A Perspective Geometry Approach to User-Perspective Rendering in Hand-Held Video See-Through Augmented Reality

Ali Samini* Karljohan Lundin Palmerius†

Department of Science and Technology, Linköping University, Sweden

Abstract

Video see-through Augmented Reality (V-AR) displays a video feed overlaid with information, co-registered with the displayed objects. In this paper we consider the type of V-AR that is based on a hand-held device with a fixed camera. In most of the VA-R applications the view displayed on the screen is completely determined by the orientation of the camera, i.e., the *device-perspective rendering*; the screen displays what the camera sees. The alternative method is to use the relative pose of the user's view and the camera, i.e., the *user-perspective rendering*. In this paper we present an approach to the user perspective V-AR using 3D projective geometry. The view is adjusted to the user's perspective and rendered on the screen, making it an augmented window. We created and tested a running prototype based on our method.

CR Categories: H.5.1 [Information Interfaces and Presentation]: Multimedia Information systems—Artificial, augmented, and virtual realities; I.3.3 [Computer Graphics]: Methodology and Techniques—Picture/Image Generation;

Keywords: augmented reality, video see-through, dynamic frustum, user-perspective

1 Introduction

The commonly used approach for hand-held *Video See-through Augmented Reality* is to create the view on the screen completely based on the view of the camera that captures the world, *device-perspective rendering* (DPR). This creates a misalignment between the real view, around the screen, and the augmented video feed on the screen. An alternative approach is to create a dynamic view and graphics projection that is based on the relative transformation of the user's view and the screen *user-perspective rendering* (UPR). The current approaches towards hand-held V-AR with UPR are based on homography and therefore they have registration inaccuracies that are caused by the detection and pose estimation errors [Yoshida et al. 2008], [Hill et al. 2011], [Tomioka et al. 2013].

We present an approach to create a dynamic user-perspective rendering using projective geometry, based on Virtual Reality principles. We apply an external camera-based motion tracking system to capture the screen and the user's eye pose. Using an external tracking system results in accurate tracking of the head and the screen although it dictates limitations to the system. It possible to replace the tracking part without changing the rendering method.

*e-mail: ali.samini@liu.se

†e-mail:karljohan.lundin.palmerius@liu.se

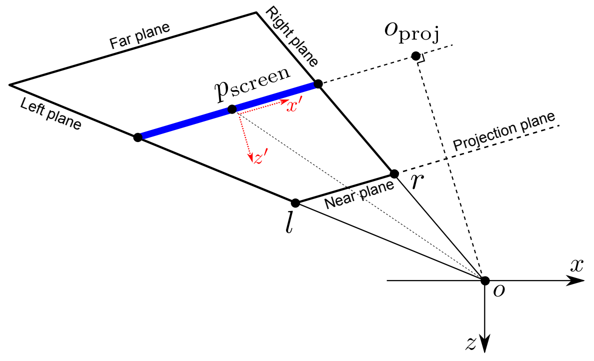

Figure 1: *Top-view of the dynamic frustum. Here, o and p_{screen}, show the eye and the screen position. These are used to calculate the frustum planes.*

2 Dynamic View and Frustum

The *view matrix* of graphics library (e.g OpenGL) expresses the extrinsics of our virtual view. The *position* of the origin of this virtual view should be the same as the real view, thus the tracked eye position of the user. We define a projection plane (the *near plane* in OpenGL) parallel to the screen, that makes the projection of virtual objects in correct perspective for the viewer, from their angle. The virtual view *orientation* should be the same as the orientation of the screen, see Fig. 1. Thus, we create the view matrix, M_{view}, as

$$M_{\text{view}} = \left(\begin{array}{c|c} R_{\text{screen}} & \vec{t}_{\text{eye}} \\ \hline 0 & 1 \end{array} \right) \tag{1}$$

where R_{screen} is the orientation of the screen and $\vec{t}_{\text{eye}}$ is the position of the user's eye given in homogeneous coordinates.

The intrinsics of the virtual view is expressed as a truncated pyramid, called the *frustum*. Our approach makes a dynamic frustum that changes with the movement of the eye and the screen and perfectly matches the edges of the screen, see Fig. 1.

The positions of frustum planes are specified on the near plane and are calculated in the camera's frame of reference. Based on triangle similarity we can estimate the left plane position, l as

$$l = \frac{n\left(\vec{o}_{\text{proj}} + \hat{x}' \cdot (\vec{p}_{\text{screen}} - \vec{o}) + \frac{1}{2}W_{\text{screen}}\right)}{|\vec{o}_{\text{proj}} - \vec{o}|} \tag{2}$$

where W_{screen} is the width of the screen and $\vec{o}_{\text{proj}}$ is the projection of the eye position, $\vec{o}$, onto the plane of the screen. The position of other frustum plane can be estimated similarly.

3 Video See-through Rendering

To create a correct alignment between the real world and the rendered video feed on the screen in V-AR, both the *extrinsic* and the *intrinsic* parameters of the camera are needed. We use a tracked camera and thus have the extrinsic parameters available for each frame, based on the movement of the camera. The camera is calibrated using the pinhole camera model approximation that gives

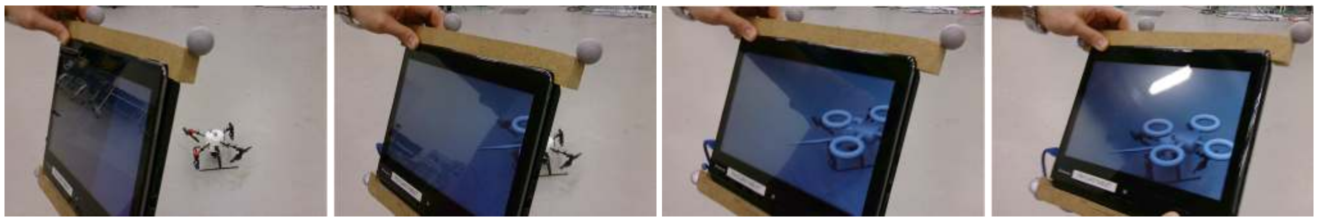

Figure 3: *Sequential pictures of our running system taken by a secondary camera that is tracked as head position.*

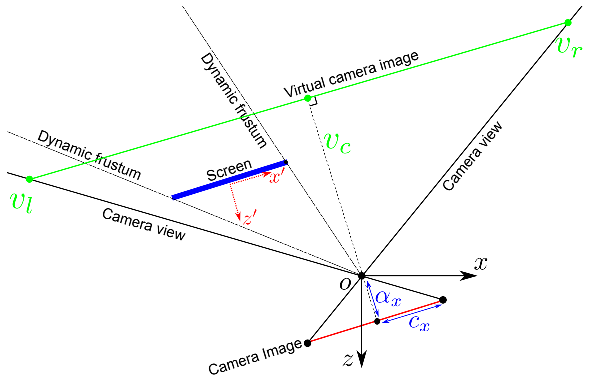

Figure 2: *The virtual camera image in the virtual world.*

us the intrinsic parameters including the focal length of the camera in x and y directions described in pixels (α_x and α_y) where, $\alpha_x = fm_x$ and $\alpha_y = fm_y$, f being the focal length of the camera and m_x, m_y are scale factors that relate distance to pixels and the position of the principal point of the camera (c_x, c_y). The video feed is then rendered on a textured rectangle that represents the virtual camera image inside the 3D virtual world. To align the real and virtual camera, the virtual image rectangle is created based on the camera's intrinsic parameters, parallel to the image plane of the camera, as shown in Fig.2. We can estimate the distance from the centre to the left of the virtual camera image, v_l as

$$v_l \quad = \quad -ov_c \frac{c_x}{\alpha_x} \qquad (3)$$

where ov_c is the arbitrary distance of the virtual plane from the eye. α_x and c_x are x components of camera's focal length and principal point expressed in pixels. The right (v_r), the top (v_t), and the bottom (v_b) boundaries of the virtual camera image can be estimated similarly.

What the camera sees and what the user sees will never be the same, since the camera cannot be co-located with the user's eye. Here, there are three views of the world: (1) the *real view*, as seen around the screen, (2) the *video feed* shown on the screen, and (3) the *virtual augmentation* that also is rendered on screen. By rendering the camera image as if the camera is located at the head position ,in equation 1, we introduce a misalignment between the real view and the video feed, as well as between the video feed and the augmentation while creating an alignment between the real view and the augmentation. The amount of misalignment introduced here, however, will be identical to the displacement of the camera from the eye.

We can improve the alignment by placing the virtual view in the same place as the real camera as if the head was located at the camera position. This corrects the alignment between the video feed and the virtual augmentation while destroys the alignment between the real world and the augmentation. Augmentations are merged with the video feed thus their misalignment are visually more apparent than the one between them and the real world. The new view matrix is calculated based on the tracked position and orientation of the camera.

4 System Test and Results

Our implementation and prototype is based on a tablet PC tracked with a Vicon camera-based tracker system, for an application in the control and visualization of Unmanned Aerial Vehicles (UAVs). The tablet is equipped with a Go-Pro camera; allowing a wide perspective, feeding the video through a frame grabber. Our prototype worked without noticeable lag or latency with a rendering speed of 30 frames per second during our test. The system was tested in our tracking enabled environment. The task was to follow a flying quadcopter with our tracked V-AR tablet using both the DPR and UPR views. The test participant had to wear a pair of tracked transparent glasses for the eye tracking. A virtual 3D model of the tracked quadcopter where augmented on the rendered video to test the alignment between the virtual augmentation and both the real view and video feed. Figure3 shows sequential pictures of our system test results while using the UPR view.

5 Conclusions

Our prototype system was successful to deliver the UPR view without noticeable jitter or latency through our informal tests. There where misalignments that where caused mostly by the displacement of the eye and the camera and also the fact that the camera and screen where not perfectly aligned. We plan to do the screen-camera calibration in the continuation of our research.

Acknowledgements

This work has been supported by SSF through a grant for the project CUAS. The authors would also like to thank Joakim Kilby for his help with the project development and the staff at the Indoor Micro Unmanned Aerial Vehicle Laboratory.

References

HILL, A., SCHIEFER, J., WILSON, J., DAVIDSON, B., GANDY, M., AND MACINTYRE, B. 2011. Virtual transparency: Introducing parallax view into video see-through ar. *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR) 0*, 239–240.

TOMIOKA, M., IKEDA, S., AND SATO, K. 2013. Approximated user-perspective rendering in tablet-based augmented reality. In *Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on*, 21–28.

YOSHIDA, T., KUROKI, S., NII, H., KAWAKAMI, N., AND TACHI, S. 2008. Arscope. In *ACM SIGGRAPH 2008 New Tech Demos*, ACM, New York, NY, USA, SIGGRAPH '08, 4:1–4:1.