More for Less: Fast Image Warping for Improving the Appearance of Head Tracking on HMDs

Edward Peek
Department of Computer Science
The University of Auckland
Auckland, New Zealand
Email: epee004@aucklanduni.ac.nz

Christof Lutteroth
Department of Computer Science
The University of Auckland
Auckland, New Zealand
Email: lutteroth@cs.auckland.ac.nz

Burkhard Wünsche
Department of Computer Science
The University of Auckland
Auckland, New Zealand
Email: burkhard@cs.auckland.ac.nz

***Abstract*—In consumer 3D applications such as video games, users frequently have the option to adjust the trade-off between graphics quality and frame rate, with a common approach being to maximise graphics quality while retaining an acceptable (but not ideal) frame rate. However, consumer head-mounted displays require ideal frame rates in order to prevent discomfort and motion sickness. In this paper we present a minimal image warping method to perceptually increase the frame rate for critical types of motion (particularly head tracking). We analyse the magnitude of graphical artefacts introduced by image warping, when used with the Oculus Rift developer kit 1, and perform a user study evaluating improvements in viewing experience achieved with our technique. The results of the study indicate that for head orientation tracking, even basic image warping is perceptually the same as rendering at an ideal frame rate.**

***Keywords*—head-mounted display, image warping, virtual reality**

I. Introduction

Despite 3D computer applications rising in popularity, and the increasing availability of immersive VR displays, most users still view 3D content on a flat panel monitor that takes up a small portion of their visual field. More immersive display technologies are however starting to replace conventional monitors in certain usage scenarios. Stereoscopy is one such technology that has appeared in computer monitors, television sets, handheld gaming consoles and smartphones. It is the only such technology gaining any traction however, and is doing so very slowly.

Another class of immersive display technology is head-mounted displays (HMDs). These produce a significantly more immersive experience than both conventional monitors and stereoscopic displays. HMDs achieve this by having a wide field-of-view, by blocking out vision of the real world, and by coupling movement of the user's head to movement of the virtual viewpoint. While HMDs have previously been targeted towards specialised application areas such as military and medical training, industrial and scientific visualisation, and VR enthusiasts, recent advances have allowed them to become targeted towards ordinary consumers. One notable example is the Oculus Rift [1] which has backing from several major video game developers and industry experts [2].

For applications to utilise such novel display devices, they must be adapted in several key ways. Firstly an application must be adapted to output images in a format compatible with the display. The application must also be adapted to reflect the conceptual model of the HMD, specifically to support head tracking. Lastly quality factors must be considered, in particular rendering frame rate and latency. While these two factors are also important when dealing with conventional monitors, they are much more important for HMDs as they are much more likely to cause simulation sickness. This matter is complicated by the fact that these factors are strongly influenced by the speed of the user's computer hardware. This leads to the scenario where a specific combination of application and computer hardware may provide an acceptable experience on an conventional monitor, but one that is unacceptable on an HMD. Such experiences are likely to discourage the use of HMDs. To counteract this, application developers need methods for improving these quality factors on HMDs, so that they are just as accessible as conventional monitors.

In this paper, we explore image warping as a method for improving the perceived latency and frame rate of one of the most important aspects of a HMD: head tracking. In particular, we have developed a test system that uses an implementation of image warping specifically tailored for use with HMDs on consumer PCs, and evaluated it to gauge how much it improves the user's perception of head tracking quality.

II. Related Work

Almost any form of latency may be reduced through prediction. In the domain of HMDs, prediction is typically applied to reduce head tracking latency, as the orientation and position of the user's head can be extrapolated reasonably accurately and this type of latency is a significant factor in causing simulation sickness. The effect of employing this type of prediction is discussed by Azuma and Bishop [3] who found that, for their HMD system, the magnitude of tracking error (caused by system latency) can be reduced by a factor of 2–10 depending on the type of prediction used. Prediction is especially useful in that is simple to implement and can be used in addition to other forms of latency reduction.

Another method to reduce tracking latency is to sample the orientation and position as late in the rendering process as possible. One possible method discussed by Kijima and Ojika [4] is to effectively use the head orientation at the time of the display scan-out through the use of a special LCD modulation circuit. This allows the visible image to be based on very up-to-date tracking information, but has limited quality as only basic adjustments may be made so late in the rendering process.

A similar approach is utilised by Olano et al. [5]. The authors use both scan-line adjustment and a highly parallel hardware configuration to reduce latency to less than a single NTSC field time, as well as guarantee an ideal frame rate. Scan lines are offset individually to compensate for the fact that the lower the line in the frame, the later it will be refreshed on the display. The architecture of such a system is however significantly different to modern consumer PCs, making of limited use.

An solution more appropriate for modern PCs is proposed by Mark et al. [6]. This is an entirely software based approach that employs a 3D image warp in a fast secondary update loop to compensate for latency introduced by a slow main render loop. Such an approach can improve frame rate as display frames can be rendered independently and in parallel to the main application render loop. While this approach is not be able to reduce all sources of latency, it can reduce the largest and most unpredictable one, leaving the rest for reduction via prediction. Further work by Smit et al. discusses an implementation targetting fish-tank VR ssytems [7] and the performance/quality tradeoffs of several 3D warp algorithms [8]. This approach is quite practical for consumer applications, as being software based it is easily accessible, and it is designed for PC systems similar to what most consumers currently own.

III. Our Approach

Our approach to image warping is similar at a high level to the single-GPU system developed by Smit et al. [8]. In this sense both systems perform image warps at a rate equal to the display's refresh rate, while rendering the virtual environment at some lower frequency. The major difference with our system is that it does not generate or use motion fields: but instead uses an algorithm that only requires the rendered images' colour data. Furthermore, the algorithm we use to perform the image warp is much more simplistic, and therefore capable of running quickly on less capable hardware. This makes it more practical, as it places no demands on how the application should render the virtual environment, while also taking few computation resources away from the application. We refer readers to the publications of Smit et al. [8] for specific architectural details.

The most important design point, other than the overall system architecture, is the algorithm that performs the warp itself. Previous research in image warping has established methods for accurately extrapolating various types of motion, including that of scene objects and the camera's viewpoint. While our algorithm is less capable than these, it is sufficient for improving the appearance of head tracking on HMDs.

Our algorithm renders warped image frames through the following process. A quad mesh is aligned to the location of the far clipping plane of the view frustum from the previously rendered simulation frame. The image data from this simulation frame is then mapped as a texture to this quad. Warped frames are produced by rendering just this quad with a new camera position and orientation, as illustrated in Figure 1.

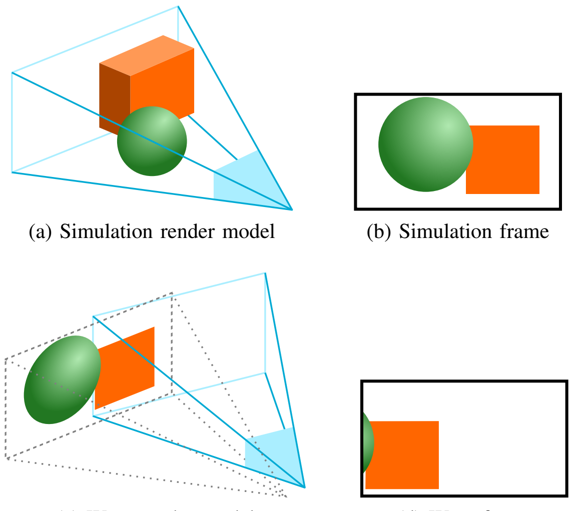


Fig. 1. Warped frames are produced by drawing a simulation frame from a new camera position

An unoptimised implementation of this algorithm runs in ~ 0.2 milliseconds on an NVIDIA GeForce GTX 580 high-end dedicated GPU, and in ~ 3 milliseconds on an Intel HD Graphics 3000 low-end integrated GPU. If the display refresh rate is 60Hz and the simulation normally renders at 30 FPS, this should cause a decrease to 29.6 FPS and 24.5 FPS respectively due to resource contention. On the high-end GPU the reduction in base frame rate is negligible. While the drop on the low-end GPU is noticeable, it shows the algorithm executes sufficiently fast on just about the entire range of modern PC GPUs. This also indicates that this algorithm is suitable for offloading to the integrated GPU in computer systems that have both dedicated and integrated GPUs.

IV. Theoretical Evaluation

Evaluating an image warping system is a matter of determining whether it correctly smooths the desired type of motion, and whether the artefacts it introduces are sufficiently small.

The algorithm described in this paper is specifically designed to only smooth head tracking on HMDs, and no other types of motion. In this section we analyse the theory of how well this goal is achieved in typical usage. We also examine the major type of artefact introduced by this algorithm, what we call *fringe artefacts*. Other types of artefacts that may be present are not discussed, as they largely depend on the lighting and shading models used for rendering the scene, making them difficult to generalise for.

A. Tracking Smoothness

Our algorithm intrinsically perfectly smooths pure rotations around the virtual camera's position. Unfortunately these types of rotation are nearly impossible with a HMD, due to the fact that there are two virtual cameras (one for each eye), and that rotations of the user's head happen around their neck (not their eyeballs). Instead, rotating the user's head typically introduces small translations that must also be smoothed. This is characterised by Figure 2, which illustrates the difference in

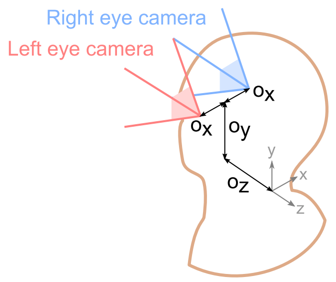


Fig. 2. The head model used to transform the virtual cameras in response to HMD orientation. HMD rotations are applied to the origin at the base of the neck, which causes both rotation and translation of the cameras

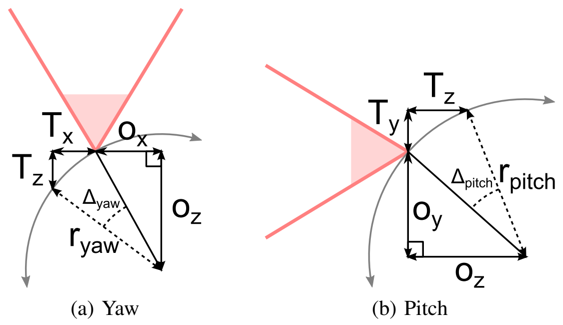


Fig. 3. Derivation of camera translation due to two types of head rotation

position between the virtual camera and the point they rotate about.

This is the point where our choice of a simple warping algorithm begins to introduce issues. While the algorithm we use can handle viewpoint translations, it can not do so perfectly. It effectively makes the assumption that all surfaces in the scene are 2D and lie on the viewing frustum's far-clip plane. This means that nearby surfaces will not be shifted by the correct amount when warping, making their motion jerky when translating the viewpoint.

The rest of this section derives the magnitude of the error between the proper shift, and the actual shit produced by our algorithm. For brevity, we only present the derivation for considering yaw head rotations (i.e. looking left and right). Yaw was chosen since it had the greatest magnitude from the types of rotation recorded in our testing. The second most prevalent type of motion was pitch (i.e. looking up and down), which can be trivially adapted from our equations by changing which of the x, y and z axes are considered.

Image warping must only deal with changes since the rendering of the last simulation frame. The most critical variable that results from this is t_c, the *compensation time*. t_c is the difference in time between the time at which head orientation was sampled for use in rendering the simulation frame, and the time at which head orientation was sampled for use in rendering the current warped frame. t_c strongly influences how different the warped frame will be from the simulation frame, and consequently the size of many types of errors. t_c varies between different warped frames, but for making assumptions we assume it to be approximately 32 milliseconds, the upper bound of what it could be in our system when increasing the effective frame rate from 30 FPS to 60 FPS.

All translations are derived from head rotation in our approach. We must first determine the magnitude of these translations. Given that $\Delta_{yaw} = t_c * \omega_{yaw}$ (where ω_{yaw} is the angular velocity of head rotation in radians), we produce the following equations.

$$\theta_{y0} = \arctan\frac{o_x}{o_z} \qquad \theta_{y1} = \theta_{y0} + \Delta yaw \qquad r_y = \sqrt{o_x^2 + o_z^2}$$

$$T_x = -r_y(\sin\theta_{y1} - \sin\theta_{y0}) \tag{1}$$
$$T_y = 0 \tag{2}$$
$$T_z = -r_y(\cos\theta_{y1} - \cos\theta_{y0}) \tag{3}$$

The derivation of these equations is illustrated in Figure 3, as well as the analogue for pitch.

Using generic values of $o_x = 0.032$m and $o_z = 0.07$m (which depend on the user, see Figure 2), we can take small angle approximations of T_x and T_z to give us

$$T_x \approx 0.067\Delta_{yaw} \tag{4}$$
$$T_y = 0 \tag{5}$$
$$T_z \approx 0.038\Delta_{yaw} \tag{6}$$

indicating that camera translation is approximately proportional to the speed of head rotation.

For surfaces in the centre of the screen, and translations of the camera directly orthogonal to the viewing direction, the error in the position of the warped surface (as an angle of the visual field) is given as e by Equation 7. Here c_f is the far clipping distance, d is the distance to the surface and T is the magnitude of the camera translation.

$$e = \arctan\frac{c_f}{T} - \arctan\frac{d}{T} \tag{7}$$

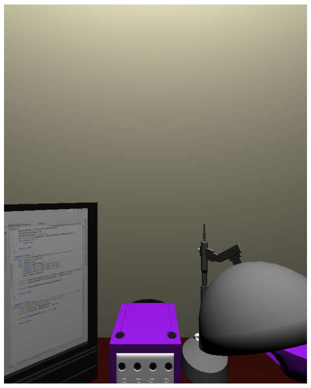
From this it can be seen that as the surface moves away from the camera, the error becomes negligible, as our assumption that all surfaces lie on the far clipping plane becomes more accurate.

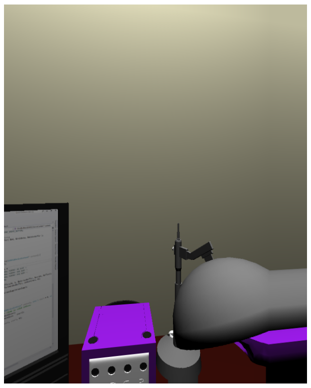
Because the error is in the location of surfaces, the visible effect will be that, as the camera translates, the surfaces will move with a jerky motion, with the jerks occurring at the frequency as the simulation update rate. This gives a similar appearance to the jerkiness caused by ordinary low frame rates. However, because the warping always *under-compensates* for translation (but still compensates a little), the size of the jerks will always be less than if no image warping was used.

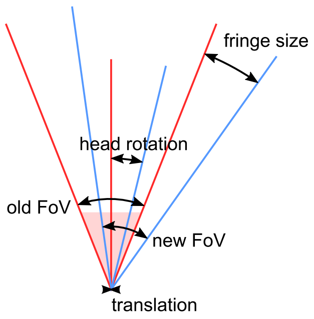
B. Fringe Artefacts

Fringe artefacts occur because the images that we use as a source for warping have a limited field-of-view. When the image is warped to create a new view direction, holes occur at the edges of the image as the old and new frustums do not perfectly overlap. These holes may be filled by stretching the edges of the simulation frame to fill the new frustum bounds.

When warping, the algorithm used in this study projects the rendered simulation frame to the distance of the far clip plane. Because the far clip distance is so large in typical applications, and the viewpoint translation due to head rotation so small (Section IV-A), translations of the virtual cameras


(a) Before warp, without artefacts


(b) After warp, with fringe artefact



(c) Derivation of fringe size

Fig. 4. Illustration of a fringe caused by camera yaw, and translation (which is negligible compared to the size of the frustum)

have a negligible impact on the size of fringe errors. Instead, for yaw and pitch head rotations, the size of the fringe artefact is exactly equal to the amount of rotation between the current image frame and the previous simulation frame (i.e. Δ_{yaw} or Δ_{pitch}). This can be seen in Figure 4.

We expect fringe artefacts to be minimally visible in typical usage. At a head rotation speed of 45 degrees/s and $t_c \leq 32$ ms, the maximum size of the fringe in the user's visual field is only 1.44 degrees. Looking forward, the artefact appears at the edge of peripheral vision, where visual acuity is worst. Additionally, these artefacts can not be looked at directly, as the geometry of the HMD causes them to be occluded by the edge of the lens when the eye is not pointing directly forward.

V. Empirical Evaluation

In addition to the the technical properties of image warping artefacts, we are also interested in how perceptible they are to users in typical scenarios. For this purpose, we performed a user study to answer the following questions.

1) Does image warping improve the perception of head tracking when the simulation frame rate is less than the display refresh rate?
2) Can users tell the difference between image warping and the ideal scenario where the simulation frame rate equals that of the display refresh rate?

A. *Test Conditions*

5 test conditions were used in our study, which were each made up of a combination of a rendering method and a fixed simulation frame rate. The two rendering methods used were

- Fixed frame rate (FR): no image warping is employed, and simulation frames are generated at a fixed fraction of the display's refresh rate.
- Image warping (IW): image warping is applied to FR rendering. Image warping is always done at 60 FPS (the native refresh rate of the Oculus Rift DK1), while the rendering of simulation frames occurs at a lower rate.

The frame rates used though the study, and the reason for choosing them are as follows:

- 30 FPS ($1/2$ DK1 refresh rate): approximately the threshold for acceptably smooth motion. This is sometimes used as a target for video games in order to improve other aspects of image quality. Strategies like post-process motion blur are often employed to make such frame rates more acceptable.
- 45 FPS ($3/4$ DK1 refresh rate): the mid point of the range of typical frame rates.
- 60 FPS (equal to DK1 refresh rate): the ideal scenario; any increase in rendering frame rate above this is clamped to the refresh rate of the display hardware. 60Hz is the maximum refresh rate of most computer monitors and television sets, although select models are able to go higher.

The 5 test conditions are made up of the following combinations of these components.

1) Fixed frame rate, with 30 FPS simulation (FR30)
2) Image warping, with 30 FPS simulation (IW30)
3) Fixed frame rate, with 45 FPS simulation (FR45)
4) Image warping, with 45 FPS simulation (IW45)
5) Fixed frame rate, with 60 FPS simulation (FR60)

Conditions 1 and 3 mimic the scenario where a computer's hardware is not fast enough to render the scene at the perfect rate. Conditions 2 and 4 represent how conditions 1 and 3 can be improved using image warping. Condition 5 is the ideal scenario, where the computing hardware is sufficiently fast.

B. *Test Scene*

A virtual environment was developed for testing, which can be seen in Figure 5. The environment allows the user to look around by turning their head, but no other interaction apart from controlling the head-up display (HUD) is possible.

The environment was developed to allow enabling and disabling image warping, and to control the rendering frequency of the simulation. A 40% extension of the rendered image size was used to compensate for the shrinking of the image caused by correcting for lens distortion. While this can hide fringe artefacts (Section IV-B), it is often done in actual HMD applications, leading us to believe it is appropriate to include.

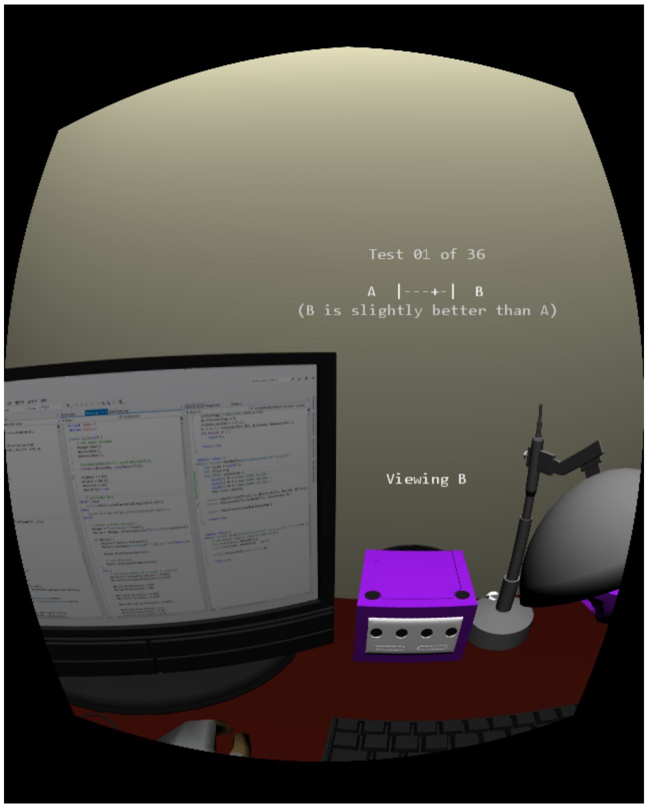


Fig. 5. Rendering of the virtual environment, with generic lens correction. The head-up display is drawn a fixed distance in front of the user's face, and provides feedback for their selection of ratings.

Our test application used an unconventional implementation of warping architectures so that there would be more control over different variables when performing the evaluation. The first difference is that, in our environment, the simulation rendering is computationally lightweight, and is therefore capable of updating several times faster than the display's refresh rate. How often the simulation thread is allowed to update is however artificially limited to a fraction of the display's refresh rate, in order to mimic an actual system. In addition to this, with our system the simulation and warp threads are not run in parallel, but are instead manually scheduled on a single thread, with all rendering being performed on a single GPU. We are able to do this since resource contention is not an issue, as enabled by the previous point. This allows us to accurately schedule timing of input sampling, simulation update and rendering, warp rendering and presentation of the warped frames.

With regard to the appearance of the environment itself, it is modelled after a simple room, where the user [9] is sitting at a desk holding several objects [10]. Lighting is fixed, there are no anisotropic materials, no post-processing and no moving objects, making the scene highly favourable for image warping. Such an environment was chosen as it is reasonably stressful for the specific types of errors produced by our warping algorithm, while still being reasonably realistic, and similar to actual virtual environments.

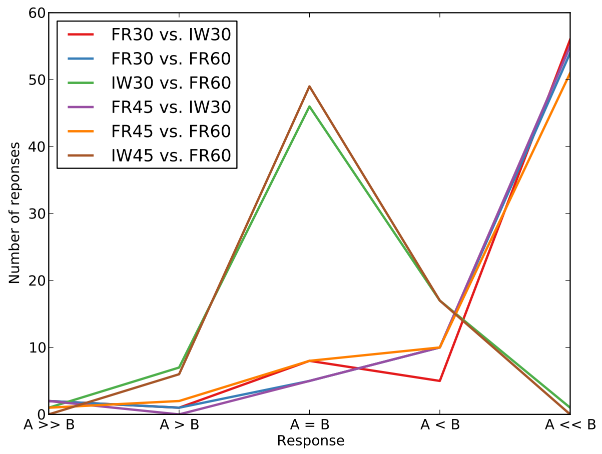


Fig. 6. Distribution of the ratings from all participants

C. Procedure

The general study procedure was for participants to perform a series of comparisons between pairs of different conditions, and rate which of the two they thought tracked head motion better. The pairing of conditions was chosen to answer the research questions outlined above.

- FR30 vs. IW30, and FR45 vs. IW45 pairs were chosen to answer Question 1.
- IW30 vs. FR60, and IW45 vs. FR60 pairs were chosen to answer Question 2.
- FR30 vs. FR60, and FR45 vs. FR60 pairs were chosen to serve as a control.

Every paring was presented to every user user 6 times — to average out inconsistencies in responses — giving 36 pair comparisons per participant. The order in which the pairings were presented was random. For each pair, participants could switch between the two conditions at will, and were asked to rate the difference in quality between the conditions of a 5-point scale.

All interaction was performed by using a keyboard with hands in fixed locations and all interactions displayed on screen. This was done because the Oculus Rift blocks vision of the real world, preventing the user from seeing their hands and keyboard during the tests.

In addition to recording the users ratings for each test, the angular velocity of head rotation was also recorded. Additionally, users were asked to fill out a questionnaire before and after they performed the testing.

D. Results and Discussion

12 people participated in the study. Most were computer science students or staff, with a split of 10 male and 2 female. 7 participants stated they had used 3D computer applications at least ”sometimes” in the last 3 months, while the rest stated they had used them either ”rarely” or ”never”.

A visual inspection of the distribution of all responses, shown in Figure 6, allows the following observations.

- For the majority of participants FR60 is better than either FR30 or FR45.
- For the majority of participants IW30 and IW45 are better than FR30 or FR45 respectively.
- Most participants can not tell the difference between IW30 and FR60, or between IW45 and FR60.
- The skew in IW30 vs. FR60, and IW45 vs. FR60, indicates that some users may be able to tell the difference between them, and they consider the difference "slight".
- Some participants either did not notice the difference in frame rate, guessed the rating, or did not move their head fast enough to notice the difference.

Wilcoxon signed-rank tests were performed to validate the first 4 of these points. We found there is a significant difference ($p < 0.05$) between the "good" conditions (IW30, IW45 and FR60), and the "poor" conditions (FR30 and FR45) in all the tests that compared them. These results back up our first two points.

What is less clear is the whether IW30 is significantly different from FR60, and IW45 from FR60. Wilcoxon tests result in p-values of $p = 0.079$ and $p = 0.022$ respectively. The fact that the tests are so borderline prevents us from establishing whether the compared modes are fully equivalent or not.

In response to this, we reformulate a more conservative hypothesis [11], that "*most* participants can not tell the difference between the two conditions" (rather than *all*). Performing sign tests against this hypothesis yields significant results ($p \approx 1$) for both comparisons. The fact that most participants can not tell the difference, but not all can not, indicates the differences are only noticeable in some usage scenarios, and/or by some people. While it would be interesting to test individual participants for their ability to tell the difference between these conditions, with only 6 samples per test we do not believe this can be done reliably with the data from our study.

We believe the existence of "incorrect" ratings (i.e. people rating lower frame rates as better than high ones) are due to participants either guessing, or incorrectly recording their rating; not because they genuinely preferred lower frame rates.

Participant remarks also revealed that some found that low frame rate conditions specifically gave them noticeable discomfort, mostly described as "headaches". Unfortunately, the testing was not structured in such a way as to pair these remarks with the actual conditions, limiting the conclusions that can be drawn from this. It does however give weak evidence that smoothness of head tracking (as a result from a high frame rate) does play a part in introducing motion sickness on HMDs.

While the results from this study are reasonably conclusive for just head tracking, the virtual environment is unrealistically static, potentially hiding quality issues that may occur in real-world applications.

In summary, these results suggest that even simplistic image warping algorithms are able to dramatically improve the smoothness of head tracking. The evidence also suggests that this type of image warping is difficult — and potentially impossible — to differentiate from ideal frame rates, even when the scene is internally being rendered at only half the ideal rate.

VI. Conclusions

In this paper we present a fast image warping system for HMDs and show that it is fast enough to be practical even on a low-end PC integrated GPU.

A theoretical analysis of the chosen warping algorithm shows how the size of errors due to incorrect assumptions grows proportionately with the speed of head rotation and the amount of delay the warp is compensating for.

An empirical evaluation — in the form of a user study — was also performed to asses the visibility of these errors on the Oculus Rift DK1. Our analysis of the results leads us to conclude that even basic image warping algorithms are able to significantly increase the perceived quality of head tracking, making their use appropriate for most 3D PC applications targeting HMDs.

Lastly, our study dealt with image warping for head rotation only. Further investigation into perception and performance of other image warping algorithms, which can better compensate for other types of motion, is needed. The visibility of artefacts due to anisotropic materials, transparent objects, and post-processing, should be incorporated into such a study, as they challenge the assumptions of most image warping algorithms.

References

[1] Oculus VR. (2012) Oculus rift - virtual reality headset for 3d gaming. [Online]. Available: http://www.oculusvr.com/

[2] ——. (2013, Aug.) John carmack joins oculus as cto. [Online]. Available: http://www.oculusvr.com/blog/john-carmack-joins-oculus-as-cto/

[3] R. Azuma and G. Bishop, "Improving static and dynamic registration in an optical see-through hmd," in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '94. New York, NY, USA: ACM, 1994, pp. 197–204.

[4] R. Kijima and T. Ojika, "Reflex hmd to compensate lag and correction of derivative deformation," in *Virtual Reality, 2002. Proceedings. IEEE*, 2002, pp. 172–179.

[5] M. Olano, J. Cohen, M. Mine, and G. Bishop, "Combatting rendering latency," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, ser. I3D '95. New York, NY, USA: ACM, 1995, pp. 19–ff.

[6] W. R. Mark, L. McMillan, and G. Bishop, "Post-rendering 3d warping," in *Proceedings of the 1997 symposium on Interactive 3D graphics*, ser. I3D '97. New York, NY, USA: ACM, 1997, pp. 7–ff.

[7] F. A. Smit, R. van Liere, and B. Fröhlich, "An image-warping vr-architecture: design, implementation and applications," in *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, ser. VRST '08. New York, NY, USA: ACM, 2008, pp. 115–122.

[8] F. Smit, R. Van Liere, S. Beck, and B. Froehlich, "An image-warping architecture for vr: Low latency versus image quality," in *Virtual Reality Conference, 2009. VR 2009. IEEE*, 2009, pp. 27–34.

[9] TiZeta, "Lowpoly man," http://www.blendswap.com/blends/view/66412, 2013.

[10] DeNapes, "desk," http://www.blendswap.com/blends/view/50183, 2012.

[11] R. H. Randles, "On neutral responses (zeros) in the sign test and ties in the wilcoxonmannwhitney test," *The American Statistician*, vol. 55, no. 2, pp. 96–101, 2001.