LATENCY COMPENSATION THROUGH IMAGE WARPING FOR REMOTE RENDERING-BASED VOLUMETRIC VIDEO STREAMING

Serhan Gül, Cornelius Hellge, Peter Eisert

Fraunhofer HHI, Berlin, Germany

ABSTRACT

Rendering multiple high-quality volumetric videos is still a challenge for today's mobile devices. Remote rendering offloads complex rendering operations to a powerful server and provides the final result to the end device as a 2D video stream. A drawback of remote rendering is the significant increase of interaction latency that can degrade the user experience. We present a planar homography-based approach that compensates minor changes of the user's head pose due to the interaction latency by warping the transmitted image on the client-side, just before it is sent to the display. In detail, we use the homography between the initial head pose when the image is rendered at the server and the latest available head pose of the user at the client. We perform controlled experiments using artificial camera traces to evaluate our approach. The results show that the proposed approach reduces the rendering errors significantly in terms of the mean-squared error between the rendered and reference images, especially combined with initial head motion prediction.

***Index Terms*—** volumetric video, remote rendering, low latency, mixed reality, augmented reality

1. INTRODUCTION

Volumetric video enables a variety of upcoming applications in several domains including cultural heritage [1, 2] and healthcare [3]. The geometry of volumetric objects is usually represented using meshes or point clouds, while high-quality volumetric meshes typically contain thousands of polygons. Therefore, rendering multiple complex volumetric videos is a demanding task for today's mobile devices [4].

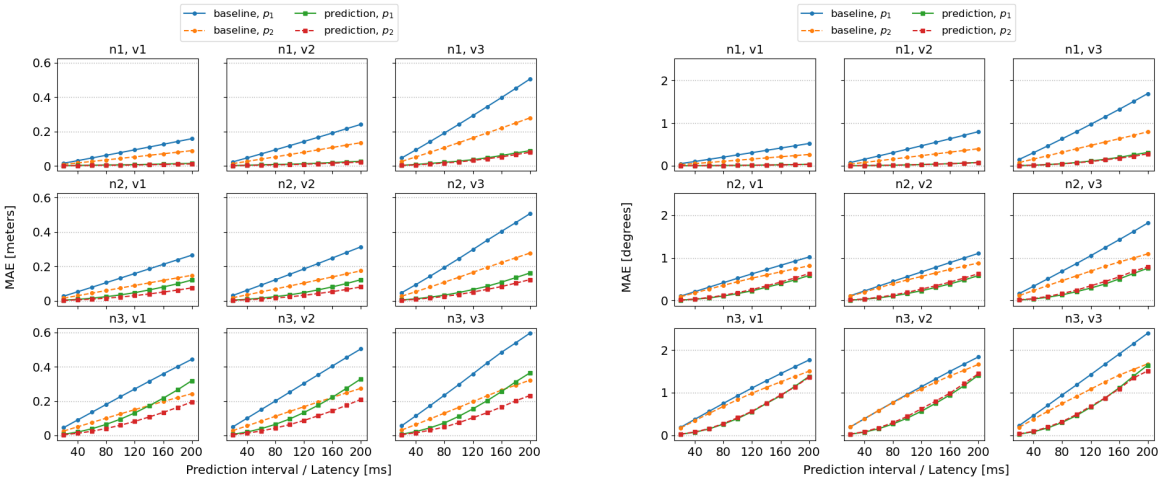
A prominent solution is to offload the complex rendering to a remote server that dynamically renders a 2D view from the volumetric video based on user's head pose [5]. The server then compresses the rendered texture into a 2D video stream and transmits it over a network to a client device. The client only needs to decode the video stream, perform a minimum amount of rendering and display it to the user. Thus, the amount of computing required on the client side is significantly reduced. Also, this approach enables usage of highly efficient video compression techniques, thereby reducing the network bandwidth requirements [6].

However, a significant drawback of remote rendering is the increased interaction (motion-to-photon) latency, which is mainly due to the network round-trip time [7]. Latency often causes the virtual objects to lag behind or swim around the intended position [8]. Several studies show that an increased interaction latency may lead to a reduced perceptual stability of objects and task performance in VR and AR environments [9, 10].

Head motion prediction approaches have been extensively studied in the 90s and 2000s in the context of predictive tracking to mitigate the dynamic registration errors of headset trackers [8, 11, 12] and also to reduce the effective latency (lag) between head motion and display response [12]. Tracking latency has largely been mitigated in the current generation of headsets. Mixed reality (MR) headsets such as Microsoft HoloLens are typically equipped with RGB cameras and inertial measurement units (IMU) that track the head pose and location within the physical world with millimeter-precision in a few milliseconds, using advanced simultaneous localization and mapping (SLAM) algorithms [13]. However, image rendering and display still takes a fair amount of time; for example, at 60 fps, it may take up to $\sim$16.66ms. Therefore, HoloLens predicts where the user's head will be when the image is shown on the display. Additionally, in a final processing stage called *late-stage reprojection* [14], HoloLens compensates the discrepancy between the predicted and actual head position by warping the rendered image by a small amount. More generally, such compensation approaches, where a rendering algorithm is followed by further processing steps to transform ("warp") the initial output, are known in computer graphics literature as post-rendering warping (PRW) [15, 16].

In remote rendering-based systems, an additional network latency has to be compensated in addition to the device latency, which is already handled by the above-mentioned techniques. Different prediction techniques such as autoregression models and Kalman filter have already been considered in the context of volumetric video streaming [17, 18]. However, previous theoretical analysis of head motion prediction shows that prediction errors grow rapidly with increasing prediction intervals and input signal frequencies [19]. Furthermore, although the previous approaches demonstrated a decrease in average tracking error, the authors reported a significant judder in the resulting images due to frequent mispredictions that occur with high spatial variance [18]. Therefore, adoption of PRW techniques is necessary for providing an accurate rendering result in remote rendering while performing latency compensation.

In our work, we propose a planar homography-based approach that warps the rendered (predicted) image according to the latest pose acquired on the client device (Sec. 2). The proposed approach is evaluated using artificial camera traces generated in Blender (Sec. 3.1) using a framework that simulates remote rendering-based volumetric video streaming (Sec. 3.2). Results are presented and discussed in Sec. 3.3 that provides an analysis of the proposed approach both with and without initial motion prediction.



(a) Positional error (Euclidean distance).

(b) Angular error (spherical distance).

Fig. 1: Average prediction errors for both camera paths for different camera velocity and noise over different prediction intervals.

2. HOMOGRAPHY-BASED IMAGE WARPING

The projective geometry between two views is defined by the epipolar geometry which is independent of the scene structure and only depends on the cameras' internal parameters and relative poses. In a more specific case, images of the points on a world plane are related to the corresponding image points in a second view by a (planar) homography. It is said that the plane *induces* a homography between the two views which is uniquely determined by the plane. In other words, the points are transferred from one view to the other using the homography map [20].

The following two cases are equivalent for homography computation: 1) two cameras are present in the scene simultaneously viewing a plane in 3D space, 2) a single camera is present that changes its position over time. Our simulated remote rendering framework pertains to the latter. The first camera pose is the (predicted) head pose of the user at the time of rendering a view from the volumetric video at the remote server. The second camera pose is the latest acquired pose (after network and processing delays) just before the rendered image is sent to the display of the end device. We use the displacement between the two camera poses to compute the homography. Our assumption is that all objects in the scene lie on a plane, i.e., we have *planar* objects; otherwise different camera views would not be related to each other by a homography.

Assuming that our camera is calibrated, we can write the 3×4 camera matrices for any two camera poses as

$$\mathbf{P}_1 = \mathbf{K}\left[\mathbf{R}_1 \mid \mathbf{t}_1\right], \quad \mathbf{P}_2 = \mathbf{K}\left[\mathbf{R}_2 \mid \mathbf{t}_2\right] \tag{1}$$

where $\mathbf{K} \in \mathbb{R}^{3x3}$ is the camera intrinsics matrix, $\mathbf{R}_1, \mathbf{R}_2 \in \mathbb{R}^{3x3}$ are the rotation matrices and $\mathbf{t}_1, \mathbf{t}_2 \in \mathbb{R}^{3x1}$ are the translation vectors corresponding to the first and second camera poses, respectively. Considering two images of points p on a 2D plane P in 3D space, the Euclidean homography matrix that transforms one 3D point from one camera frame to the other can be written as [21]

$$\mathbf{H} = \mathbf{R}_{12} - \frac{1}{d}\mathbf{t}_{12}\mathbf{n}^T \tag{2}$$

where $\mathbf{R}_{12} = \mathbf{R}_2\mathbf{R}_1^T$ is the rotation, $\mathbf{t}_{12} = -\mathbf{R}_2\mathbf{R}_1^T\mathbf{t}_1 + \mathbf{t}_2$ is the translation between the two camera frames, $\mathbf{n}$ is the unit normal vector of the plane P with respect to the first camera frame, and d is the distance from the plane P to the optical center of the first camera. Then, the projective homography matrix can be computed (up to a scale factor) as [21]

$$\mathbf{G} = \gamma\mathbf{K}\mathbf{H}\mathbf{K}^{-1}. \tag{3}$$

A perspective transformation[1] can then be applied to a given source image using $\mathbf{G}$ as the 3×3 transformation matrix.

3. EXPERIMENTS

3.1. Artifical camera traces

Although it is possible to retrieve the recorded head pose data from MR headsets such as HoloLens 2, it would be difficult to perform controlled experiments with real user traces since we cannot precisely control the users' motion paths, velocities and the amount of noise (jitter) in their movements. Therefore, we prefer to use artificial camera traces generated using a virtual camera in Blender.

Two camera paths $\{p_1, p_2\}$ are created where on p_1, the camera pans along the X axis, and on p_2, it approaches towards the object, along the Y axis[2]. For both paths, a user trace is generated by moving the camera in all permutations of the three velocity levels $\{v_1 < v_2 < v_3\}$ and three noise levels (camera shake) $\{n_1 < n_2 < n_3\}$ (n_1 means no camera shake), resulting in a total of 18 artificial user traces.

The velocity levels $\{v_1, v_2, v_3\}$, where $v_1 < v_2 < v_3$, are adjusted by modifying the number of animation frames in Blender such that the camera takes different amounts of time to move along the same path. Table 1 shows the used number of animation frames as well as the mean linear and angular speed measured from the generated traces. The speed values simulate movements ranging from slow to fast considering that humans typically walk at a preferred speed around 1.4m/s [22].

[1] In our implementation, we use the `cv2.warpPerspective` function from OpenCV.

[2] Blender uses a right-handed coordinate system with Z axis pointing upwards.

The noise levels $\{n_1, n_2, n_3\}$ are set by adding *noise modifiers* to the camera's trajectory and orientation in Blender. In detail, two different noise modifiers are added to the camera position, i.e. the camera's location along the predefined trajectory, and the camera rotation, i.e. the camera's *look-at* direction. The parameters of the noise modifiers *strength*, *scale*, and *depth* control the amplitude, dispersion/scale, and the amount of fine detail of the noise function, respectively [23]. The parameters are set to different values to generate the set of noise levels $\{n_1 < n_2 < n_3\}$ (Table 2). The values are adjusted by visual inspection to simulate realistic levels of camera shake.

Table 1: Speed parameters for the artificial user traces generated in Blender.

	v_1	v_2	v_3
# Animation frames	900	600	300
Mean linear speed [m/s]	0.75	1.13	2.27
Mean angular speed [deg/s]	4.99	7.49	14.98

Table 2: Noise parameters for the artificial user traces generated in Blender. The values are given as triplets of (strength, scale, depth) of the noise function controlled by the *noise modifiers* in Blender [23].

	n_1	n_2	n_3
Positional	(0,0,0)	(0.25, 50, 1.0)	(1.0, 20, 1.0)
Rotational	(0,0,0)	(0.5, 50, 2.0)	(2.0, 20, 1.0)

3.2. Simulation framework

After trace generation, predicted poses are obtained for the interval $\{20, 40, \ldots, 200\}$ms using a Kalman filter-based predictor proposed in [18]. Two different volumetric video sequences (*Josh* and *Mitch*) are imported[3] into Blender as OBJ sequences. The volumetric objects are placed onto a solid background color. There are no other elements (other 2D/3D objects, background image etc.) present in the scene.

At each time instant, the camera is moved to the predicted pose, and an image I_P is rendered and stored. For reference, an image I_R corresponding to the recorded pose is also rendered and stored. The camera intrinsics matrix $\mathbf{K}$ can be directly obtained from Blender, and the distance d between the object plane and the camera can be calculated from the 3D scene, since the volumetric object is placed at a pre-defined position. Finally, the homography between two camera poses is computed using (3) and I_P is warped to obtain the image I_W, which is stored for evaluation.

3.3. Results and discussion

In the following, we evaluate the performance of the Kalman filter-based predictor on the artificial user traces and the homography-based warping on the images rendered in Blender. As reference, we define a *baseline* case where no prediction is performed, i.e. the rendered image is outdated by a given interaction latency. For the prediction case, we assume that the interaction latency is constant and known, thus the prediction interval can be set equal to it.

[3]We use the Blender add-on: `https://github.com/neverhood311/Stop-motion-OBJ`

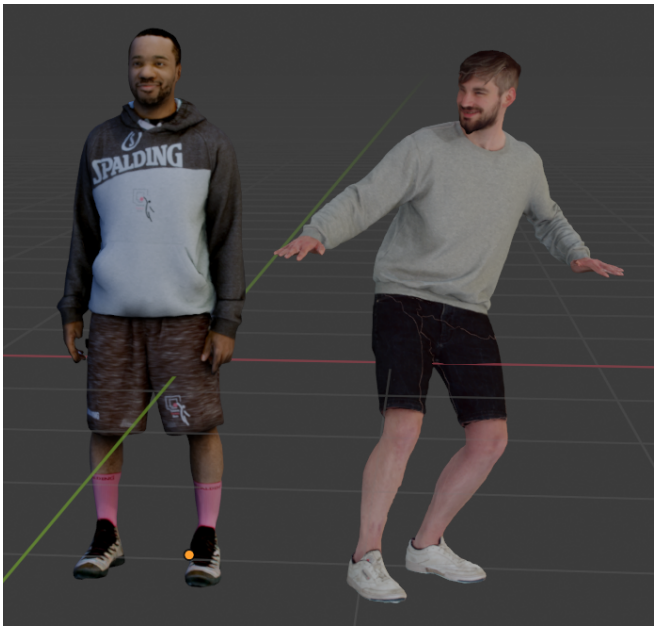


Fig. 2: The used volumetric sequences: left: Josh, right: Mitch

3.3.1. Prediction performance

The predictor's performance is evaluated for the different user traces i.e. different (camera path, noise, velocity) triplets. As in [18], positional and angular errors are computed for each time point, and the mean absolute error (MAE) is calculated over each user trace, for different prediction intervals. Fig. 1 shows the MAE of the Euclidean distance and spherical distance, respectively, between the predicted and reference samples for each trace. As a benchmark, the values for the baseline (no-prediction) case are also provided. We observe that both positional and angular error become larger with increasing prediction intervals (interaction latency, for the baseline case). Also, increasing camera velocity and camera shake negatively affect the prediction accuracy. Comparison of the predefined camera paths show that both for the baseline and prediction cases, the average error for p_1 is higher than p_2. Thus, we can conclude that p_1 (horizontal panning) is more challenging for the prediction algorithm than p_2 (camera approaching the object).

3.3.2. Evaluation of warping

In Fig. 3, we compare the MSE between the warped images (I_W) and reference images (I_R) to the MSE between the predicted (I_P) and reference images (I_R). We perform the same analysis for the baseline too, where in that case $I_{P'}$ is the image rendered without prediction and displayed at the client after a lag equal to the interaction latency. We observe the following: i) Longer prediction window results in higher MSE (as expected from the results in Fig. 1, ii) Both higher camera velocity and higher camera shake result in greater MSE. For the tested set of noise and velocity levels, noise has a greater impact, iii) Without warping, predicted images have a lower MSE than the baseline. Warping decreases MSE further in both cases. First prediction, then warping provides the lowest MSE. Inspecting the MSE differences between the two cases where warping is applied directly to the baseline (*base-warped*) and after an initial prediction step (*pred-warped*), we observe that *pred-warped* achieves similar MSE to *base-warped* at lower latency values. For

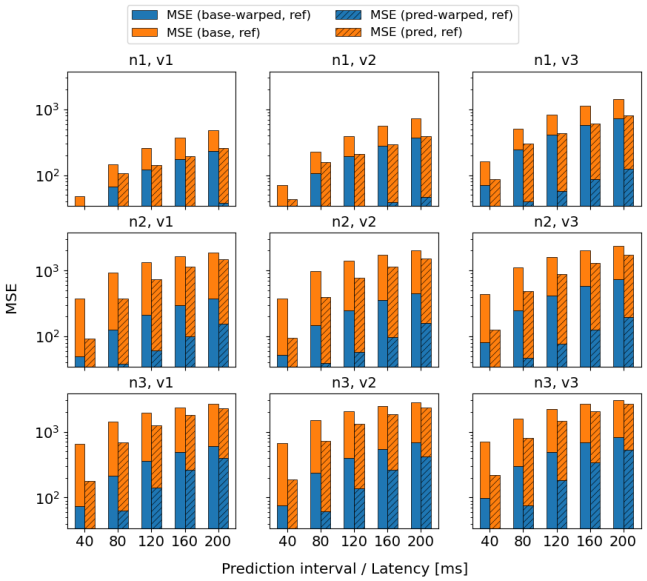


Fig. 3: Comparison of MSE between the images rendered from predicted head pose with and without homography-based warping (shown for *Josh* and the camera path p_1).

example, for the pair (n_2, v_2), the MSE values in both cases are almost equal (198.45 at pw=80ms for *base-warped*, and 199.34 at pw=200ms for *pred-warped*, respectively). This means that initial prediction increases the affordable latency budget of the system (in the example by 120ms), and thus can help providing a smooth user experience at high-latency environments.

Next, we analyze the effect of camera path on the warping performance. For this, we fix a single noise-velocity pair (n_2, v_2) and analyze the performance for the two camera paths p_1 and p_2. The results are shown in Table 3. We observe that the MSE after warping is slightly better for p_2 both for baseline and prediction cases. From this, we can conclude that dynamics of motion along p_2 is less challenging than p_1, under the same velocity and noise conditions.

Table 3: MSE for different camera paths p_1 and p_2 for the noise-velocity pair (n_2, v_2) and object *Josh* over different prediction windows (pw).

pw	Path	Base	Base-Warped	Pred	Pred-Warped
40	p_1	611.08	96.12	189.57	34.22
	p_2	566.90	49.51	155.58	27.2
80	p_1	1008.03	198.45	527.92	47.15
	p_2	996.36	89.13	455.56	40.15
120	p_1	1277.67	299.93	890.13	73.39
	p_2	1298.32	132.14	840.29	62.84
160	p_1	1485.73	402.61	1210.13	127.38
	p_2	1556.40	173.01	1218.06	99.05
200	p_1	1637.09	499.30	1493.59	199.34
	p_2	1774.71	212.19	1603.89	155.08

Lastly, we analyze the effect of different and multiple objects on the warping performance. For this, we fix a single trace (p_1, n_2, v_2) and analyze the different cases where either only a single object (*Josh* or *Mitch*) is present, or two objects are co-located in the scene. In the latter case, *Mitch* is placed slightly in front of *Josh* (closer to the camera), such that *Josh* is only partially visible at the beginning of the sequence and is gradually disoccluded as the camera pans from left to right. The results are shown in Table 4. We observe that there is little variation of MSE for the two different objects; this is expected because the homography approach is independent of the scene content. The small differences between the MSE results is explained by the different number of object pixels and different dynamics of the video sequences. For the case of two objects, MSE is significantly higher since there are more object pixels (relative to the solid background) in the image than the single object case, increasing the error probability.

Table 4: MSE for different object configurations with the trace (p_1, n_2, v_2), pw=100ms.

	Base	Base-Warped	Pred	Pred-Warped
Josh	1154.89	250.72	714.18	58.18
Mitch	1235.75	166.21	790.59	62.63
Both	2731.17	1100.81	1559.45	255.88

It is expected that warping can be problematic when a view contains insufficient information, i.e. the new view looks into missing regions. Warps of such images tend to create holes in the output images as objects become disoccluded [15]. However, it is hard to detect this issue in our videos, possibly because our scene solely contains single volumetric objects and not a full image frame, e.g. as in the case of 360-degree video streaming. On the other hand, we observe problems in edge regions where the information for the new view is not available and the pixels in the warped image have to be simply duplicated. However, this problem can easily be solved by warping an "overscanned" version of the image, i.e. an image larger than what will be displayed to the user, and cropping the redundant parts after warping, as shown in Fig. 4.

Fig. 4: Effect of overscanning. The image without overscan (left) has duplicated pixels at the top border. This is mitigated by rendering a 25% larger area and cropping the image to the desired size after prediction and warping (middle). The reference image is shown for comparison (right).

4. CONCLUSION

Increased interaction latency is a significant problem for remote rendering-based volumetric video streaming applications. Head motion prediction at the remote server is one of the potential solutions proposed to mitigate the latency. However, although accurate prediction may reduce the average error, it fails to provide a stable image due to jitter/mispredictions, degrading the user experience especially at higher prediction intervals as well as higher camera velocity and

noise levels (camera shake). We propose a homography-based image warping technique that uses the latest head pose on the client side to mitigate the prediction errors and stabilize the video. Our results show that the proposed technique can also be applied without initial prediction, with minimal loss of rendering accuracy.

5. ACKNOWLEDGMENTS

This research has received funding from the German Federal Ministry of Education and Research as part of the VoluProf project under Grant 16SV8705.

6. REFERENCES

[1] Néill Odwyer, Emin Zerman, Gareth W Young, Aljosa Smolic, Siobhán Dunne, and Helen Shenton, "Volumetric video in augmented reality applications for museological narratives: A user study for the Long Room in the library of Trinity College Dublin," *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 14, no. 2, pp. 1–20, 2021.

[2] Markus Worchel, Marcus Zepp, Weiwen Hu, Oliver Schreer, Ingo Feldmann, and Peter Eisert, "Ernst Grube: A contemporary witness and his memories preserved with volumetric video," in *Proc. Eurographics Workshop on Graphics and Cultural Heritage (GCH2020)*, Granada, Spain, Nov. 2020.

[3] Sylvie Dijkstra-Soudarissanane, Tessa Klunder, Aschwin Brandt, and Omar Niamut, "Towards XR communication for visiting elderly at nursing homes," in *ACM International Conference on Interactive Media Experiences*, 2021, pp. 319–321.

[4] Sebastian Schwarz and Mika Pesonen, "Real-time decoding and AR playback of the emerging MPEG video-based point cloud compression standard," *Nokia Technologies; IBC: Helsinki, Finland*, 2019.

[5] Shu Shi and Cheng-Hsin Hsu, "A survey of interactive remote rendering systems," *ACM Computing Surveys*, vol. 47, no. 4, pp. 1–29, May 2015.

[6] Feng Qian, Bo Han, Jarrell Pair, and Vijay Gopalakrishnan, "Toward practical volumetric video streaming on commodity smartphones," in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. ACM, 2019, pp. 135–140.

[7] Zhehui Zhang, Shu Shi, Varun Gupta, and Rittwik Jana, "Analysis of cellular network latency for edge-based remote rendering streaming applications," in *Proceedings of the ACM SIGCOMM 2019 Workshop on Networking for Emerging Applications and Technologies*, 2019, pp. 8–14.

[8] Ronald Tadao Azuma, *Predictive tracking for augmented reality*, Ph.D. thesis, University of North Carolina at Chapel Hill, 1995.

[9] Robert S Allison, Laurence R Harris, Michael Jenkin, Urszula Jasiobedzka, and James E Zacher, "Tolerance of temporal delay in virtual environments," in *Proceedings IEEE Virtual Reality 2001*. IEEE, 2001, pp. 247–254.

[10] Mark A Livingston and Zhuming Ai, "The effect of registration error on tracking distant augmented objects," in *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE, 2008, pp. 77–86.

[11] Joseph J LaViola, "A comparison of unscented and extended kalman filtering for estimating quaternion motion," in *Proceedings of the 2003 American Control Conference, 2003.* IEEE, 2003, vol. 3, pp. 2435–2440.

[12] Edgar Kraft, "A quaternion-based unscented kalman filter for orientation tracking," in *Proceedings of the sixth international conference of information fusion*, July 2003, pp. 47–54.

[13] Dorin Ungureanu, Federica Bogo, Silvano Galliani, Pooja Sama, Xin Duan, Casey Meekhof, Jan Stühmer, Thomas J Cashman, Bugra Tekin, Johannes L Schönberger, et al., "Hololens 2 research mode as a tool for computer vision research," *arXiv preprint arXiv:2008.11239*, 2020.

[14] Microsoft, "Mixed Reality documentation: Hologram stability," `https://docs.microsoft.com/en-us/windows/mixed-reality/develop/advanced-concepts/hologram-stability`, 2018, Online; accessed: 2022-02-12.

[15] William R Mark, Leonard McMillan, and Gary Bishop, "Post-rendering 3D warping," in *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, 1997, pp. 7–ff.

[16] Matthew Regan and Ronald Pose, "Priority rendering with a virtual reality address recalculation pipeline," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994, pp. 155–162.

[17] Serhan Gül, Dimitri Podborski, Thomas Buchholz, Thomas Schierl, and Cornelius Hellge, "Low-latency cloud-based volumetric video streaming using head motion prediction," in *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2020, pp. 27–33.

[18] Serhan Gül, Sebastian Bosse, Dimitri Podborski, Thomas Schierl, and Cornelius Hellge, "Kalman filter-based head motion prediction for cloud-based mixed reality," in *Proceedings of the 28th ACM International Conference on Multimedia*, New York, NY, USA, 2020, MM '20, p. 36323641.

[19] Ronald Azuma and Gary Bishop, "A frequency-domain analysis of head-motion prediction," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 401–408.

[20] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[21] Ezio Malis and Manuel Vargas, *Deeper understanding of the homography decomposition for vision-based control*, Ph.D. thesis, INRIA, 2007.

[22] Betty J Mohler, William B Thompson, Sarah H Creem-Regehr, Herbert L Pick, and William H Warren, "Visual flow influences gait transition speed and preferred walking speed," *Experimental brain research*, vol. 181, no. 2, pp. 221–228, 2007.

[23] Blender, "F-curve modifiers," `https://docs.blender.org/manual/en/latest/editors/graph_editor/fcurves/modifiers.html`, 2022, Online; accessed: 2022-02-21.