Extended TimeWarp Latency Compensation for Virtual Reality

Daniel Evangelakos∗
NVIDIA

Michael Mara†
NVIDIA

Figure 1: A simple scene with the camera moving forward towards the boxes and glossy sphere. The first figure is the initial image, and the second is the ground truth solution. The two images on the right show two enriched warping solutions. Both get the proper positions for the boxes and spheres. The red rectangles in both images show the visual effects that result from using these warping solutions.

Keywords: virtual reality, timewarp, reprojection

Concepts: Computing methodologies → Virtual reality; Image-based rendering;

1 The Latency Compensation Problem

Head-mounted virtual reality displays exhibit latency between sensing the user head position and updating the camera transformation for the image on the screen. With current sensors, rendering pipelines, and displays, this latency is on the order of several milliseconds and is believed to contribute to simulator sickness symptoms of disorientation, headache, and nausea.[LaValle et al. 2014]

There are two, combinable main techniques for reducing the perception of this latency without modifying the rendering pipeline. Prediction schemes replace the current actual camera frame from the tracker at rendering time with the anticipated camera frame at the time that the image will be displayed, using techniques such as dead reckoning and Kalman filters [e.g., 1]. This is used by all current VR SDKs.

The TimeWarp algorithm, which is currently deployed by Oculus in their VR SDK, warps the final rendered image immediately before display. This warp attempts to account for any rotation discrepancy between the predicted and actual camera at display time. While the same techniques could be applied for extrapolation when the renderer falls below frame rate, this may produce an inferior experience.

We evaluated the rotation-only TimeWarp algorithm as described in Oculus’s public materials [Oculus 2016], and some extensions to translation and animation warping. We evaluated the visual results of applying these warping solutions on scenes in which the subject and scene were moving.

2 Oculus TimeWarp

Oculus’s current TimeWarp solution performs a 2-dimensional rotational warp on the rendered image right before it is sent to the HMD. The warp involves rendering the original image as a billboard at infinity using the most recent tracking data for the camera position. To allow for significant head motion to be accounted for, large guard bands must be rendered on the border of the original image. This algorithm accounts perfectly for any rotation of the camera but fails for all other changes in the scene.

2.1 Enhanced TimeWarp

To produce a warping solution that can account for more than just camera rotation, more data must be stored when rendering. Using depth and motion vectors, a three dimensional model of per pixel quads can be used in place of geometry as a model of the scene that is then re-rendered using the most recent camera position just before display.

This three dimensional approach can be used to compensate for a greater range of changes in the scene but also leads to issues related to dislocations or holes. This can be seen in the third figure in the teaser. In these cases some part of the scene that was previously occluded becomes visible to camera. Some common strategies for dealing with this problem have involved stretching quads to fill the dislocation. This can also be achieved by using a connected height-field instead of separate quads. Instead of stretching to fill the holes, data from other eye or the results of a simple rectangular warp can be used.

Using data from the other eye would allow for correct results but would not be guaranteed to fill all holes. An extension of this solution would be to render a second depth level and then perform a warp using both levels. This would of course require a large investment of resources and would still not guarantee that all holes are filled. As a final back up holes can be filled using the simple two dimensional warp that Oculus currently uses. While this would not have any guarantee of being correct, the error is less than having nothing displayed at all.
All of these warping solutions use the original frame for shading information. Therefore they are unable to handle transparency or glossy reflections correctly. This is especially noticeable when motion vectors are used in the warping process. In the cases where motion vectors are used the shadows are also incorrect. Since the small latency in object positions has a far less noticeable effect than the latency of the camera position, the use of motion vectors in the warp is questionable.

Acknowledgements

We would like to thank Morgan McGuire and Dave Luebke (NVIDIA) for their help on this project.

References
