IEEE STANDARDS ASSOCIATION

[VR Lens Distortion]

[Sangkwon Peter Jeong / JoyFun Inc.,]

Compliance with IEEE Standards Policies and Procedures

IEEE P3333.3

•**Subclause 5.2.1 of the *IEEE-SA Standards Board Bylaws* states, "While participating in IEEE standards develop ment activities, all participants...shall act in accordance with all applicable laws (nation-based and internation al), the IEEE Code of Ethics, and with IEEE Standards policies and procedures."**

•The contributor acknowledges and accepts that this contribution is subject to

- The IEEE Standards copyright policy as stated in the *IEEE-SA Standards Board Bylaws*, section 7, http://standards.ieee.org/develop/policies/bylaws/sect6-7.html#7, and the *IEEE-SA Standards Board Operations Manual*, section 6.1, http://standards.ieee.org/develop/policies/opman/sect6.html
- The IEEE Standards patent policy as stated in the *IEEE-SA Standards Board Bylaws*, section 6, http://standards.ieee.org/guides/bylaws/sect6-7.html#6, and the *IEEE-SA Standards Board Operations Manual*, section 6.3, http://standards.ieee.org/develop/policies/opman/sect6.html

VR Lens Distortion

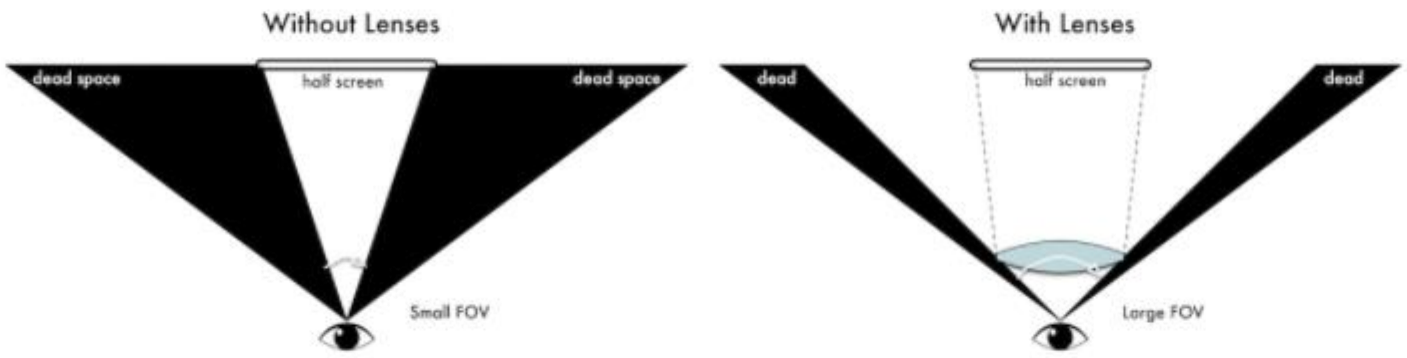
Date: 2017-04-24

Author(s):

Name	**Affiliation**	**Phone [optional]**	**Email [optional]**
Sangkwon Peter Jeong	JoyFun Inc.,	+82 10 8667 7329	ceo@joyfun.kr

Three approaches to VR lens distortion

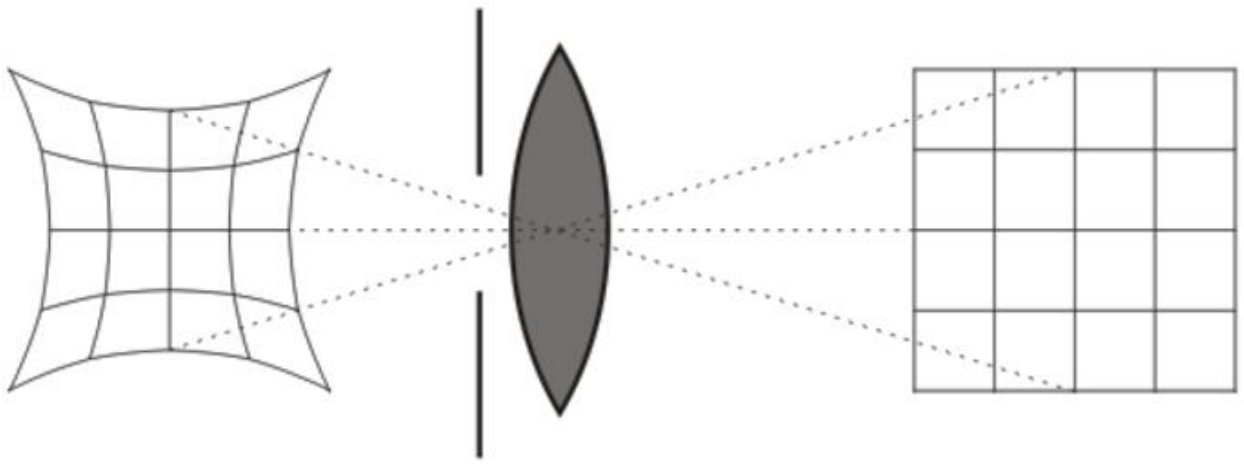
Immersion requires a large field of view. This could be achieved by putting a large curved spherical display on your face, but alas such technology is prohibitively expensive. A more affordable solution to increasing the field of view is to look at small ubiquitous rectangular displays through lenses:



Three approaches to VR lens distortion

Lenses placed close to your eyes greatly increase your field of view, but there is a cost: the image becomes spherically distorted. The larger the field of view, the more distorted the image. This post is a quick summary of three different approaches to undistorting the image, all of which have been implemented in JavaScript for various WebVR-related projects.

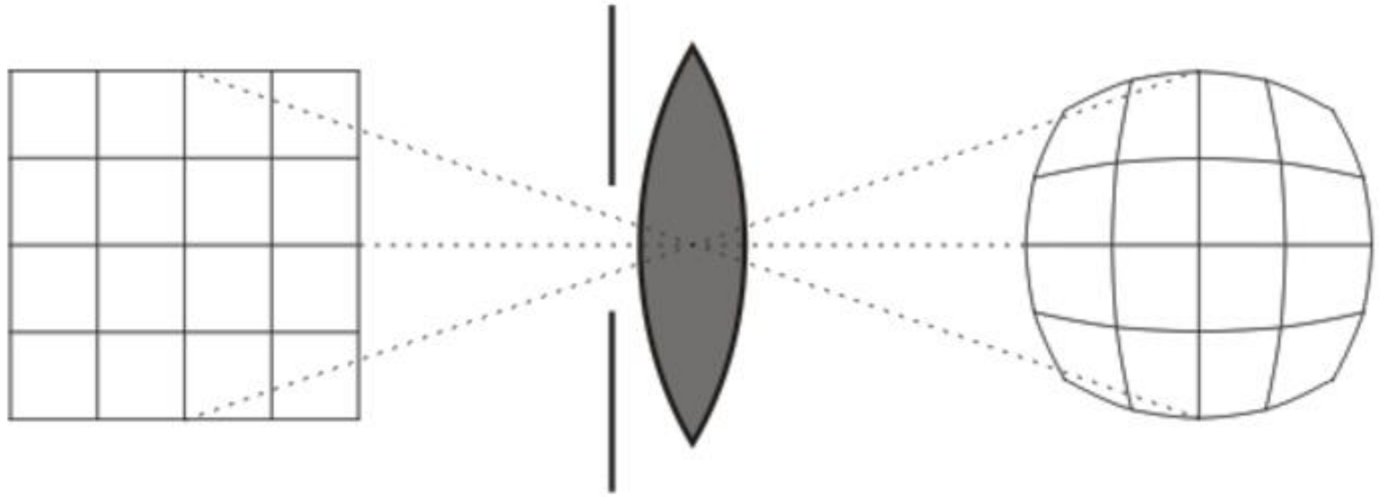
Here is a closer look at the lens distortion of a typical head mounted display. The lenses cause a pincushion effect:



Three approaches to VR lens distortion

IEEE P3333.3

The solution is to apply barrel distortion to the image. When we look at it through the distorting lenses, the image looks neutral:

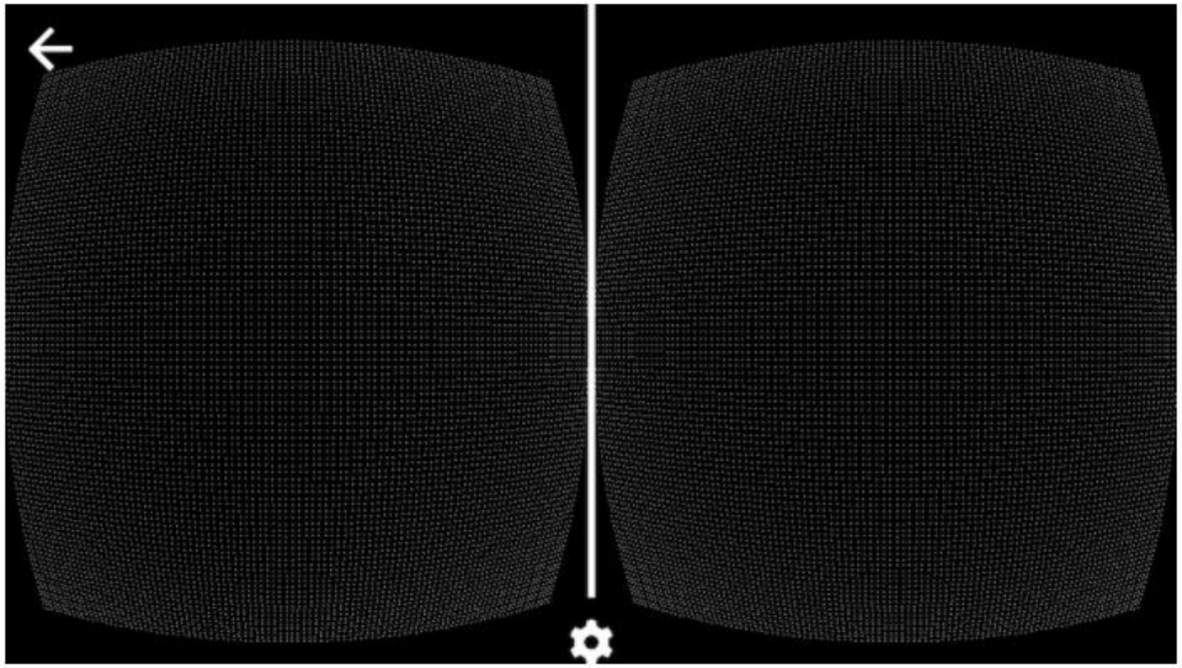


Lens distortion is well understood mathematically, governed by equations like these, with distortion coefficients corresponding to the particular lens. To undo the distortion properly, we also need to calculate the centers of the eyes, which requires knowing a bit about the geometry of the display and the enclosure itself. This can all be done, even on the web! I summarize a few implementation options below.

Fragment based solution (bad)

IEEE P3333.3

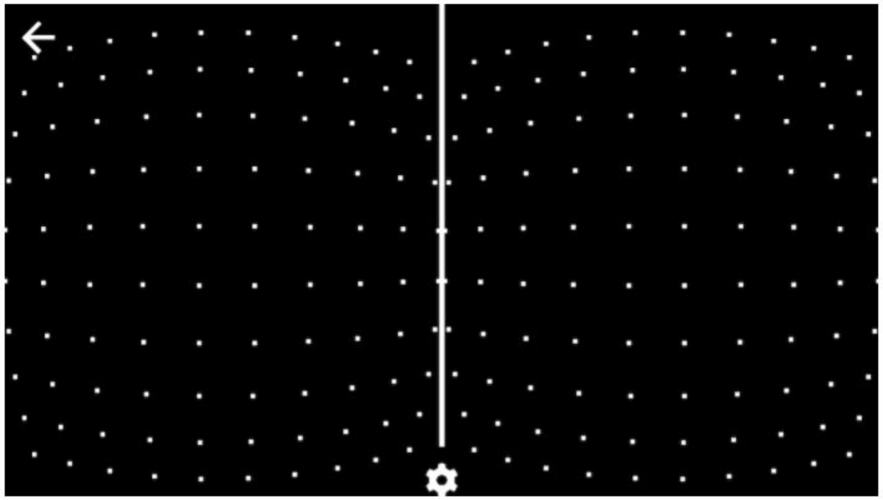
The simplest way to using two pass rendering. First, we render the left and right eyes onto a texture, and then process that texture with a fragment (pixel) shader, moving each pixel inward in relation to the centroid of the eye:



This is the first and simplest method, which is also the least efficient, since each pixel is processed separately. The first version of the WebVR Boilerplate implemented this method.

Mesh based solution (better)

Rather than processing each pixel separately, we distort the vertices of a relatively sparse mesh (40x20 works well).



This can save some direct computation and let the GPU do a fair amount of interpolation. Rather than having to apply to every single pixel (1920 * 1080 ~= 2e6), we do the calculation for every vertex in the mesh (40 * 20 = 800). The result is a significant reduction (3 magnitudes or so) of computation, and a nice boost in performance. The WebVR Polyfill currently implements this approach.

Applying distortion isn't the only expensive part in this rendering method. A lot of time is wasted copying the whole scene to an intermediate texture.

Vertex displacement based solution (best)

This brings us to the most efficient method of the three, which eliminates the need to render to an intermediate texture in the first place. In this approach, the geometry itself is distorted using a custom vertex shader. The idea is that knowing the position of the camera, we can displace vertices in such a way that the resulting 2D render is already barrel distorted. In this case, no shader pass is needed, and we save the expensive step of copying the rendering into a texture.

This method does require a certain vertex density on every mesh that is being deformed. Imagine the simple case of a large, 4-vertex rectangle being rendered very close to the camera. Distorting these vertices would still yield a 4-vertex flat rectangle, and clearly there's no barreling effect. Because of this, this is method does not generalize without extra work on the developer's part.

Vertex displacement based solution (best)

This approach is used in the Cardboard Design Lab and in the open sourced VR View project. Geometry-based distortion can also result in sharper looking renderings, since the two pass approach can cause aliasing, especially if the intermediate texture is small. You can read more about this distortion method in this helpful explainer.

Thank You