, Kang S B, Uyttendaele M, Winder S, Szeliski R. High-quality video view interpolation using a layered representation. ACM Transactions on Graphics, 2004, 23(3): 600–608
DOI:10.1145/1015706.1015766

12 Li S, Zhu C, Sun M T. Hole filling with multiple reference views in DIBR view synthesis. IEEE Transactions on Multimedia, 2018, 20(8): 1948–1959
DOI:10.1109/tmm.2018.2791810

13 Kalantari N K, Wang T C, Ramamoorthi R. Learning-based view synthesis for light field cameras. ACM Transactions on Graphics, 2016, 35(6): 1–10
DOI:10.1145/2980179.2980251

14 Flynn J, Neulander I, Philbin J, Snavely N. Deep stereo: learning to predict new views from the world's imagery. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA, IEEE, 2016
DOI:10.1109/cvpr.2016.595

15 Zhou T, Tucker R, Flynn J, Fyffe G, Snavely N. Stereo magnification: Learning view synthesis using multiplane images. In SIGGRAPH, 2018

16 Hedman P, Philip J, Price T, FrahmJ-M, Drettakis G, Brostow G. Deep blending for free-viewpoint image-based

rendering. ACM Transactions on Graphics, 2018, 37(6):1−15
DOI:10.1145/3272127.3275084

17 Mildenhall B, Srinivasan P P, Ortiz-Cayon R, Kalantari N K, Ramamoorthi R, Ng R, Kar A. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG), 2019

18 Srinivasan P P, Tucker R, Barron J T, Ramamoorthi R, Ng R, Snavely N. Pushing the boundaries of view extrapolation with multiplane images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019, 175−184

19 He K, Sun J, Tang X. Guided Image Filtering. Berlin, Heidelberg, Springer Berlin Heidelberg, 2010,1−14

20 Brox T, Bruhn A, Papenberg N, Weickert J. High Accuracy Optical Flow Estimation Based on a Theory for Warping. Berlin, Heidelberg, Springer Berlin Heidelberg, 2004, 25−36