

A Real-time View-Dependent Shape Optimization for High Quality Free-Viewpoint Rendering of 3D Video

Shohei Nobuhara Wei Ning Takashi Matsuyama

Graduate School of Informatics, Kyoto University

{nob, ning, tm}@vision.kuee.kyoto-u.ac.jp

Abstract

This paper is aimed at proposing a new high quality free-viewpoint rendering algorithm of 3D video. The main challenge on visualizing 3D video is how to utilize the original multi-view images used to estimate the 3D surface, and how to manage the mismatches between them due to calibration and reconstruction errors. The key idea to solve this problem is to optimize the 3D shape on a per-viewpoint basis on the fly. Given a virtual viewpoint for visualization, our algorithm optimizes the 3D shape so as to maximize the photo-consistency over the surface visible from the virtual viewpoint. An evaluation demonstrates that our method outperforms the state-of-the-art rendering qualitatively and quantitatively.

1. Introduction

3D video is a visual media which captures the full-3D description of the real scene as is [11, 18]. It is a time-series of 3D mesh sequence reconstructed from multi-view videos, and many researchers have proposed algorithms to capture, process, transmit, and visualize it [1, 2, 9, 10, 19, 20].

This paper is aimed at proposing a new high quality free-viewpoint rendering algorithm of 3D video. The main challenge on visualizing 3D video is how to utilize the original multi-view images used to estimate the 3D surface, and how to manage the mismatches between them. That is, we need to select or synthesize a texture from multi-view real images for each part of the estimated 3D surface, while considering mismatches between the surface geometry and the images due to calibration and reconstruction errors.

The key idea to solve this problem in this paper is to optimize the 3D shape on a per-viewpoint basis on the fly. Given a virtual viewpoint for visualization, our algorithm optimizes the 3D shape so as to maximize the photo-consistency over the surface visible from the virtual viewpoint.

This approach is based on the following observations.

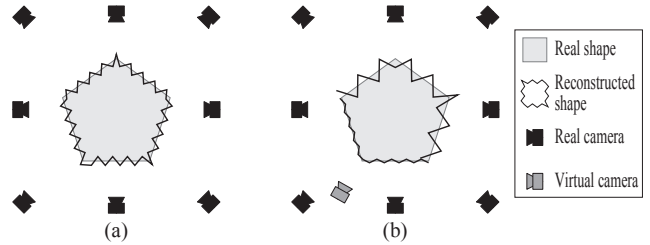


Figure 1: A view-dependent shape optimization concept. Compared with conventional view-independent 3D shape reconstruction (a), our method optimizes only for the area visible from the virtual camera for rendering (b).

In the typical 3D video production process, a 3D shape of the object is estimated from multi-view images captured by multiple cameras calibrated beforehand. In this process, the 3D shape is globally optimized so that the photo-consistency is maximized over the entire surface. This global optimization forces the process to return a 3D shape that is as valid as possible from any view directions, or we can say mismatches between the estimated 3D surface and the multi-view images due to calibration errors and/or shape estimation failures can exist everywhere on the surface (Figure 1(a)).

On the other hand, our view-dependent shape optimization intentionally ignores such mismatches if they are not visible from the virtual viewpoint of interest. Instead, our algorithm optimizes the 3D shape only for the local area visible from the virtual viewpoint. By means of this local optimization, we can expect that the photo-consistency over the locally visible area can be improved (Figure 1(b)).

The rest of this paper is organized as follows. Section 2 reviews studies related to 3D video visualization. Section 3 provides the algorithm and Section 4 describes how we can implement this algorithm with GPU. Section 5 shows qualitative and quantitative evaluations in comparison with state-of-the-art methods. Section 6 provides discussions and conclusions.

2. Related work

In order to improve the rendering quality of 3D video, many researchers have proposed algorithms based on different approaches. These approaches can be characterized by if they update the image, the calibration, or the 3D shape.

In [13], Matsuyama *et al.* have proposed a view-dependent per-vertex coloring algorithm to account for the calibration and reconstruction errors. They prioritize the cameras based on the distance to the virtual camera position for texture generation, but do not update the texture, the calibration and the 3D shape.

In [4], Eisemann *et al.* have proposed a concept named floating-textures which utilizes optical flows between renderings using different camera images. They warped mismatched renderings which can introduce blurs if mixed as is. Casas *et al.* also proposed an alignment-based blending technique to realize seamless transitions across 3D videos [3]. In [23], Takai *et al.* have proposed a concept of harmonized texture mapping in which they deformed the multi-view images based on the virtual camera position. These methods update the images, but keep the calibration and the 3D shape static.

In [6], Furukawa and Ponce have proposed an algorithm which refines the calibration based on the initial 3D shape reconstruction. In [25], Tung *et al.* have proposed a super-resolution technique which refines both the multi-view images and the 3D shape. In [8], Goldlücke and Cremers have proposed another super-resolution technique which operates on the target 3D surface. These methods refine the calibration, the images, or the shape, but they unnecessarily consider the surface areas which do not contribute to improve the renderings quality since the virtual camera cannot observe there. In addition they are not designed to run in real time.

The concept of view-dependent surface optimization has been introduced in [16] and [17]. In [16], Miller *et al.* have proposed optimizing visual hulls to be photo-consistent for each virtual rendering viewpoint given a priori. In [17], the view-dependent optimization is applied for each real camera, and the optimized per-camera depth-maps were used to synthesize novel views.

Unlike the above-mentioned methods, our algorithm refines the 3D shape for the current virtual viewpoint on the fly while optimizing the photo-consistency which directly affects the visual quality of the rendering. More specifically, it refines the 2.5D depth-map of the virtual viewpoint so that the photo-consistency on it is maximized. This process accounts for both the calibration and 3D shape reconstruction errors in a unified way, and also can run in real time as shown in the evaluation. In this sense, our method can be regarded as an image-based rendering using sparse multi-view cameras. Unlike regular image-based rendering systems relying on dense camera arrays [22, 24, 27], our al-

gorithm explicitly handles occlusions and the 3D geometry of the object to handle the sparse camera arrangement.

3. Algorithm

As describe in Section 1, the key idea of this paper is to refine the 3D surface visible from the virtual viewpoint of interest on the fly. This section first introduces the computational model, and then provides the algorithm to achieve the view-dependent shape optimization. In addition we introduce how to introduce a spatial consistency over virtual view changes.

3.1. Computational model

Suppose we have a set of multi-view real cameras $C = \{C_k\}$ ($k = 1, \dots, n$) in the scene. They are fully calibrated beforehand, and their calibration parameters allow us to project a 3D point $\mathbf{q} = (x, y, z)^\top$ in the world coordinate system by

$$d_k(u_k, v_k, 1)^\top = A_k(R_k \mathbf{p} + \mathbf{t}_k), \quad (1)$$

where A_k , R_k , and \mathbf{t}_k denote the intrinsic and extrinsic parameters of C_k , $(u_k, v_k)^\top$ is the projection of \mathbf{q} in C_k , and d_k is the depth of \mathbf{q} from C_k .

Given such a set of calibrated cameras, we assume that a 3D shape \mathcal{M} is estimated as a 3D video frame using the multi-view images $\mathcal{I} = \{I_k\}$ ($k = 1, \dots, n$) captured at an exact timing by the real cameras. Here we assume that the 3D shape is given as a triangle mesh surface denoted by $\mathcal{M} = \{f_i\}$ ($i = 1, \dots, m$), where f_i denotes a triangle facet. Also we introduce a virtual camera \hat{C} parameterized by its own intrinsic and extrinsic parameters \hat{A} , \hat{R} , and $\hat{\mathbf{t}}$ for rendering. These are the inputs of the algorithm.

The output of the algorithm is a refined 3D shape and its rendering. To this end, we introduce a 2.5D depth-map \hat{D} associated with the virtual viewpoint \hat{C} , and a subset of the real cameras $C_{\hat{p}}$ for each pixel \hat{p} of the depth-map \hat{D} such that its members can observe the 3D point corresponding to the pixel. We also denote $C_{\hat{p}}$ as “visible cameras” of \hat{p} .

3.2. Outline

As shown in Algorithm 1, our algorithm starts by initializing the depth-map \hat{D} of the virtual viewpoint \hat{C} and the visible cameras of all pixels $C_{\hat{p}}$ in the depth-map. Then we refine the depth-map by seeking the best depth for each pixel in \hat{D} in terms of both a photo-consistency and a smoothness measure between the neighboring pixels. Finally we render the virtual image using the refined depth-map.

3.3. Depth-map and visibility computation

Given a 3D shape and the calibration parameters of the real and virtual cameras, we first compute the depth-maps

Algorithm 1 Free-viewpoint rendering by view-dependent shape optimization

Input: R_k, t_k, A_k, I_k ($k = 1, \dots, n$), $\hat{R}, \hat{t}, \hat{A}, \mathcal{M}$

Output: \hat{I}

```

for each real and virtual camera  $C_k$  ( $k = 1, \dots, n$ ) and  $\hat{C}$  do
  Generate the depth-map  $D_k$  and  $\hat{D}$  by virtually rendering  $\mathcal{M}$ . # Section 3.3
end for
for each non-null pixel  $\hat{p}$  in  $\hat{D}$  do
  Compute the visible camera set  $C_{\hat{p}}$  using  $D_k$  ( $k = 1, \dots, n$ ) by Eq (2). # Section 3.3
end for
for each non-null pixel  $\hat{p}$  in  $\hat{D}$  do
  for each depth offset  $\delta_{\hat{p}}$  in  $[\delta_{\min} : \delta_{\max}]$  do
    Compute the photo-consistency of  $\hat{p}$  by assuming its depth is  $\hat{d}_{\hat{p}} + \delta_{\hat{p}}$  using Eq (4). # Section 3.4.1
  end for
end for
repeat
  for each non-null pixel  $\hat{p}$  in  $\hat{D}$  do
    Compute the belief-propagation message update using the photo-consistency and the smoothness constraint (Eq (6)). # Section 3.4.2
  end for
until convergence
  for each non-null pixel  $\hat{p}$  in  $\hat{D}$  do
    Assign the pixel color at the best depth returned by the belief-propagation to  $\hat{I}$ .
  end for

```

of both the real and the virtual cameras by projecting the 3D shape \mathcal{M} to them. Let $d_{p,k}$ denote the depth stored at a pixel p in the depth map D_k of a camera C_k . Similarly let $\hat{d}_{\hat{p}}$ denote the depth at $\hat{p} = (\hat{u}, \hat{v})^\top$ of \hat{D} . Using these real and virtual camera depth-maps, a camera C_k can be said visible for p if the following condition holds.

$$d_{p',k} \geq d'_{p',k}, \quad (2)$$

where

$$\begin{aligned}
 d'_{p',k}(u_{p',k}, v_{p',k}, 1)^\top &= A_k (R_k \mathbf{q}' + \mathbf{t}_k), \\
 \mathbf{q}' &= \hat{d}_{\hat{p}} (\hat{R}^{-1} \hat{A}^{-1} (\hat{u}, \hat{v}, 1)^\top + \hat{\mathbf{t}}).
 \end{aligned} \quad (3)$$

Here \mathbf{q}' denotes the 3D position on the 3D surface \mathcal{M} corresponding to the pixel \hat{p} in the virtual camera, and $p' = (u_{p',k}, v_{p',k})$ denotes its projection to the camera C_k . $d'_{p',k}$ and $d_{p',k}$ denote the distances from the projection center of C_k to \mathbf{q}' and the one to the closest fact in \mathcal{M} stored in D_k respectively. This function simply tests if the point \mathbf{q}' corresponding to \hat{p} is self-occluded in camera C_k 's view by the object \mathcal{M} or not.

In addition, this process labels if a pixel \hat{p} is a part of the projection, *i.e.* the silhouette, of \mathcal{M} . We call pixels outside of the silhouette as *null* pixels. In the following steps, we can skip null pixels safely.

3.4. View-dependent shape optimization

3.4.1 Photo-consistency

Once obtained the visible cameras for each non-null pixel of the virtual camera, we can compute how the photo-consistency changes by modifying the depth from the original distance. Let $\rho(\hat{p}, \hat{d}_{\hat{p}} + \delta_{\hat{p}})$ denote the photo-inconsistency of pixel \hat{p} with its depth modified by $\delta_{\hat{p}}$ defined as follows:

$$\begin{aligned}
 \rho(\hat{p}, \hat{d}_{\hat{p}} + \delta_{\hat{p}}) &= \\
 &= \frac{\sum_{C_k, C_{k'} \in C_{\hat{p}}} |I_k(p_k(\hat{p}, \hat{d}_{\hat{p}} + \delta_{\hat{p}})) - I_{k'}(p_{k'}(\hat{p}, \hat{d}_{\hat{p}} + \delta_{\hat{p}}))|^2}{C(|C_{\hat{p}}|, 2)},
 \end{aligned} \quad (4)$$

where $C(|C_{\hat{p}}|, 2)$ denotes the number of combinations selecting two cameras from $C_{\hat{p}}$, $p_k(\hat{p}, d)$ denotes the projection in C_k of the 3D point which appears at \hat{p} in the virtual camera at distance d . $I_k(p)$ denotes the intensity of the pixel p of I_k . Notice that this $\rho(\cdot)$ is defined as a photo-inconsistency. That is, we implement our view-dependent shape optimization as a photo-inconsistency minimization problem with considering a smoothness defined between neighboring pixels.

Also it should be noted that the set of visible cameras of each pixel is kept unchanged even though this process assumes different depths to compute the photo-consistency. This is based on the assumption that our view-dependent

shape optimization process does not introduce a shape modification which invalidates the original visibility. This point is justified by the evaluation using the visual hull in Section 5.

3.4.2 Edge-preserving smoothness constraint

In order to prevent making the 3D shape be overfitted, we employ the following edge-preserving smoothness constraint which controls the smoothness factor depending on the original depth gap between the pixel pair in question:

$$\sigma(\delta_{\hat{p}}, \delta_{\hat{p}'}) = \lambda_{\hat{p}, \hat{p}'} |\delta_{\hat{p}} - \delta_{\hat{p}'}|^2, \quad (5)$$

where \hat{p} and \hat{p}' are neighboring pixels, and $\lambda_{\hat{p}, \hat{p}'}$ is a scale factor inversely proportional to the original depth gap:

$$\lambda_{\hat{p}, \hat{p}'} = \frac{K}{1 + |d_{\hat{p}} - d_{\hat{p}'}|^2}. \quad (6)$$

Here K is a fixed parameter whose value is determined by hand to balance the smoothness and the photo-consistency. This σ enforces a larger smoothness for pixels with similar original depths, while it leaves distant pixels in depth be unconstrained.

3.4.3 Optimization

Up to this point, we defined the photo-inconsistency $\rho(\cdot)$ for each non-null pixel of the virtual camera, and the smoothness constraint $\sigma(\cdot)$ between the neighboring pixels. To find a best combination of $\delta_{\hat{p}}$ of all the pixels which minimizes these two factors, we render this as the following energy minimization problem:

$$\min_{\delta_{\hat{p}}} \left(\sum_{\hat{p} \in \hat{\mathcal{P}}} \rho(\hat{p}, \hat{d}_{\hat{p}} + \delta_{\hat{p}}) + \sum_{\hat{p}, \hat{p}' \in \hat{\mathcal{N}}} \sigma(\delta_{\hat{p}}, \delta_{\hat{p}'}) \right), \quad (7)$$

where $\hat{\mathcal{P}}$ denotes the non-null pixels of the virtual camera, and $\hat{\mathcal{N}}$ denotes neighboring pixel pairs in $\hat{\mathcal{P}}$. As is well-known, this optimization can be solved efficiently using a loopy belief propagation [5] if we discretize the possible value of $\delta_{\hat{p}}$ into a finite number of candidates, and convert the problem into a multi labeling problem. In practice we employ the checkerboard message computation scheme for efficiency [5].

Once obtained a solution, we can generate the final rendering of the scene using the best depth offset $\delta_{\hat{p}}$ of each pixel. In computing the final pixel color, we utilized a weighting strategy based on the similarity of the viewing directions between the virtual and each of the real cameras [13].

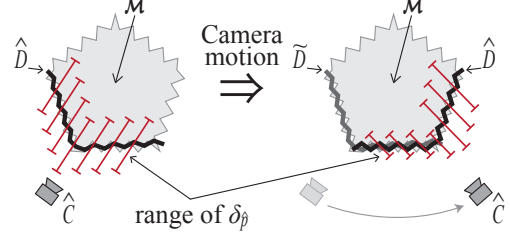


Figure 2: Biased optimization

3.4.4 Biased optimization for consistency across virtual camera moves

One important factor to provide a better experience for viewers is to keep the rendering be smoothly updated during the virtual camera moves. To this end, we utilize the optimization result of the previous camera position to initialize the depth $\delta_{\hat{p}}$ and limit the possible depth modifications $\delta_{\hat{p}}$.

Suppose the optimized depth-map at the previous virtual camera position is available as \tilde{D} in the current virtual camera coordinate system. Our biased optimization first compares \tilde{D} with \hat{D} , and labels non-null pixels of \hat{D} if they are null in \tilde{D} or not. If they are null in \tilde{D} , they correspond to the object surface newly appeared in the virtual camera view. Let such pixels be denoted as “newly non-null” pixels. Otherwise, they correspond to the surface optimized at the previous virtual camera position, and they should be kept consistent with the previous result for consistency. Let denote these pixels as “previously non-null” pixels (Figure 2).

Based on these observations, we optimize the newly non-null pixels as described above, while we replace the initial depth of the previously non-null pixels using the depth in \tilde{D} and make the maximum $\delta_{\hat{p}}$ be smaller in order to enforce the constraint.

Notice that this biased optimization can result in different visualization results at a same virtual camera position if the virtual camera trajectories to arrive at the viewpoint are different. This hysteresis is a limitation of this process.

4. Real-time implementation using GPU

This section discusses how the algorithm in Section 3 can be implemented to achieve a real-time visualization. In particular, we discuss how to implement this algorithm on GPU. This is because our goal is to render images from virtual viewpoints, and we can utilize massive parallel computation capability of recent GPUs as a natural consequence.

As described in Algorithm 1, the proposed algorithm consists of three steps: the visibility computation, the photo-consistency computation, and the optimization. An important observation here is that all these steps are pixel-independent. That is, visible cameras, photo-consistencies

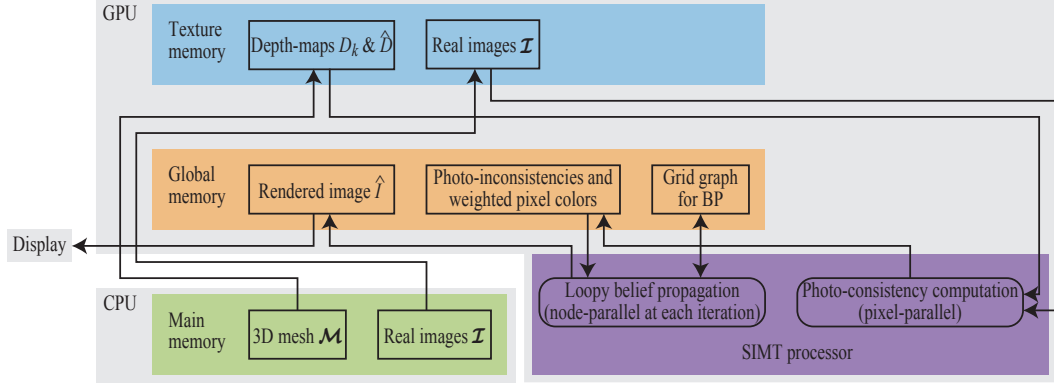


Figure 3: Block diagram of our GPU implementation.

at different depths, and messages exchanged at each iteration of the belief propagation can be computed on a per-pixel basis without considering the other pixels while referring to a same dataset such as the depth-maps D_k, \hat{D} and the original images \mathcal{I} . Hence we follow the single-instruction-multiple-threads (SIMT) programming model which is commonly available in recent GPUs. Also, we can observe that the photo-consistency computation for a pixel depends only on the visibility computation of the pixel, and does not need to wait for the visibility computations of the others. On the other hand, the belief propagation process needs to wait for the all the pixels to finish computing the photo-consistency. Therefore the parallel processing part can be implemented as follows: (1) simultaneous computation of visibility and photo-consistency, (2) a barrier to wait for the photo-consistency of all the pixels be ready, and (3) iterative simultaneous computation of message updates for the belief propagation.

Another important design point is how to use the memories in GPU. As described above, the depth-maps and the original images are shared by all the threads and are kept read-only from them. Hence we map them into a read-only memory space that is typically available as the texture memory. On the other hand, the other data including the computed photo-consistency values should be allocated at a writable global memory in GPU.

Based on these considerations, the block diagram of the GPU implementation can be designed as shown in Figure 3.

5. Evaluation

In order to evaluate the performance of our view-dependent shape optimization approach, this section provides experimental results using a real dataset. In this evaluation, we optimized the depth offset $\delta_{\hat{p}}$ in the range of $\pm 5.0\text{mm}$ at 0.5mm discretization step.

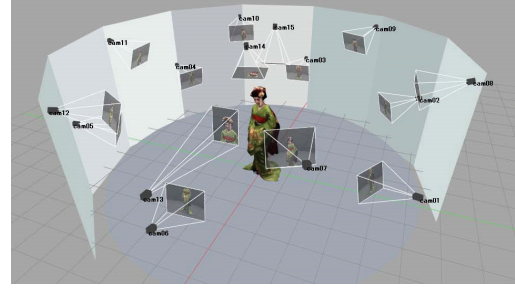


Figure 4: Camera arrangement.



Figure 5: Multi-view images.



(a) Visual hull.



(b) Photo hull.

Figure 6: View-independent 3D shapes.

Quantitative evaluation Figure 4 illustrates the camera arrangement of a studio of 6m diameter. This studio has 15 XGA cameras (Sony XCD-710CR) calibrated beforehand, and captures the target images as shown in Figure 5. Using

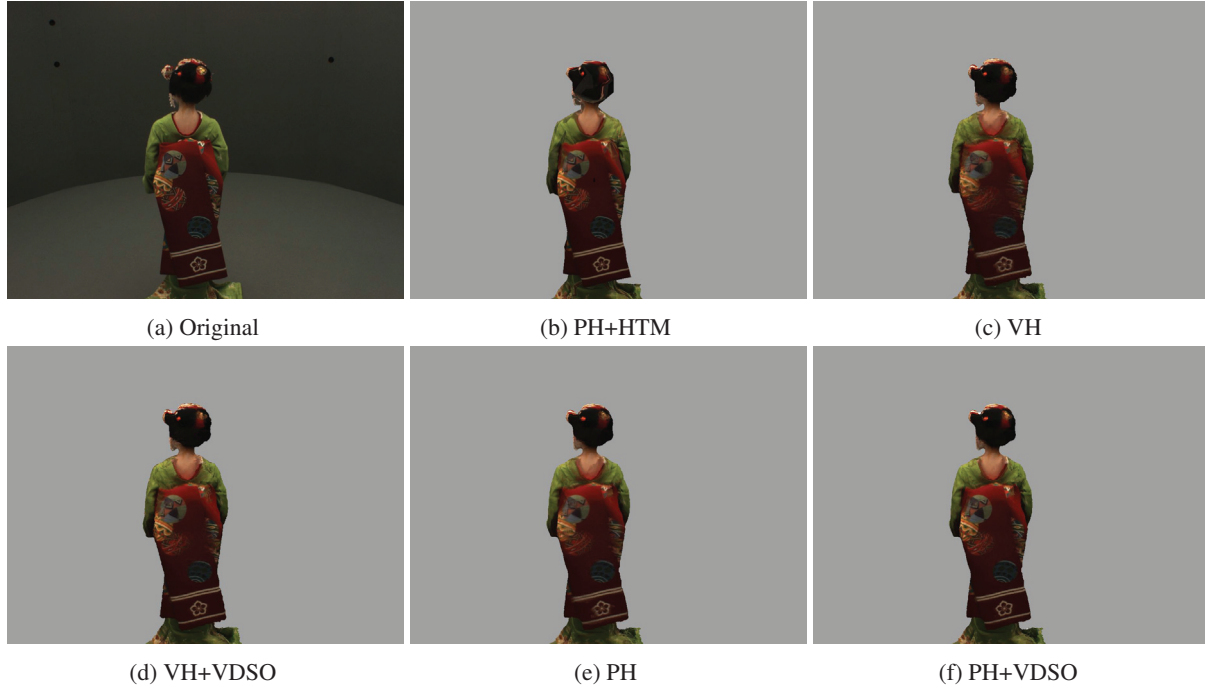


Figure 7: Rendering results at Camera 9.

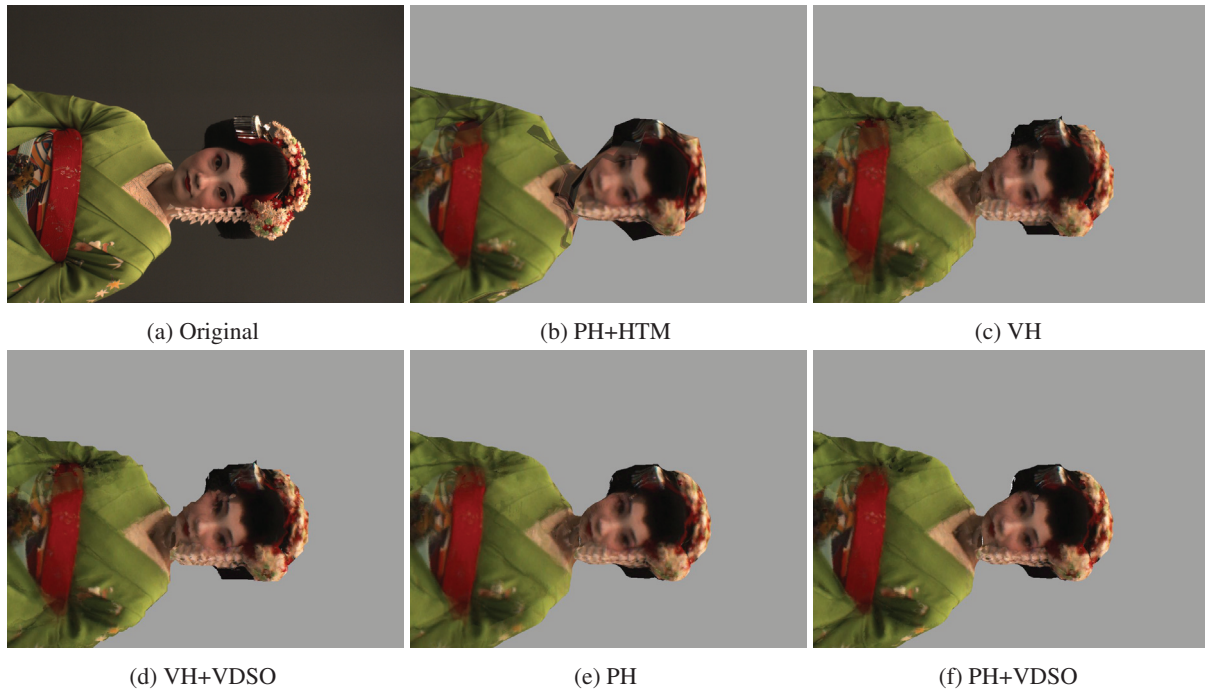


Figure 8: Rendering results at Camera 13.

the visual hull (VH) by shape-from-silhouette [12] as the initial 3D estimation, we obtained a photo hull (PH) which maximizes the photo-consistency on a view-independent manner [21] as shown in Figure 6. The resolution of these

mesh models (average length of triangle edges) is set to approximately 1cm.

Figures 7 and 8 show the rendering results by the proposed method (VDSO) and the state-of-the-art harmonized

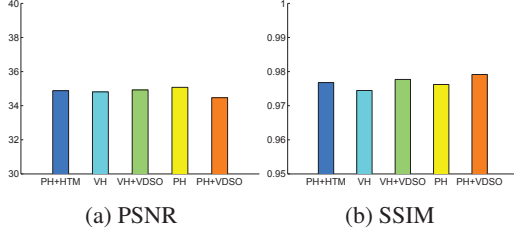


Figure 9: PSNR and SSIM of Figure 7

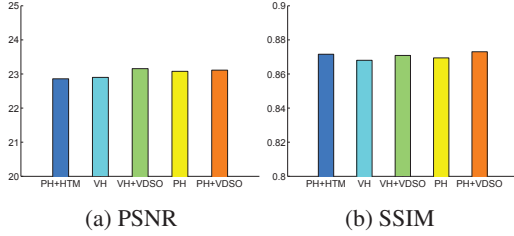


Figure 10: PSNR and SSIM of Figure 8

texture mapping method (HTM) [23]: (a) the original image as the ground truth, (b) rendering by HTM using PH as the input 3D shape, (c) rendering using VH, (d) rendering by VDSO using VH as the input 3D shape, (e) rendering using PH, and (f) rendering by VDSO using PH as the input 3D shape. These images are rendered from Cameras 9 and 13 in Figure 4 respectively, without using the original images. Therefore we can measure the rendering quality quantitatively by comparing the rendered and the original images in this leave-one-out test.

Figures 9 and 10 show PSNR and SSIM (structural similarity) [26] of the rendering results in Figures 7 and 8. Notice that the results in Figure 7 show much better scores than the ones in Figure 8. This is because the camera used for Figure 8 is an only one camera having a longer focal length. Since this is a leave-one-out evaluation, rendering for this camera using the other camera images is an upsampling process by definition.

We can observe that our view-dependent shape optimization can improve the rendering in comparison with the state-of-the-art qualitatively and quantitatively.

Quality impact by the initial 3D mesh The results shown in Figures 7, 8, 9 and 10 indicate that our VDSO approach can achieve a rendering quality comparable to the state-of-the-art HTM approach, even using the visual hull (Figure 6a).

This fact suggests that we can realize a fully image-based free-viewpoint 3D rendering pipeline with sparsely arranged multi-view cameras. That is, since our method uses the view-independent 3D shape \mathcal{M} only to provide depth-maps for the real and virtual cameras, we can replace

Table 1: Processing costs

Step	Cost (milliseconds)
Depth-map generation and visibility test	57 to 59
Photo-inconsistency computation	50 to 120
Global optimization	100 to 500

this part by an image-based visual hull computation method [7, 14, 15] which returns depth-maps directly from 2D multi-view silhouettes without computing the 3D geometry explicitly. Hence a possible system will capture multi-view images, extract silhouettes, and then feed them to GPUs so that all the time-consuming process are offloaded to them.

Processing cost In this experiment, we used a PC equipped with an Intel Core-i5 4670 3.4GHz CPU and an NVIDIA GeForce GTX 780 Ti GPU, and implemented our method using Direct3D and CUDA on Windows 7. Table 1 reports typical processing costs of each view-dependent shape optimization steps when rendering at XGA resolution (as same as the original camera). Other steps such as final rendering to the video buffer cost all within 10 milliseconds, and hence omitted from the table.

Obviously the optimization process is the most time-consuming part of the algorithm. In addition, its cost ranges from 100ms to 500ms, depending on the number of non-null pixels in the virtual camera. That is, if the camera zooms up to the object, then the GPU are fed more pixels to be processed. We believe these processing times can be reduced by using multiple GPUs, since these steps are all pixel-independent.

6. Conclusion

In this paper, we have introduced a view-dependent shape optimization concept to improve the visualization quality of 3D video in real-time. The evaluations showed that our method can provide a better rendering qualitatively and quantitatively, and our GPU implementation can run in real-time.

To further improve the rendering quality, we can integrate our method with a super-resolution technique. In particular, our method is well suited to techniques which run not on the object 3D surface but on the 2D rendering image such as [25] since our method provides only the surface visible from the rendering viewpoint of interest and all the processes are defined on a per-pixel basis of the virtual camera.

In addition, the fact that our view-dependent shape optimization can provide a comparable quality even using visual hulls indicates a possible scenario which realizes a fully image-based free-viewpoint 3D media production chain as

discussed in Section 5. That is, combined with an image-based visual hull algorithm [7, 14, 15], our method can be used to realize a real-time system which can offload all the time-consuming process including the visual hull computation, visibility computation, and photo-consistency maximization to GPUs. One possible application of this GPU-offloading is to provide a preview function for 3D video capture studios even during a capture session.

Moreover this can open a possibility to realize a practical free-viewpoint 3D *live* transmission pipeline for sparsely arranged multi-view camera systems which does the 2D processes (capture and silhouette extraction) in the capture side, and then transmits the 2D multi-view data via network with a conventional 2D video compression so that the viewer-sides can synthesize a novel view of the scene using their own GPUs. Realizing such a practical system is also a next step of this research work.

Acknowledgement

This research was partially supported by Joint Research with NTT Media Intelligence Laboratories.

References

- [1] J. Allard, J.-S. Franco, C. M  nier, E. Boyer, and B. Raffin. The grimace platform: A mixed reality environment for interactions. In *Proc. of International Conference on Computer Vision Systems*, pages 46–46, 2006.
- [2] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Transactions on Computer Graphics*, 22(3), 2003.
- [3] D. Casas, M. Volino, J. Collomosse, and A. Hilton. 4d video textures for interactive character appearance. *Proc. of EUROGRAPHICS*, 33(2), 2014.
- [4] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. *Proc. of EUROGRAPHICS*, 27(2):409–418, 2008.
- [5] P. Felzenszwalb and D. Huttenlocher. Efficient belief propagation for early vision. *IJCV*, 70:41–54, 2006.
- [6] Y. Furukawa and J. Ponce. Accurate camera calibration from multi-view stereo and bundle adjustment. *IJCV*, 84(3):257–268, 2009.
- [7] J. Goldfeather, J. P. M. Hultquist, and H. Fuchs. Fast constructive solid geometry display in the pixel-power graphics system. In *Proc. of ACM SIGGRAPH*, pages 107–116, 1986.
- [8] B. Goldl  cke and D. Cremers. Superresolution texture maps for multiview reconstruction. In *Proc. of ICCV*, pages 1677–1684, 2009.
- [9] O. Grau, G. Thomas, A. Hilton, J. Kilner, and J. Starck. A robust free-viewpoint video system for sport scenes. In *3DTV*, pages 1–4, 2007.
- [10] K. Hisatomi, K. Tomiyama, M. Katayama, and Y. Iwade. Method of 3d reconstruction using graph cuts, and its application to preserving intangible cultural heritage. In *IEEE Workshop on eHeritage and Digital Art Preservation*, pages 923–930, 2009.
- [11] T. Kanade, P. Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia*, pages 34–47, 1997.
- [12] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE TPAMI*, 16(2):150–162, 1994.
- [13] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation and high fidelity visualization for 3d video. *CVIU*, 96:393–434, 2004.
- [14] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proc. of ACM SIGGRAPH*, pages 369–374, 2000.
- [15] G. Miller and A. Hilton. Exact view-dependent visual hulls. In *Proc. of ICPR*, volume 1, pages 107–111, 2006.
- [16] G. Miller, A. Hilton, and J. Starck. Interactive free-viewpoint video. In *Proc of the 2nd European Conference on Visual Media Production*, pages 52–61, 2005.
- [17] G. Miller, J. Starck, and A. Hilton. Projective surface refinement for free-viewpoint video. In *Proc. 3rd European Conference on Visual Media Production*, pages 153–162, 2006.
- [18] S. Moezzi, L.-C. Tai, and P. Gerard. Virtual view generation for 3d digital video. *IEEE Multimedia*, pages 18–26, 1997.
- [19] A. Smolic, K. Mueller, P. Merkle, C. Fehn, P. Kauff, P. Eisert, and T. Wiegand. 3d video and free viewpoint video - technologies, applications and mpeg standards. In *Proc. of International Conference on Multimedia and Expo*, pages 2161–2164, July 2006.
- [20] J. Starck and A. Hilton. Surface capture for performance based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.
- [21] J. Starck, A. Maki, S. Nobuhara, A. Hilton, and T. Matsuyama. The multiple-camera 3-d production studio. *IEEE TCSVT*, 19(6):856–869, 2009.
- [22] Y. Taguchi, K. Takahashi, and T. Naemura. Real-time all-in-focus video-based rendering using a network camera array. In *3DTV Conference*, pages 241–244, May 2008.
- [23] T. Takai, A. Hilton, and T. Matsuyama. Harmonised texture mapping. In *Proc. of 3DPVT*, 2010.
- [24] M. Tanimoto, M. Tehrani, T. Fujii, and T. Yendo. Free-viewpoint tv. *IEEE Signal Processing Magazine*, 28(1):67–76, Jan 2011.
- [25] T. Tung, S. Nobuhara, and T. Matsuyama. Simultaneous super-resolution and 3d video using graph-cuts. In *Proc. of CVPR*, pages 1–8, 2008.
- [26] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004.
- [27] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *Proc. of ACM SIGGRAPH*, pages 600–608, 2004.