Light-weight Novel View Synthesis for Casual Multiview Photography

Inchang Choi, Yeong Beum Lee, Dae R. Jeong, Insik Shin, and Min H. Kim

KAIST School of Computing

Abstract. Traditional view synthesis for image-based rendering requires various processes: camera synchronization with professional equipment, geometric calibration, multiview stereo, and surface reconstruction, resulting in heavy computation, in addition to manual user interactions throughout these processes. Therefore, view synthesis has been available exclusively for professional users. In this paper, we address these expensive costs to enable view synthesis for casual users even with mobile-phone cameras. We assume that casual users take multiple photographs using their phone-cameras, which are used for view synthesis. First, without relying on any expensive synchronization hardware, our method can capture synchronous multiview photographs by utilizing a wireless network protocol. Second, our method provides light-weight image-based rendering on the mobile phone, where heavy computational processes, such as estimating geometry proxies, alpha mattes, and inpainted textures, are processed by a server to be shared in an interactable time. Finally, it allows us to render novel view synthesis along a virtual camera path on the mobile devices, enabling bullet-time photography from casual multiview captures.

Keywords: view synthesis · computational photography · multiview.

1 Introduction

Novel view synthesis from multiview photographs is an image-based rendering method and requires various computational processes to synthesize new viewpoints from captured photographs. Starting from camera synchronization with a professional camera synchronization hardware, heavy computational processes need to be conducted, such as camera tracking for unstructured cameras, geometric calibrations of camera properties, multiview stereo matching for dense point clouds, surface reconstruction, inverse rendering, and so on. Also, these computational processes require user interaction for synchronization, segmentation, inpainting, geometric reconstruction, etc. Therefore, novel view synthesis has been limited to professional setups, not available on casual computing devices, such as mobile phones.

In this work, by addressing these cost challenges of expensive hardware, heavy computations, and manual user interaction, our practical solution enables a casual, automated view synthesis for general users. We developed an efficient automated image-based rendering method on a casual setup that consists of multiple

mobile devices connected to a server with a wireless network. Once multiple mobile cameras capture a scene, our method creates bullet-time photography automatically played on the mobile devices, where it freezes time to navigate the scene from novel viewpoints.

Overview Figure 1 provides overview of our *automated* process of casual view synthesis for mobile multiview photography. Since multiple cameras need to capture a scene simultaneously, we first develop a synchronization method for mobile camera devices using a network time protocol (Section 3.1). We then transfer synchronous photographs to a server that computes geometry and texture elements. In the geometry processing step, we estimate the camera parameters of the unstructured (hand-held) mobile devices and the point clouds, which forms the geometry proxy of the scene (Section 3.2). Concurrently, we convert multiview photographs to texture elements for rendering. In this step, we separate the foreground and background objects in input images. We also inpaint occlusions in background images rapidly (Section 3.3). Once conversion processes are finished, graphics components are transferred to mobile devices through a wireless network, yielding real-time rendering of a virtual scene along the virtual camera path that shows bullet-time photography (Section 3.4).

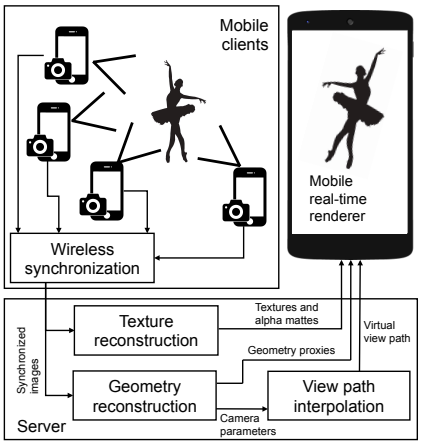

Fig. 1. Overview of our view synthesis.

2 Related Work

Prior works in image-based rendering synthesize novel views from a set of images. They can be classified into three categories according to what geometry information is required [7]. First, a light field can represent light transport of a scene, which can be interpolated to render a novel view from a captured light field. Instead of requiring the geometry information of a scene, light-field rendering exploits a large number of image data. These images are sampled from a dense and uniform camera grid, to approximate the plenoptic function of the scene. To avoid uniform grid sampling, Buehler et al. [1, 14, 13] proposed unstructured lumigraph rendering, but it requires a rough geometry proxy of an object. Second, view-dependent texture mapping (VDTM) can render a novel view by utilizing the geometry model of a scene as an input [17]. Debevec et al. [5] proposed a VDTM method, where images are projected on to a given input geometry model and a new view is rendered with those projected textures. However, the geometry models used in VDTM need to be either modeled manually or scanned by a 3D

scanner. Sinha et al. [16] proposed a view synthesis that uses a geometry model produced by approximating planes on the scene, but the approximation is valid only for planar objects such as buildings. Lastly, depth image-based rendering (DIRB) utilizes a camera projection mechanism and textured depth layers [4]. Instead of taking a geometry model as an input, it estimates the depth information from the corresponding feature points between stereo images, but this approach restricts a novel viewpoint to be close to the real viewpoints. A few studies have performed view synthesis using depth maps [11] or layered depth images [3, 15]. Unlike these prior works, we generate geometry proxies from the point cloud of the scene using a multiview stereo approach [18], instead of using any scanned geometry or depth maps. This work enables an automated, efficient image-based rendering with plausible view synthesis. The geometry proxies in our method are subject to a two-pass rendering process like the textured depth layers in DIRB. Refer to Section 3.2 for more details.

Recently, Wang et al. [21] introduced a novel view synthesis method, targeting a mobile platform. The study explores potentials of mobile application preliminarily; however, it does not focus on both automation and efficiency of view synthesis, taking *more than an hour* to process with the help of manual input for each stage. To address the impracticability of the prior work, we focus on both *automation and efficiency* for casual view synthesis, enabling us to produce plausible results for about a minute on a mobile platform even without requiring manual interactions.

3 Casual View Synthesis

3.1 Mobile Camera Synchronization

For practical synchronization for the casual setup, we adopt the network time protocol (NTP) [12] to synchronize the clock of every mobile device. In our setup, one of the mobile devices acts as a sync host device that triggers shooting, and the others work as clients (Figure 2). To synchronize the clocks of devices, all the client devices send the host device an NTP request packet that records a timestamp t_0 of the time when the packet is sent. When the host device receives the packet, it records a timestamp t_1 on the packet and then returns it to the client with a new timestamp t_2 when it departs. The client then records a timestamp t_3 when it receives the packet.

Finally, the time difference t_d between devices is calculated as $\{(t_1 - t_0) + (t_2 - t_3)\}/2$ to adjust the clock on each client device. Each device's clock is then updated by adding t_d on its clock from the time of receiving the return packet. When the host mobile device triggers the synchronous shooting command to make

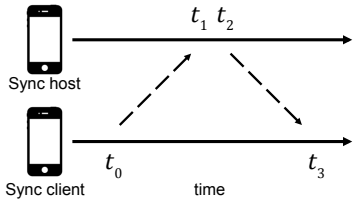

Fig. 2. Packet-based synchronization.

each client device capture a scene si-
multaneously, we allow a short delay of the one-way trip time of the packet t_r as $\{(t_3 - t_0) - (t_2 - t_1)\}/2$ with an additional allowance of approximately $\sim$100 milliseconds to account for the fluctuation of the wireless network speed in our experiments. Note that since the accuracy of NTP is highly affected by the network environments, all mobile devices are connected via WiFi Direct [2] to minimize any potential error caused by the network environments.

3.2 Automated Estimation of Geometry Proxies

Since the server receives multiple synchronous photographs, it performs the bundle adjustment [20] to obtain point cloud representations of the scene with camera parameters and then approximates geometry proxies of surfaces from clouds for rendering. Through this process, we obtain the point clouds of the foreground and background objects and camera parameters. From the estimated point clouds, we generate geometry proxies for rendering.

Multiview Stereo We follow the standard multiview stereo method, so-called structure-from-motion, to obtain point clouds and camera parameters. Given a set of synchronized multiview images $\mathbf{I} = \{I_1, I_2, \ldots, I_n\}$, we estimate a set of point clouds $\mathbf{P} = \{P_1, P_2, \ldots, P_m\}$ of the scene and a set of camera parameters $\mathbf{C} = \{C_1, C_2, \ldots, C_n\}$. The bundle adjustment process begins by performing feature matching between input images [20], where SIFT feature points [9] are paired and filtered through RANSAC [6]. We use a set of n camera devices to yield a set of m point clouds, where q_{ij} is the *observed* projection of feature points of the j-th point cloud in the i-th camera. Then the objective function for obtaining both $\mathbf{P}$ and $\mathbf{C}$ can be defined using a pinhole-projection function Φ that projects point P_j to image I_i of camera C_i as $\min_{\boldsymbol{C},\boldsymbol{P}} \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \left\| q_{ij} - \Phi\left(C_i, P_j\right) \right\|^2$, where the weight term w_{ij} is set to 1 when P_j is visible from C_i so that P_j exists in image I_i. Otherwise, it is set to 0. The objective function is solved through a state-of-the-art parallel optimization method [22]. Figures 3(a) shows an example of a point cloud of a scene and the camera frustums estimated by the bundle adjustment. Figure 3(b) shows the estimated geometric proxies for the foreground and background objects.

Geometry Proxy We next separate the point cloud $\mathbf{P}$ into two sets using the k-means clustering algorithm [10]. The k-means clustering algorithm with $k = 2$ quantizes the input point cloud $\mathbf{P}$ to the point cloud for an object in the foreground $\mathbf{F}$ and the point cloud of the background $\mathbf{B}$. The clustering process is performed by optimizing the cost function: $\min_{\boldsymbol{F},\boldsymbol{B}} \sum_{P \in \boldsymbol{F}} \|P - \mu_{\boldsymbol{F}}\|^2 + \sum_{P \in \boldsymbol{B}} \|P - \mu_{\boldsymbol{B}}\|^2$, where $\mu_{\boldsymbol{F}}$ and $\mu_{\boldsymbol{B}}$ indicate the center of the foreground and the background cluster, respectively. Given the separated point clouds, we generate geometry proxies for them. For the foreground point cloud $\mathbf{F}$, we build a cylinder of radius r, of which the center is $\mu_{\boldsymbol{F}}$. The background point cloud $\mathbf{B}$ is approximated by a plane containing the center point $\mu_{\boldsymbol{B}}$ with the normal vector of $(0, 0, 1)$. We denote the foreground cylindrical proxy and the background planar as $\mathbf{G}^{\mathcal{F}}$ and $\mathbf{G}^{\mathcal{B}}$, respectively.

3.3 Efficient Production of Textures

We found that the prior work [21] is based on a piece-wise planar stereo approach [16], which is significantly expensive and less compatible with ordinary scenes to determine depth. We therefore introduce a novel rendering workflow that combines learning-based segmentations of synchronized photographs with an efficient rendering approach based on geometry proxies to achieve both efficiency and plausibility. To this end, we convert multiview photographs to textures of the foreground and the background for layered geometric structures.

Foreground Segmentation Since we produce two separate geometry proxies of the foreground and the background, we segment the input images into two sets of textures, respectively. To exclude any necessity of extra user inputs, we adopt a state-of-the-art semantic image segmentation method based on a convolutional neural network (CNN) [23]. This semantic segmentation method enables us to generate the foreground masks $\mathbf{M}^{\mathcal{F}} = \{M_1^{\mathcal{F}}, M_2^{\mathcal{F}}, \ldots, M_n^{\mathcal{F}}\}$ and the background masks $\mathbf{M}^{\mathcal{B}} = \{M_1^{\mathcal{B}}, M_2^{\mathcal{B}}, \ldots, M_n^{\mathcal{B}}\}$ for the input images $\mathbf{I} = \{I_1, I_2, \ldots, I_n\}$ without any user input. Masks $M_i^{\mathcal{F}}$ and $M_i^{\mathcal{B}}$ for image I_i are binary-valued arrays, where i represents that the pixel is classified as the corresponding label. Note that we assume that the main subject for multiview photography is one or more people for simplicity. The masks are then used for alpha-blending when rendering view synthesis.

Background Inpainting Using the foreground and background masks, we obtain two sets of textures $\mathbf{T}^{\mathcal{F}} = \{T_i^{\mathcal{F}} | T_i^{\mathcal{F}} \triangleq I_i \circ M_i^{\mathcal{F}}\}$ and $\mathbf{T}^{\mathcal{B}} = \{T_i^{\mathcal{B}} | T_i^{\mathcal{B}} \triangleq I_i \circ M_i^{\mathcal{B}}\}$, where $\circ$ is the Hadamard product operator. There exist empty regions in each background texture in $\mathbf{T}^{\mathcal{B}}$ caused by the background mask. The missing regions particularly in the background occur by the occlusion of the foreground object. It causes inevitable artifacts when a virtual camera V_k renders a novel, synthetic view. To alleviate this occlusion problem, we fill the empty regions by applying an inpainting algorithm. Targeting the interactable performance of our method, we employ a fast-marching inpainting algorithm [19] that approximates unknown pixel values with the weighted sum of pixel known neighboring pixels. After holes are inpainted in the background textures $\mathbf{T}^{\mathcal{B}}$, they are passed to the

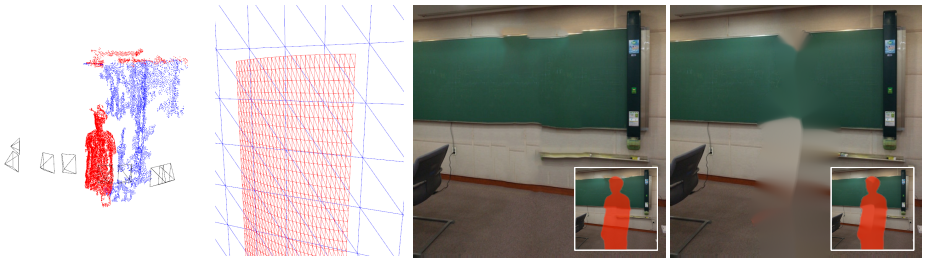

Fig. 3. An example of estimated point clouds (a) and proxy geometry (b) of a scene. (c) is inpainted by a Laplacian-based method [8] in 362 secs. (d) is completed by a fast-marching method [19] that we chose in 1.07 secs. Insets show inpainted regions.

rendering pipeline on a mobile device, together with the foreground texture $\mathbf{T}^{\mathcal{F}}$, to synthesize novel frames. Figures 3 (c) and (d) compare background textures by different inpainting methods.

3.4 Automated Image-based Rendering

For synthetic view rendering, we aim to automatically create novel view synthesis for bullet-time photography to yield a frozen-time animation. We therefore generate a trajectory array of virtual cameras at novel viewpoints from the array of the real cameras.

Virtual Camera Orientations We sort the set of camera parameters $\mathbf{C} = \{C_1, C_2, \ldots, C_n\}$ by the x coordinates of the cameras, and we assume that C_1 indicates the leftmost located camera and C_n refers to the rightmost located camera. From the i-th camera parameters C_i, we then extract orientation parameters for rotation R_{C_i} and position parameters for translation T_{C_i}, respectively. The intrinsic parameters of the virtual cameras are set to those of the real cameras. See Figure 4 for our view path generation.

Suppose we want to build a set of l novel virtual camera parameters $\mathbf{V} = \{V_1, V_2, \ldots, V_l\}$. The orientation parameters R_{V_k} of the k-th virtual camera V_k are estimated by accounting for neighboring cameras' orientations $R_{\mathbf{C}}$ with spatially-varying weights so that the virtual cameras look at a similar position in the scene. We found that this makes users feel comfortable while watching view synthesis. The weight is defined by the distance between the virtual camera V_k and the i-th real camera C_i in the camera set $\mathbf{C}$. The rotation parameter R_{V_k} is calculated as follows:

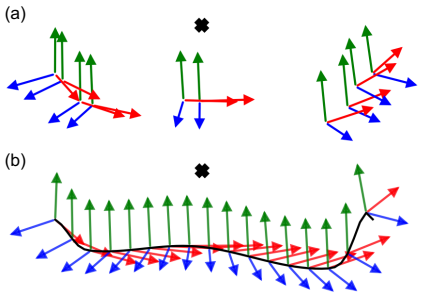

Fig. 4. (a) an object at the cross position is captured by ten cameras. (b) presents the interpolated virtual camera frames.

$$R_{V_k} = \frac{\sum_{i=1}^{n} w_{ik} R_{C_i}}{\sum_{i=1}^{n} w_{ik}}, \qquad w_{ik} = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\|T_{V_k} - T_{C_i}\|^2}{2\sigma^2}}$$

where the weight w_{ik} is set to have a Gaussian distribution of the distance between the virtual camera position T_{V_k} and the i-th real camera position T_{C_i}. Therefore, the interpolated orientation of the virtual camera is more influenced by that of closer real cameras than those of distant real cameras. The parameter σ controls how fast the weight decreases as the distance between two camera positions increases.

Virtual Camera Positions We first create a set of k virtual cameras that are uniformly distributed along the x-axis from the leftmost located camera $T^x_{C_1}$ to the

rightmost camera $T^x_{C_n}$. For both y and z positions, we interpolate intermediate values using a cubic spline function $\Psi(x)$ for given x coordinates. For instance, we determine y coordinates using $y = \Psi_y(x)$ in the x-y plane using the set of original camera points $\{(T^x_{C_i}, T^y_{C_i})|C_i \in \mathbf{C}\}$, where the spline function $\Psi_y(x)$ is in the form of the third-order polynomials. The corresponding z-coordinates of the virtual cameras are computed by the same procedure, estimating a cubic spline $\Psi_z(x)$ spanning the x-z plane.

Two-Pass Rendering We have prepared all the graphics components including the foreground/background geometries $\mathbf{G}^{\mathcal{F}}$ and $\mathbf{G}^{\mathcal{B}}$, the virtual viewpoints $\mathbf{V}$, and the foreground and background textures $\mathbf{T}^{\mathcal{F}}$ and $\mathbf{T}^{\mathcal{B}}$ in addition to the alpha masks $\mathbf{M}^{\mathcal{F}}$ for the foreground object. For a target virtual viewpoint V_k, we first retrieve the real camera C_m having the minimum distance to V_k. We adopt a two-pass rendering approach that draws the background object and the foreground object separately and blends the two images by a generated alpha matte. The procedure of rendering the background starts from unprojecting texture T on geometry g from the perspective of the real camera C. The scene from the target viewpoint V_k is then rendered. Rendering the foreground object is performed similarly, but it further outputs an alpha matte of the foreground object by warping the binary mask M from the view of C to V_k. Since we use a geometry proxy $\mathbf{G}^{\mathcal{F}}$, which is the very rough approximation for the real geometry of the object, the alpha matte is necessary to refine the rough rendering, not in the model space but the image space. As mentioned earlier, the final output frame is produced via alpha blending of two intermediate images. We iterate the rendering procedure for all virtual cameras in $\mathbf{V}$.

4 Results

We conducted experiments on our view synthesis workflow using eight Google Nexus 5X mobile phones. Our server is equipped with Intel Core i7-3770 CPU with 32 GB memory and NVIDIA GeForce GTX 970 GPU. We took images of a scene in 1440×1080 resolution, but they were scaled to 800×600 when transmitting to the server for both computational and communicational efficiency.

Novel View Synthesis Given the accurately synchronized mobile devices, we generated synthetic views using our view synthesis method. In Figure 5, we present interpolated novel views (b) between two real camera views, shown in Images (a) and (c). These novel views in Column (b) were created to enable the bullet-time effect along the virtual camera path automatically. Image (d) presents a point cloud with real camera frustums. The two rightmost camera frustums correspond to two real views shown in (a) and (c), respectively. Images (e) and (f) reveal intermediate geometry proxies for projective camera mapping and an alpha map for foreground segmentation yielding the foreground image (g), which is rendered on top of the inpainted background (h). Six cameras were used to capture this scene.

Figure 6 compares our view synthesis of bullet-time photography to a state-of-the-art method [21]. Wang et al. [21] adopt a piecewise planar stereo method [16]

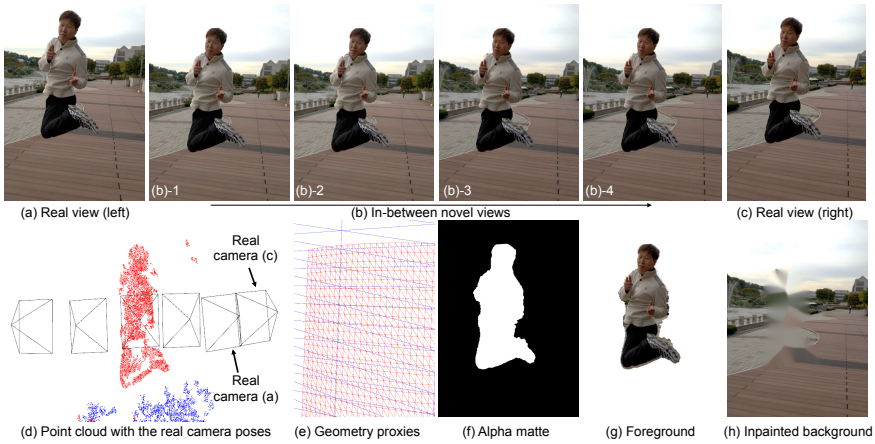

Fig. 5. Automated view interpolation for bullet-time photography along with a synthetic view path. Between two real views (a) and (c), we created intermediate views (b)s synthetically. Image (d) shows a point cloud with the real camera frustums, where the two rightmost frustums show real views (a) and (c) in the upper row, respectively. Images (e) – (h) present graphics components for the image-based rendering of a synthetic view: geometric proxies, an alpha mask, a segmented foreground image, and an inpainted background image.

Fig. 6. Images (a) and (b) compare view interpolations of our method with Wang et al.

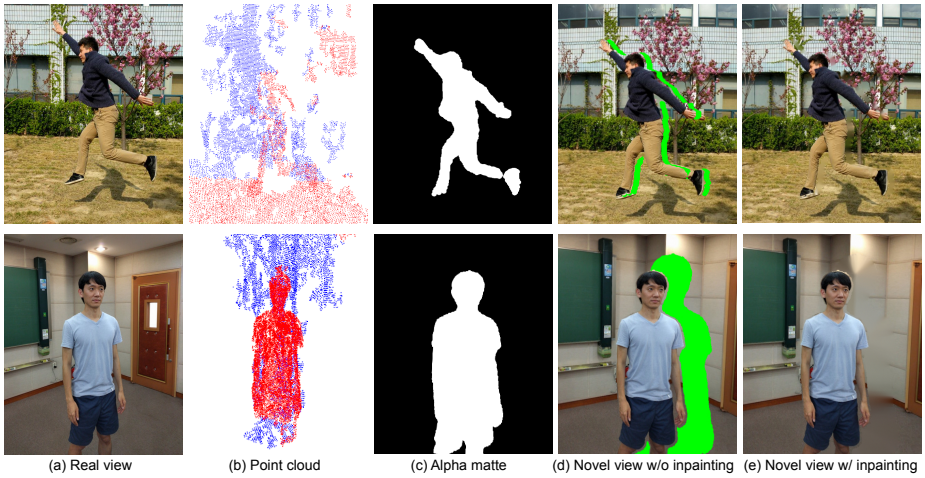

Fig. 7. Column (a) shows an input image. Columns (b) and (c) demonstrate intermediate point clouds and corresponding alpha mattes. Columns (d) and (e) compare novel view images with/without background inpainting. Column (e) presents a synthetic novel view.

[Unit: second]

Wang et al.			Ours		
Geometry processor	Bundle Adjustment (BA)	11.00	Geometry processor	Bundle Adjustment (BA)	11.00
	3D Line Reconstruction (LR)	1.13		Proxy Generation (PG)	0.0020
	Plane Detection (PD)	13.00	Texture processor	Segmentation	57.92
				Background Inpainting (BI)	9.54
	Depth-map Generation (DG)	4481.00	View path generator	Point and Pose Estimation (PPE)	0.33
Total		1hour 15mins. (=4506.13secs)	Total		1min. 18secs. (=78.79 secs)

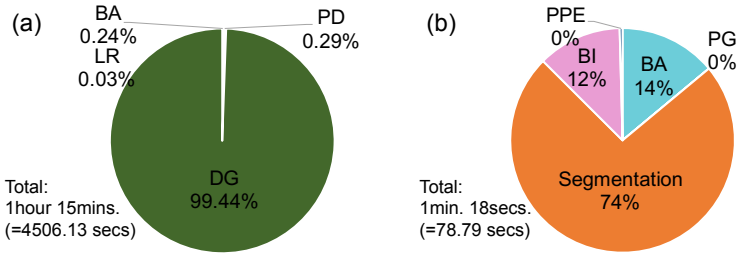

Fig. 8. Performance analysis for computing a bullet-time video. The table on the top compares the time performance in seconds for each step of Wang et al. [21] and our method. The pie charts (a) and (b) compare the proportion of computational times spent for each step of both methods, respectively.

to estimate depth information of the scene for depth image-based rendering. While the planar stereo method is devised initially to estimate the depth of an urban scene that comprises many planar objects such as walls, houses, buildings, etc., we found that it often fails for natural outdoor scenes and human subjects, as shown in Figure 6(a). Therefore, Wang et al. present frequent DIBR artifacts over rendering results due to erroneous depth estimation. In contrast, our method outperforms the state-of-the-art method regarding DIBR rendering artifacts.

Figure 7 shows additional results, where input real views, point clouds, alpha mattes, and synthesized novel views are presented in each column. Columns (d) and (e) compare the impact of our background inpainting. The inpainting process of the background helps the novel views appear more natural and plausible without requiring severe computational costs. Refer to the supplemental material for more video results.

Performance Analysis Although our view synthesis is rendered on a mobile device in an interactable time, we have to employ a server to perform heavy computational pre-processing such as texture segmentation, bundle adjustment for geometry, and view path generation to achieve interactive performance. We evaluate a performance analysis on our system by measuring the running time for each process to process a scene. Figure 8 compares the computational costs of our method with Wang et al. [21]. The total running time for our method was 78.79 seconds (1 minute 18 seconds). In contrast, the state-of-the-art method of Wang et al. took 4506.13 seconds (1 hour 15 minutes) for handling the same scene. Note that our method is near the interactive time, which is about 57 times faster than the other method. The most time-consuming step in our method is the semantic segmentation part, which took 57.92 seconds; i.e., it accounts for about 74 % of the total processing time. We believe that the bottleneck of the segmentation step can be alleviated by substituting the current segmentation method with a more efficient state-of-the-art method in the future.

Camera Synchronization To assess the accuracy of our synchronization, we experimented with taking synchronized images of a running stopwatch. The stopwatch is capable of displaying time in milliseconds. Using eight mobile devices, we performed ten trials to capture the synchronized images of the time on the stopwatch. We demonstrate the results of the experiment in Figure 9. As a criterion for the precision, we used the standard deviation of the sample times captured in the synchronized images. Smaller standard deviation indicates less error for synchronization. The average standard deviation for ten trials was only 15.77 milliseconds. For the fourth and sixth trials, the standard deviations were virtually zero, meaning no errors.

5 Discussion and Conclusion

In summary, we have presented an automated casual view synthesis method that bridges the gap between pervasive mobile computing and multiview photography. Our method enables bullet-time photography of frozen-time animation on

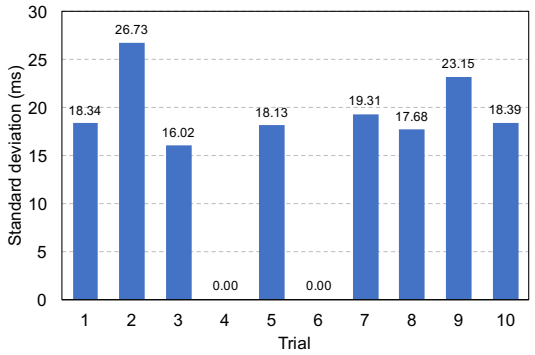

Fig. 9. Synchronization results of ten trials using eight mobile devices. The vertical axis shows the standard deviation of the captured time differences in milliseconds. The horizontal axis represents the trial number. We had the perfect synchronization of eight devices for the 4^{th} and 6^{th} trials. The average standard deviation was only 15.77 milliseconds.

wirelessly connected mobile devices in an interactable time. The current method is implemented for still shots. The synthetic video navigation will be our future work. The performance of our synthetic view rendering could be affected by the accuracy of the CNN-based semantic segmentation algorithm [23] that we used. An advanced segmentation method could improve results.

Acknowledgements

Min H. Kim acknowledges Korea NRF grants (2019R1A2C3007229, 2013M3A6A-6073718) and additional support by Cross-Ministry Giga KOREA Project (GK17-P0200), Samsung Electronics (SRFC-IT1402-02), ETRI(19ZR1400), and an ICT R&D program of MSIT/IITP of Korea (2016-0-00018).

References

1. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '01, ACM (2001)
2. Camps-Mur, D., Garcia-Saavedra, A., Serrano, P.: Device-to-device communications with wi-fi direct: overview and experimentation. IEEE Wireless Communications **20**(3), 96–104 (June 2013)
3. Chang, C.F., Bishop, G., Lastra, A.: Ldi tree: A hierarchical representation for image-based rendering. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co. (1999)
4. 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 (1993)
5. Debevec, P., Yu, Y., Borshukov, G.: Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. Springer Vienna (1998)

6. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**(6) (Jun 1981)
7. Kang, S.B., Shum, H.Y.: A review of image-based rendering techniques. Institute of Electrical and Electronics Engineers, Inc. (June 2000)
8. Lee, J.H., Choi, I., Kim, M.H.: Laplacian patch-based image synthesis. In: Proc. IEEE Computer Vision and Pattern Recognition (CVPR 2016). pp. 2727–2735. IEEE, Las Vegas, USA (2016)
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**(2), 91–110 (2004)
10. Macqueen, J.: Some methods for classification and analysis of multivariate observations. In: In 5-th Berkeley Symposium on Mathematical Statistics and Probability. pp. 281–297 (1967)
11. McMillan, Jr., L.: An Image-based Approach to Three-dimensional Computer Graphics. Ph.D. thesis, Chapel Hill, NC, USA (1997), uMI Order No. GAX97-30561
12. Mills, D.L.: Internet time synchronization: the network time protocol. IEEE Transactions on Communications **39**(10), 1482–1493 (Oct 1991)
13. Scharstein, D.: Stereo vision for view synthesis. In: Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 852–858 (Jun 1996)
14. Seitz, S.M., Dyer, C.R.: View morphing. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. pp. 21–30. SIGGRAPH '96, ACM, New York, NY, USA (1996)
15. Shade, J., Gortler, S., He, L.w., Szeliski, R.: Layered depth images. In: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques. pp. 231–242. SIGGRAPH '98, ACM, New York, NY, USA (1998)
16. Sinha, S., Steedly, D., Szeliski, R.: Piecewise planar stereo for image-based rendering. In: Twelfth IEEE International Conference on Computer Vision (ICCV 2009). IEEE, Kyoto, Japan (September 2009)
17. Siu, A.M.K., Lau, R.W.H.: Image-based modeling and rendering with geometric proxy. In: Proceedings of the 12th ACM International Conference on Multimedia, New York, NY, USA, October 10-16, 2004. pp. 468–471 (2004)
18. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring photo collections in 3d. In: SIGGRAPH Conference Proceedings. pp. 835–846. ACM Press, New York, NY, USA (2006)
19. Telea, A.: An image inpainting technique based on the fast marching method. J. Graphics, GPU, & Game Tools **9**(1), 23–34 (2004)
20. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment - a modern synthesis. In: Proceedings of the International Workshop on Vision Algorithms: Theory and Practice. pp. 298–372. ICCV '99, Springer-Verlag, London, UK, UK (2000)
21. Wang, Y., Wang, J., Chang, S.: Camswarm: Instantaneous smartphone camera arrays for collaborative photography. CoRR **abs/1507.01148** (2015)
22. Wu, C., Agarwal, S., Curless, B., Seitz, S.M.: Multicore bundle adjustment. In: Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition. IEEE Computer Society (2011)
23. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.S.: Conditional random fields as recurrent neural networks. CoRR (2015)