User-Perspective Rendering for Handheld Applications

User-Perspective Rendering for Handheld Applications

Jing Yang* Shiheng Wang* Gábor Sörös*

Department of Computer Science
ETH Zurich, Switzerland

ABSTRACT

A problem of most handheld applications is that the digital content is not rendered from the user's viewpoint, but rather from an arbitrary perspective. In this work, we propose an easy-to-use user-perspective rendering algorithm which can be applied to general handheld mobile devices. We present a smartphone App that can enhance the sense of presence when perceiving the virtual objects, by adjusting the 2D projections, lighting conditions, and spatial sound effects according to the user's viewpoint in real time.

Keywords: User-perspective rendering, spatial sound, multimodal output, handheld applications.

Index Terms: Human-centered computing—Sound-based input/output; Computing methodologies—Rendering

1 INTRODUCTION

Handheld applications are very popular in entertainment, maintenance, and educational contexts. In a typical scenario, the user holds a mobile device (smartphone or tablet) to view virtual contents, which are usually rendered from an arbitrary perspective (e.g., the virtual player's viewpoint in mobile games), or from the rear camera of the device (e.g., in AR applications), then displayed on the device screen. An apparent problem in the above scenario is that the virtual content is not created from the user's viewpoint. This results in a lack of immersion, i.e., the feeling of holding a 2D display rather than truly perceiving a 3D object.

In this work, we propose a user-perspective rendering approach to overcome this problem by adapting both the camera view and the rendered content to match the user's viewpoint, which is achieved by tracking the user's face and eyes with respect to the handheld device using its front-facing camera. Given such a user-perspective rendering system, more visual and audio cues can be added to enhance the perception of virtual objects.

Our system consists of three modules, namely, *front camera-screen calibration*, *viewpoint estimation*, and *rendering*, in which the *rendering* includes *2D projection*, *lighting configuration*, and *spatial sound creation*. Our main contributions are twofold:

- While most user-perspective rendering methods are implemented on platforms with specific hardware like depth camera, head tracking sensor, etc., we develop an easy-to-use user-friendly algorithm which can be applied to general commercial mobile devices with simple RGB front camera.
- On top of our algorithm, we develop a mobile App that shows the rendered scene with varying 2D shapes, lighting conditions, and spatial sound effects based on the user's viewpoint in real time, which provides the user with more immersive experience when perceiving virtual objects.

*{yangj|wangshi|soeroesg}@ethz.ch

2 RELATED WORK

In this section, we overview the literature related to each module of our approach.

2.1 Front Camera-Screen Calibration

The external calibration of a mobile device aims at finding the relative pose between the front camera and the screen, which is rather important since all the following geometric poses are calculated based on it. However, this step has not been well studied in some previous user-perspective rendering algorithms [1, 3, 4, 6, 14, 16, 18, 22, 23]. They either use specifically designed systems of which the screen-camera poses cannot be generalized out of their project contexts [1, 3, 14, 16, 18, 22, 23], or the poses are manually measured for their mobile devices [4, 6].

To develop a more general calibration approach, Baričević et al. [2] propose a calibration algorithm using the mirror-based method developed by Matsuyama et al. [15]. Though applicable on many devices, this calibration method requires measuring the device geometry in advance. Schöps et al. [19] calibrate screen-camera poses using a marker on the device screen and a marker in the environment. This approach is robust and easy, but it needs the markers and an external camera.

In this work, we intend to develop an easy-to-use calibration approach which is supposed to rely on the external environment as little as possible with minimal setup and extra hardware requirements. The goal is to calibrate the relative pose from the front camera to the screen, which is not directly visible for the front camera. Inspired by [5], we propose a mirror-based calibration method with a ChArUco[1] marker displayed on the device screen.

2.2 Viewpoint Estimation

In order to render virtual objects from the user's perspective, the system is supposed to track the user's viewpoint relative to the display. Compared with other application situations, viewpoint estimation in user-perspective rendering is special in a sense that a user keeps the gaze direction consistent with the head orientation in most cases, which means that the viewpoint can be approximately represented by the user's head or face orientation.

To this end, some previous works [14, 19] use predefined or simulated head pose. Some systems [3, 18] track the user's head pose with head-mounted sensors and stream the tracking results to the handheld devices. In our work, we propose a method based on a more advanced facial landmark detection algorithm [9] to achieve more accurate head pose and viewpoint estimation. Plus, we add a primary user face tracker to deal with the drawback that the classifier always tries to detect all faces appearing in the image while only one of them belongs to the primary user. Like in the works [2, 16, 22], we also track the user's face using RGB front camera.

2.3 User-Perspective Rendering

There exist many works on user-perspective rendering [2, 3, 6, 11, 16, 18–21] for AR or entertainment applications. All of them only focus on the visualization which renders correct geometric shapes

[1] https://docs.opencv.org/3.4.1/df/d4a/tutorial_charuco_detection.html

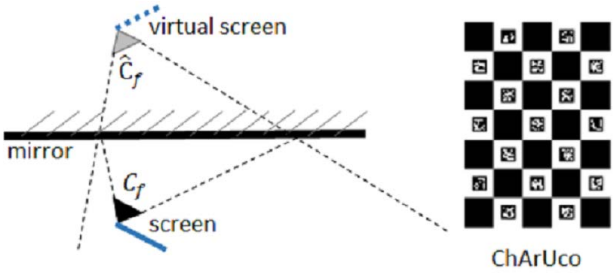

Figure 1: Top view of a smartphone facing a mirror with an arbitrary angle. C_f and $\widehat{C}_f$ are the front camera and the virtual front camera behind the mirror. The dash-lines mark the field of view of the front camera with reflection. On the screen a ChArUco marker is displayed.

according to the user's viewpoints. In our work, we also include user-perspective lighting configuration and spatial sound creation to further improve the sense of presence.

3 Our Approach

To realize user-perspective rendering, we estimate the user's viewpoint relative to the front camera, and register it in the screen coordinate system with the pre-calibrated pose between the front camera and the screen of the device. Then, based on the user's viewpoint, we re-project the virtual object as well as model the corresponding lighting and spatial sound effect. As described in previous sections, the whole system consists of three modules, which will be elaborated in the following sections.

3.1 Front Camera-Screen Calibration

We calibrate the pose between the front camera and the screen using a mirror. As illustrated in Figure 1, we display a ChArUco marker on the device screen, and place a planar mirror towards the front camera, which can then capture the virtual screen in the mirror. As demonstrated in the figure, the screen does not have to be parallel to the mirror. However, we do assume that the screen is planar.

With T_f representing the relative pose between the real screen and the real front camera, S_i representing the mirror reflection matrix, and V_i representing the virtual front camera pose, we have the following equation

$$V_i = T_f S_i \tag{1}$$

Our target is to recover T_f and S_i from V_i. This problem is known in the literature as Images of Planar Mirror Reflections (IPMR). To solve this problem and estimate the real front camera pose, we utilize the approach described in [17]. Referring to the work [5], we also implement a Bundle Adjustment (BA) method to optimize the calibration result.

The calibration model is calculated on a PC offline, and then loaded back to the mobile device for further rendering steps.

3.2 Viewpoint Estimation

We estimate the user's viewpoint in real time using the following steps.

We first detect the user's face using the local binary patterns (LBP) cascade classifier. This face detector can find all faces in the frame but we only focus on the primary user. Therefore, we introduce a primary face tracker to exclude all the other random faces which happen to be in the frame. We assume that in most cases, the primary user is the closest to the screen since the user views the screen while holding it in hands. Therefore, at first, all detected faces are evaluated based on their sizes, then we initialize the largest face (the closest one) as the primary user, which will be tracked in the following frame based on the previous location and appearance. In our work, we employ a state-of-the-art kernelized correlation filters (KCF) tracker as described in [7] to track the primary user's face. This tracker is fast and can also deal with changes of the tracking target over time (e.g., when the face rotates in 3D).

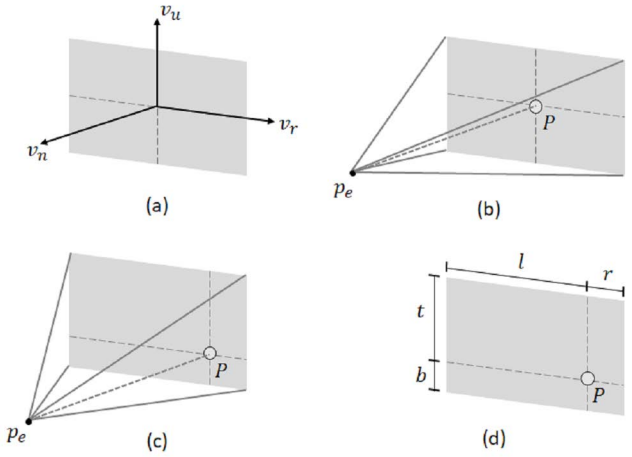

Figure 2: From top-left to bottom-right: (a) the coordinate frame of the screen space, (b) an on-axis perspective projection, (c) an off-axis perspective projection, and (d) the left l, right r, bottom b, and top t extents of the perspective projection [10].

Based on the previous two steps, we detect and track facial landmarks in each frame. We utilize one of the state-of-the-art methods, ERT Face Alignment [9], to extract 2D facial lndmarks, which are then tracked using the Lucas-Kanade optical flow method [13].

So far, we can track the primary user's face and extract stable facial landmarks, which are then utilized to calculate the head pose and approximate the viewpoint. We solve the pose by the means of Perspective-n-Point (PnP) approach using the 2D facial landmarks and their corresponding 3D points on a face model, which is a lightweight 3D mean face model from a 3D morphable model (3DMM) [8]. This face model can represent general face properties of human beings. Considering the accuracy and efficiency on mobile devices, we utilize the efficient PnP (EPnP) algorithm [12]. To accommodate the head sizes of different users, during the implementation, we first scale the 3D mean face model according to the user's interpupillary distance (IPD). Once the head pose is calculated, the viewpoint is registered to the screen coordinate system. From the 3D face model, we can extract the position of the eye P_l^E in its local face coordinate system. The approximate viewpoint relative to the screen P_s^E is estimated by transferring the eye position to the screen coordinate system by $P_s^E = T_s^H P_h^E$, where T_s^H is the head pose relative to the screen and P_h^E is the eye pose relative to the head.

If the facial landmark tracking fails, the viewpoint will be estimated by a temporal smoothing filter, which calculates an average pose of the latest five frames.

3.3 Rendering

Our rendering module includes three parts: 2D projection, lighting configuration, and spatial sound creation.

3.3.1 2D Projection

As elaborated in [10], we employ off-axis projections to convert 3D models into 2D images according to the user's viewpoint. To understand off-axis projection, one should first understand the concept of view frustum, which is a pyramid that represents the region of visual space. As demonstrated in Figure 2, the shaded plane represents the

projection plane (the screen). P is the projection of the viewpoint p_e on the plane. In Figure 2(b), the viewpoint projection is on the center of the screen, which represents the on-axis perspective projection. Figure 2(c) demonstrates an off-axis perspective projection, of which the viewpoint falls off the screen center.

Given the viewpoint coordinate on the screen space, the left, right, bottom, and top extents (l,r,b,t) of the perspective projection can be easily computed. According to [10], based on the plane extents (l,r,b,t), the screen space basis (v_x,v_y,v_z), the viewpoint position p_e, and the nearest and the farthest visible distance (n,f) of the view frustum, we can first calculate a standard projection matrix P which refers to the view position on the original screen space, then calculate a linear transformation matrix M to rotate the screen orientation, and finally calculate a transformation matrix T to correct the viewpoint offset. Based on P, M, and T, a generalized perspective projection matrix P' can be calculated using $P' = PM^T T$. With this perspective projection matrix P', the 2D projection according to the user's viewpoint can be calculated.

3.3.2 Lighting Configuration

To create a realistic viewing experience, we write shader scripts to configure the ambient light, the diffuse light, and the specular light in the virtual environment. While the ambient light is rendered at the background, the diffuse light and the specular light are dynamically placed at the position of the user's head. This creates the effect of wearing a head lamp when inspecting the object. When the user approaches the screen from different directions and distances, the part of the virtual object which is the closest to the user is lit most, while the farther components gradually become dim. This design helps to enhance the sense of exploration when the user views the virtual objects.

3.3.3 Spatial Sound Creation

In our work, spatial sound effects are created by having the sound source and the listener at different positions with a relative orientation. To accommodate the head poses from varying angles and distances to the screen, we localize the sound source at the centroid of the virtual object, and dynamically register the listener's position at the user's head position. When the user moves the head around, the listener's position is updated so to experience different spatial sound effects, which are simulated based on the head-related transfer functions (HRTFs) as implemented in the SDK we use.

4 Implementation & Evaluation

In this section, we first introduce our hardware and software configurations, then we evaluate the performance of our user-perspective rendering system on a smartphone.

4.1 Hardware & Software

The mobile device we use is a Google Nexus 6 smartphone. It is equipped with a Qualcomm 2.7GHz CPU, an Adreno 420 GPU, and a 3GB RAM. The screen size is $12.7cm \times 7.4cm$ and the display resolution is 2560×1440 pixels. The Nexus 6 has two high-quality cameras which support various resolutions at a high speed of 30fps. To maximize the interaction region, we utilize the landscape layout as the default posture. With this posture, the translation from the front camera to the screen center is approximately $(7.1cm, 2.6cm, 0.0cm)$. We calculate the camera-screen calibration model offline on a MacBook Pro, which is equipped with an Intel Iris Pro 1536MB GPU, an Intel Core i7 2.2GHZ CPU, and a 16GB RAM. A pair of ordinary earphones is needed to experience the spatial sound.

The system is developed under the Android environment. The OpenCV library[2] with extra modules is utilized for most algorithms in the viewpoint estimation section. We render the 2D and 3D vector graphics with OpenGL ES 3[3] on the smartphone. The Google Resonance Audio SDK[4] is utilized to generate the spatial sound.

[2] https://github.com/opencv/opencv

Table 1: Front Camera-Screen Pose

	Translation	Rotation
Ground truth	$(7.1cm, 2.6cm, 0.0cm)$	$(0°, 0°, 0°)$
Calculated results	$(6.931cm, 2.461cm, -0.373cm)$	$(0.17°, 0.84°, -0.77°)$

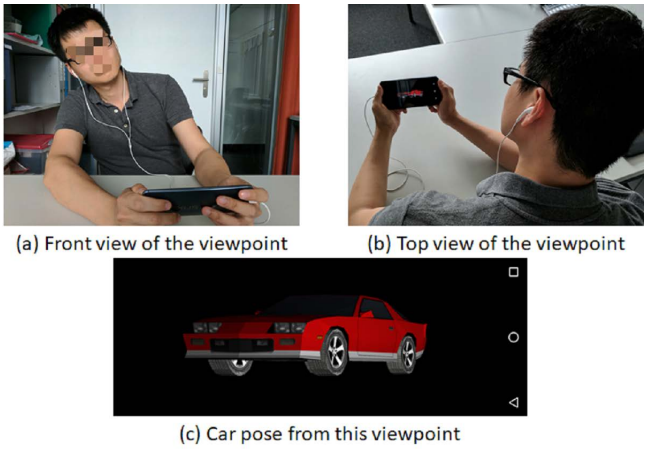

Figure 3: A user is viewing the car model from the right.

4.2 Evaluation

4.2.1 Front Camera-Screen Calibration

We set up the experiment environment according to the depiction in Section 3.1. The mirror is set static. The smartphone is held facing the mirror to capture the virtual screen at different positions with arbitrary angles. Each time, we randomly select three pictures to calculate the calibration model. The computation time is mainly influenced by the Bundle Adjustment optimization process which is dependent on the input pictures. The entire calculation takes approximately $3 \sim 10$ seconds.

To evaluate our calibration algorithm, we first manually measured the geometry of the smartphone as the ground truth, which was compared with the calculated results from our algorithm. Table 1 shows the comparison of the smallest error. Considering that the screen size is $12.7cm \times 7.4cm$, our algorithm correlates with the ground truth with a translation error of 1.3% in length and 1.9% in width. This result is comparable with the work of [5], in which the error is 0.9% in length and 1.5% in width. This calibration model is utilized in our mobile App.

4.2.2 Viewpoint Estimation

We implement the viewpoint estimation on Nexus 6 following the steps in Section 3.2. To evaluate the real-time performance of our system, we conduct experiments and calculate the mean fps every 100 frames. When the user moves slightly, an average rate of 30fps can be achieved. A fast move will influence the utilized optical flow method, which results in a lower rate of 23fps approximately.

4.2.3 Rendering

We implement the whole system as a 3D model reviewer App on Nexus 6. In this App, we provide four virtual models: box, teapot, car, and spaceship. The front camera has a field of view (FoV) of around 60° horizontally and 40° vertically. As long as the user

[3] https://www.khronos.org/opengles/
[4] https://developers.google.com/resonance-audio/

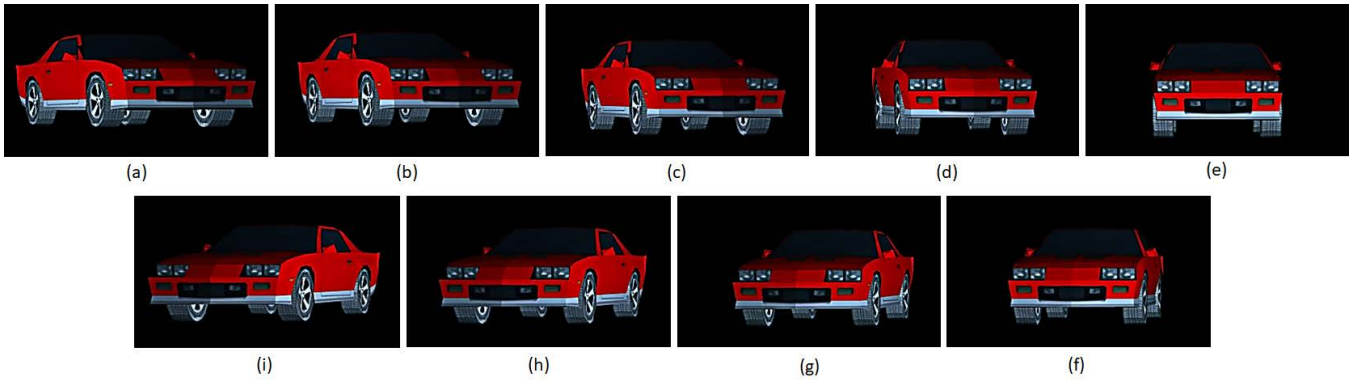

Figure 4: The rendered virtual models from different viewpoints when the user moves around the screen. Please refer to the demonstration video.

views the virtual object within the FoV of the camera, the rendered 2D image, lighting condition, and the audio experience will change according to the user's viewpoint in real time. Figure 3 shows an example of the user viewing the car model from the right.

Figure 4 demonstrates the rendered results from different viewpoints and further results are in the accompanying video[5]. It can be seen that even if only adjusting the viewpoint by a small angle, the re-projection varies significantly in terms of the geometric shape and the lighting. This car model is attached with a sound clip of an engine. When perceiving the sound using the earphones, the feeling corresponds to the expectation from the specific viewpoint[6].

A study involving 5 people (2 female and 3 male, average age of 26.6) was conducted to evaluate the user's experience. They reported that the system could run smoothly when they moved at a normal speed. When they moved fast or around the border of the front camera's FoV, a slight sense of lagging could be experienced at a few frames, but the overall viewing experience was smooth in real time and the rendered objects met their expectation of the geometrical shapes. The lighting and spatial sound effects were perceived as anticipated, which enhanced the sense of realism.

5 Conclusion & Future Work

In this work, we developed a low-cost user-perspective rendering system on general mobile devices with simple RGB front camera. The system is implemented as a mobile App on Nexus 6. This App improves the immersive experience when perceiving virtual objects on handheld devices, which is achieved by real-time viewpoint dependent 2D projection, lighting configuration, and spatial sound creation.

One limitation is that our rendering is not stereo for two eyes. Although the viewpoint is estimated for both eyes, the rendering is only based on the dominant eye, which is the right eye in our case. Therefore, like any other similar system, it only fits for one eye. Another restriction is the position of the front camera when we define the landscape layout as the default viewing posture. In this case, the front camera is located at the side, which results in an asymmetrical range of interaction with respect to the center of the screen.

There are several interesting directions of exploration in the future. First, we can also transfer the calibration computation to the mobile end. Another possible future work is to extend the user-perspective rendering to handheld AR applications, which not only requires the front and the rear cameras at the same time, but also involves the challenges with estimating the 3D world geometry on the fly.

[5] https://youtu.be/9AsvQJYbzpk

[6] There is a demonstration session at the workshop.

Acknowledgments

We thank Prof. Friedemann Mattern and Dr. Martin R. Oswald for their insightful discussions and help.

References

[1] D. Andersen, V. Popescu, C. Lin, M. E. Cabrera, A. Shanghavi, and J. Wachs. A hand-held, self-contained simulated transparent display. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pp. 96–101. IEEE, 2016.

[2] D. Baričević, T. Höllerer, P. Sen, and M. Turk. User-perspective AR magic lens from gradient-based IBR and semi-dense stereo. *IEEE transactions on visualization and computer graphics*, 23(7):1838–1851, 2017.

[3] D. Baričević, C. Lee, M. Turk, T. Hollerer, and D. A. Bowman. A hand-held AR magic lens with user-perspective rendering. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 197–206. IEEE, 2012.

[4] K. Čopič Pucihar, P. Coulton, and J. Alexander. Evaluating dual-view perceptual issues in handheld augmented reality: device vs. user perspective rendering. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pp. 381–388. ACM, 2013.

[5] A. Delaunoy, J. Li, B. Jacquet, and M. Pollefeys. Two cameras and a screen: How to calibrate mobile devices? In *2014 2nd International Conference on 3D Vision (3DV)*, vol. 1, pp. 123–130. IEEE, 2014.

[6] J. Francone and L. Nigay. Using the user's point of view for interaction on mobile devices. In *Proceedings of the 23rd Conference on l'Interaction Homme-Machine*, p. 4. ACM, 2011.

[7] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.

[8] P. Huber, G. Hu, R. Tena, P. Mortazavian, P. Koppen, W. J. Christmas, M. Ratsch, and J. Kittler. A multiresolution 3D morphable face model and fitting framework. In *Proceedings of the 11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, 2016.

[9] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1867–1874. IEEE, 2014.

[10] R. Kooima. Generalized perspective projection. *J. Sch. Electron. Eng. Comput. Sci*, 2009.

[11] J. C. Lee. Hacking the Nintendo WII remote. *IEEE pervasive computing*, (3):39–45, 2008.

[12] V. Lepetit, F. Moreno-Noguer, and P. Fua. EP*n*P: An accurate $O(n)$ solution to the P*n*P problem. *International journal of computer vision*, 81(2):155, 2009.

[13] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th In-*

ternational Joint Conference on Artificial Intelligence, pp. 674–679, 1981.

[14] Y. Matsuda, F. Shibata, A. Kimura, and H. Tamura. Poster: Creating a user-specific perspective view for mobile mixed reality systems on smartphones. In *2013 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 157–158. IEEE, 2013.

[15] T. Matsuyama, S. Nobuhara, and K. Takahashi. A new mirror-based extrinsic camera calibration using an orthogonality constraint. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1051–1058. IEEE, 2012.

[16] P. Mohr, M. Tatzgern, J. Grubert, D. Schmalstieg, and D. Kalkofen. Adaptive user perspective rendering for handheld augmented reality. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 176–181. IEEE, 2017.

[17] R. Rodrigues, J. P. Barreto, and U. Nunes. Camera pose estimation using images of planar mirror reflections. In *European Conference on Computer Vision*, pp. 382–395. Springer, 2010.

[18] A. Samini and K. L. Palmerius. A perspective geometry approach to user-perspective rendering in hand-held video see-through augmented reality. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, pp. 207–208. ACM, 2014.

[19] T. Schöps, M. R. Oswald, P. Speciale, S. Yang, and M. Pollefeys. Real-time view correction for mobile devices. *IEEE transactions on visualization and computer graphics*, 23(11):2455–2462, 2017.

[20] M. Spindler, W. Büschel, and R. Dachselt. Use your head: tangible windows for 3d information spaces in a tabletop environment. In *Proceedings of the 2012 ACM international conference on Interactive tabletops and surfaces*, pp. 245–254. ACM, 2012.

[21] I. Stavness, B. Lam, and S. Fels. pCubee: a perspective-corrected handheld cubic display. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1381–1390. ACM, 2010.

[22] M. Tomioka, S. Ikeda, and K. Sato. Approximated user-perspective rendering in tablet-based augmented reality. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 21–28. IEEE, 2013.

[23] Y. Unuma and T. Komuro. [poster] natural 3d interaction using a see-through mobile AR system. In *2015 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 84–87. IEEE, 2015.