Real-Time Omnidirectional Stereo Rendering:
Generating 360° Surround-View Panoramic Images for Comfortable Immersive Viewing

Thomas Marrinan and Michael E. Papka

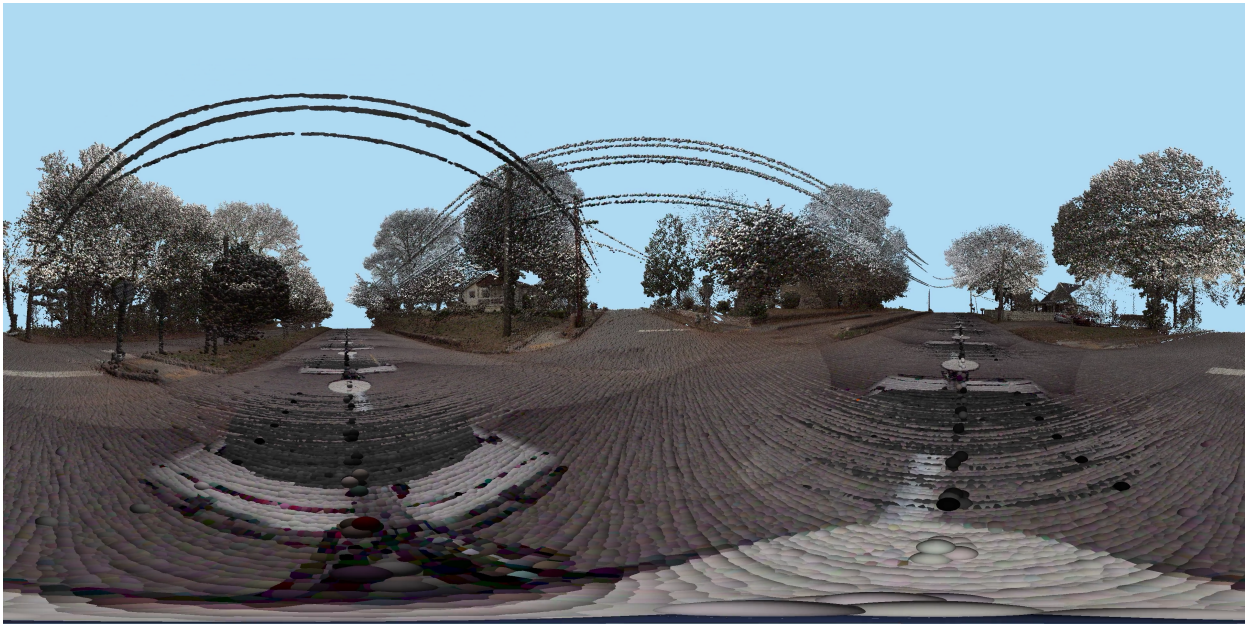


Fig. 1. Surround-view panoramic rendering of a LiDAR data set with over 5 million points using an equirectangular projection.

Abstract— Surround-view panoramic images and videos have become a popular form of media for interactive viewing on mobile devices and virtual reality headsets. Viewing such media provides a sense of immersion by allowing users to control their view direction and experience an entire environment. When using a virtual reality headset, the level of immersion can be improved by leveraging stereoscopic capabilities. Stereoscopic images are generated in pairs, one for the left eye and one for the right eye, and result in providing an important depth cue for the human visual system. For computer generated imagery, rendering proper stereo pairs is well known for a fixed view. However, it is much more difficult to create omnidirectional stereo pairs for a surround-view projection that work well when looking in any direction. One major drawback of traditional omnidirectional stereo images is that they suffer from binocular misalignment in the peripheral vision as a user's view direction approaches the zenith / nadir (north / south pole) of the projection sphere. This paper presents a real-time geometry-based approach for omnidirectional stereo rendering that fits into the standard rendering pipeline. Our approach includes tunable parameters that enable *pole merging* – a reduction in the stereo effect near the poles that can minimize binocular misalignment. Results from a user study indicate that pole merging reduces visual fatigue and discomfort associated with binocular misalignment without inhibiting depth perception.

Index Terms—Omnidirectional stereo, 360° panorama, pole merging, real-time rendering, virtual reality.

1 Introduction

Head Mounted Displays (HMDs) enable users to fully immerse themselves in a virtual environment. A user can change their view by simply moving or rotating their head. While traditional virtual reality (VR) applications allow users 6 degrees of freedom (DoF) to fully explore their environment, they require real-time view-dependent rendering. 360° surround-view panoramas, on the other hand, are view-independent – a singular image captures all possible viewing angles. This enables users to be immersed in content that has been generated without relying on user-specific view information. While 360° surround-view panoramas are limited to only 3 DoF (with the location being fixed), they still provide immersive and enriching experiences for users [7].

HMDs contain separate displays for the left and right eyes enabling VR content to provide stereo depth cues. Since stereoscopic three-dimensional (S3D) visualizations have been shown to help with tasks such as depth estimation and mental rotations of three-dimensional objects [10], it becomes important to incorporate stereoscopy into 360° surround-view panoramas. For fixed view directions with standard planar projections, rendering stereo pairs is well known [12]. However, for an equirectangular projection (spherical projection used for 360° surround-view panoramas), challenges remain on how to best create an omnidirectional stereo (ODS) image – a 360° image that maintains proper stereo regardless of a user's view direction.

The main challenge with creating ODS images stems from the fact that stereo depth comes from fusing two slightly different views of a scene – one from each eye. Since 360° surround-view panoramas

- *Thomas Marrinan is with the University of St. Thomas (also affiliated with Argonne National Laboratory). E-mail: tmarrinan@stthomas.edu.*
- *Michael E. Papka is with Argonne National Laboratory (also affiliated with Northern Illinois University). E-mail: papka@anl.gov.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

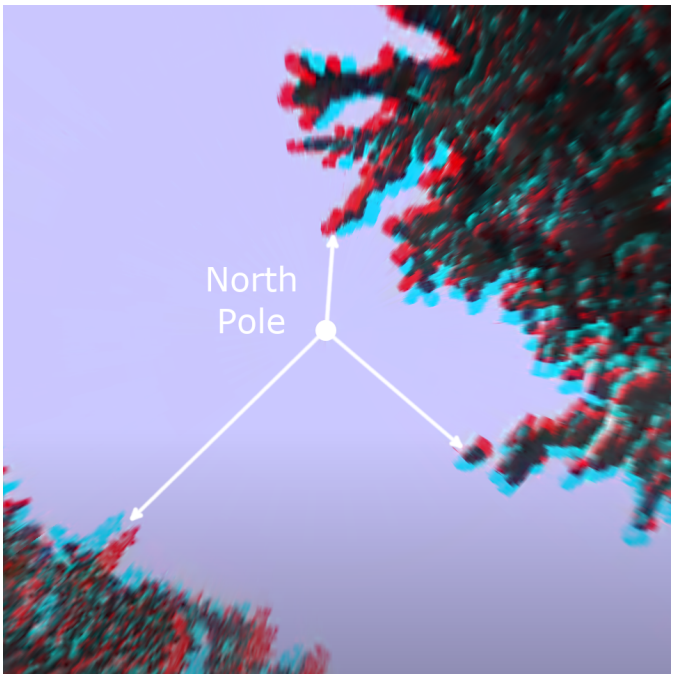


Fig. 2. Anaglyph S3D rendering that shows the binocular misalignment that occurs when viewing an ODS image near its north or south pole. Note, for proper view-dependent anaglyph stereo, cyan should be on the left and red on the right for dark objects.

capture all possible viewing angles, the locations of the left and right eyes in an ODS image are not fixed. In order to generate omnidirectional stereo pairs, each item is rendered by assuming the user is facing it (i.e. rotating the virtual cameras for the left and right eyes to face that item). While this accomplishes decent stereo in most situations, it is not perfect. When a user views such an image in VR, the stereo is only correct at the center of the image. Imperfections will occur in the peripheral vision since those items were rendered assuming the user was directly facing them. Such imperfections are negligible when looking horizontally (along the equator of the image), but become more pronounced as the user's view approaches looking straight up or straight down (near the north or south pole of the image) [11]. In fact, this results in binocular misalignment near the poles, where a user cannot fuse the images presented to the left and right eyes into a single object, and can lead to high levels of visual discomfort [9]. Fig. 2 is a snapshot of an ODS image rendered in anaglyph stereo that highlights the issues that occur when looking nearly straight up or straight down.

One approach to reduce binocular misalignment in ODS images is *pole merging* – a reduction in the interocular distance near the poles – which has been implemented in commercial rendering software such as Blender [4] and V-Ray [8]. Pole merging results in a gradual reduction in the stereo effect near poles. Utilizing pole merging therefore has a trade-off between perceived 3D-ness of objects in the focal region and binocular misalignment in the peripheral vision. Maintaining correct stereo in the focal region leads to large binocular misalignment in the peripheral vision near the poles, while reducing binocular misalignment in the peripheral vision leads to a loss of stereo depth in the focal region near the poles.

In this paper we present a real-time geometry-based approach for generating 360° surround-view panoramas (such as the one shown in Fig. 1) that fits into the standard rendering pipeline. Our approach includes dynamic tessellation to ensure sufficiently dense meshes, equirectangular projection of mesh vertices, and handling cases when there is a discontinuity in the projection (e.g. when a triangle crosses the back of the projection sphere and therefore wraps from the left to the right of the final image). We then detail how our algorithm is extended for creating ODS images with tunable parameters that control pole merging. Finally, we present performance results of our dynamic tessellation algorithm and results from a user study on depth perception and visual comfort to gain insight on the effects of pole merging.

2 Related Work

In order to develop a real-time technique for rendering comfortable ODS images, we investigated prior work primarily from two fields of research – real-time rendering with non-planar projections and stereoscopic depth perception. Finally, we discuss how our work differs from prior work done on ODS images.

2.1 Real-time Non-planar Projections

Traditionally, non-planar projections (such as those for cylindrical or spherical panoramas) have used an image-based approach [21]. In this type of approach, a standard cubemap of the scene is dynamically rendered in a first pass. A second pass is then used to perform the non-planar projection by sampling the cubemap at the appropriate location for each projected pixel. While image-based techniques typically work well, there are two situations where artifacts occur. First, in regions that become magnified / stretched in the final non-planar projection, the resolution is insufficient in the cubemap. Therefore, these locations can look pixelated or blurry. This occurs near the poles in the equirectangular projection used for surround-view panoramas. Second, artifacts can occur at the seams of the cubemap when attempting to render ODS images. This occurs since the two virtual cameras must be rotated to face each side of the cubemap, resulting in gaps or overlaps in the rendered geometry.

Due to the artifacts that can occur using image-based projections, Trapp et al. investigated a geometry-based approach for directly performing non-planar projections on model vertices [19]. The main issue with directly projecting vertices is that straight edges may need to be rendered as curves when using non-planar projections. Since an edge between two vertices is straight, the authors used dynamic mesh refinement, as proposed by Lorenz and Döllner [15], to iteratively break a single triangle into many smaller triangles using geometry shaders with transform feedback. This enables what used to be a single straight edge to approximate the correct curve in the final projection. The authors show that not only can this technique lead to higher fidelity images, but it also outperforms the image-based technique since the rendering is done in a single pass. However, this paper does not mention how they handle discontinuities in the final projection.

While the work by Trapp et al. used geometry shaders with transform feedback for iterative tessellation, Petkov et al. leverage the tessellation shader available on newer versions of graphics libraries to sufficiently refine meshes for non-planar projections [16]. However, this work also ignores discontinuities – explicitly stating that triangles that span a discontinuity are simply not rendered. Ardouin et al. also utilize tessellation shaders for geometry-based non-planar projections, but do propose a technique for handling discontinuities [1]. They use the tessellation shader to sufficiently refine a mesh, and then use a geometry shader to manually subdivide refined triangles at the discontinuity location (slightly shifting vertex locations to ensure new triangles no longer cross a discontinuity).

The three approaches mentioned above all appear to result in negligible artifacts by sufficiently refining meshes so that no projected triangle edge is too long. However, it was difficult to fully reproduce any of the three techniques, since the exact parameters for tessellation were not presented. We therefore detail our dynamic tessellation algorithm and present performance results comparing it to various static tessellation levels.

2.2 Stereoscopy

It is well known that stereoscopic vision enhances performance in depth perception tasks [6]. While a number of depth cues are captured in monoscopic photos, binocular disparity in stereo photography can provide an important additional depth cue. Lee et al. even demonstrate that stereoscopy improves depth perception with non-photorealistic visualization [14]. Therefore, S3D can become a valuable affordance for computer generated imagery regardless of its degree of realism.

However, there can be some negative side-effects when viewing S3D content. When left / right stereo pairs are not perfectly aligned, binoculuar misalignment will occur. Tyler et al. [20] show that vertical and torsional misalignment can result in visual discomfort and an inability to fuse the two images together. Horizontal misalignment, on the other hand, can easily be compensated for with a slight increase or decrease in eye convergence while looking at the scene, and therefore was not found to cause any discomfort. Since ODS images do not create perfectly accurate stereo, noticeable vertical and torsional misalignment can occur in the peripheral vision.

Binocular misalignment in ODS images is most pronounced near the zenith / nadir (north / south pole) of the projection sphere. While pole merging can be used to reduce the misalignment between the two images, it does so by gradually reducing the interocular distance to 0 near the poles. In theory, this would negatively affect the stereo depth perception of a user viewing such an image. However Rosenberg [17] conducted a study on depth perception as it relates to interocular distance in S3D images and found no decrease in depth estimation, even when the interocular distance was reduced just below half its actual value. Therefore, pole merging should not degrade the stereo depth for the vast majority of the ODS image, including much of the transitional region where the interocular distance is being reduced.

2.3 Omnidirectional Stereo

While there is a significant amount of prior work on image alignment for ODS photography [5, 13], the work that focuses on rendering computer generated ODS images is more limited in scope. Simon and Beckhaus developed a method for rendering cylindrical ODS images in real-time by discretizing the projection cylinder into a faceted one [18]. A scene could be rendered with the two virtual cameras pointing towards each face of the faceted cylinder. Stitching together the renderings of each face results in the final cylindrical ODS image. With this approach, some artifacts occur since the projection cylinder has been approximated – the more facets that are used, the better the approximation, but more computationally intense it becomes to render.

In order to capture a full 360° surround-view panorama, the techniques already described in Sect. 2.1 can be deployed for spherical ODS as well. When projecting each vertex of the refined mesh, the virtual camera can be translated left / right with respect to the direction to that vertex. Doing so will result in creating a traditional ODS image, with negligible stereo artifacts near the equator, but potentially significant stereo artifacts near the poles.

If real-time rendering is not necessary, the use of commercial ray tracing software is commonplace. In order to create 360° surround-view panoramas using ray tracing, renderers can simply map each pixel to its location on the projection sphere, then cast a ray from the virtual camera through that point and into the scene [11]. For ODS images, the virtual camera can be translated left / right with respect to the direction to each pixel. This method alone will also result in a traditional ODS image. In order to address the visual discomfort that is associated with the stereo artifacts near the poles, many of the commercial ray tracing packages enable pole merging [4, 8]. Pole merging is accomplished by gradually reducing the interocular distance near the poles. For ray tracing renderers, this simply means using a different interocular distance per row of pixels in the final projected image.

Our work is the first to our knowledge that not only proposes an implementation of pole merging for real-time ODS rendering, but also evaluates the effects of pole merging on visual comfort and depth perception of authentic VR users.

3 Geometry-based Equirectangular Projection

360° surround-view panoramas are created using an equirectangular projection that has a 2:1 aspect ratio. This is performed by directly mapping longitude and latitude from the surface of the projection sphere to x and y coordinates of the final image. For geometry-based projections, mesh vertices are projected onto a unit sphere. Equation 1 is used to determine the direction from the camera to vertex v. This is in turn used in Equation 2 and Equation 3 to calculate the longitude and latitude of the projected vertex respectively. The result will have a longitude value in the range $[-180°, 180°]$ and a latitude value in the range $[-90°, 90°]$.

$$v_{dir} = v_{pos} - camera_{pos} \tag{1}$$

$$longitude = \begin{cases} -90° & v_{dir_z} = 0 \text{ and } v_{dir_x} \geq 0 \\ 90° & v_{dir_z} = 0 \text{ and } v_{dir_x} < 0 \\ -\arctan2\left(v_{dir_x}, v_{dir_z}\right) & v_{dir_z} > 0 \end{cases} \tag{2}$$

$$latitude = \arcsin\left(\frac{v_{dir_y}}{\|v_{dir}\|}\right) \tag{3}$$

After calculating a vertex's longitude and latitude coordinates on the projection sphere, the resulting point (with $x = longitude$, $y = latitude$, and $z = -\|v_{dir}\|$) can be multiplied by a standard orthographic projection matrix (with $left = -180°$, $right = 180°$, $bottom = -90°$, and $top = 90°$) to transform the vertex into the canonical view volume. The *near* and far values for the orthographic projection matrix can be modified to meet the needs of each application.

3.1 Dynamic Tessellation

While performing an equirectangular projection directly on input mesh vertices is fairly straightforward, it often results in an image that contains many artifacts. This is due to the fact that straight edges in the mesh may need to be represented with curves in the 360° surround-view panorama. If mesh edges are sufficiently short in the projected image, the resulting artifacts become negligible. Therefore, using tessellation to refine a course mesh will reduce artifacts and can lead to a high-quality 360° surround-view panorama.

Since the length of an edge in the projected image depends not only on its physical size but also on its distance away from the camera, tessellation cannot occur a priori. Therefore, run-time tessellation approaches must be used to ensure sufficient mesh refinement given the location of the mesh with respect to the virtual camera.

We have implemented dynamic run-time tessellation of triangular meshes in OpenGL 4.1, with the use of the tessellation control shader (TCS) and tessellation evaluation shader (TES). The TCS is used to determine outer tessellation levels (how many segments each edge of

Algorithm 1 Dynamic tessellation levels for equirectangular projection

Input: $v_{pos}[3]$
Output: $tess\_level_{outer}[3]$, $tess\_level_{inner}$

for $i \leftarrow 0$ **to** 3 **do**
  $projected[i] = \text{equirectangular}(v_{pos}[i])$
  $opp\_midpoint = (v_{pos}[(i+1)\%3] + v_{pos}[(i+2)\%3])/2$
  $mid\_projected[i] = \text{equirectangular}(opp\_midpoint)$
for $i \leftarrow 0$ **to** 3 **do**
  $opp_a = (i+1)\%3$
  $opp_b = (i+2)\%3$
  $\Delta lon = \text{lonDist}(projected[opp_a].lon, projected[opp_b].lon)$
  $\Delta lat = \max($
    $\text{abs}(projected[opp_a].lat - projected[opp_b].lat)$,
    $\text{abs}(projected[opp_a].lat - mid\_projected[i].lat)$,
    $\text{abs}(projected[opp_b].lat - mid\_projected[i].lat)$
  $)$
  $max\_lat = \max($
    $\text{abs}(projected[opp_a].lat)$,
    $\text{abs}(projected[opp_b].lat)$,
    $\text{abs}(mid\_projected[i].lat)$
  $)$
  $edge = \text{vec2}(\Delta lon, 0.5 \cdot \Delta lat)$
  $d = \max(0.125 * \|edge\|, 1.0)$
  $pole\_multiplier = \max((2.8284/90.0) \cdot max\_lat, 1.0)$
  $tess\_level_{outer}[i] = \min(pole\_multiplier \cdot d, 64.0)$
$max\_tess = \max(tess\_level_{outer})$
$tess\_level_{inner} = \max(max\_tess - 2.0, \min(max\_tess, 3.0))$

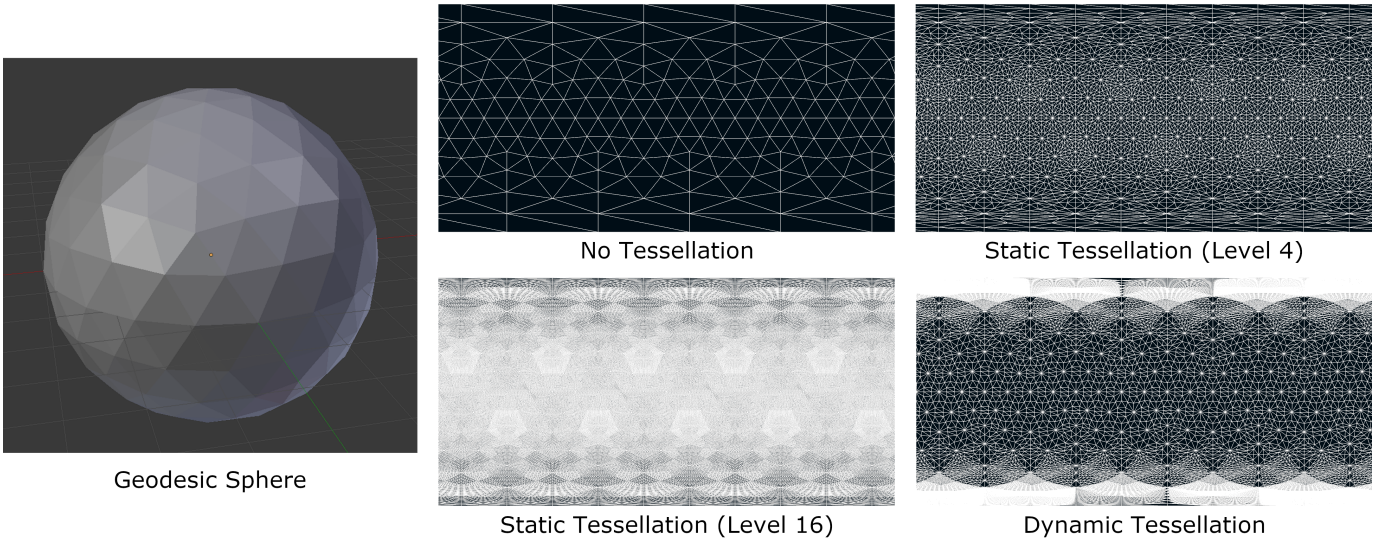


Fig. 3. Geometry-based equirectangular projection of a geodesic sphere from the inside-out (i.e. camera is at the center of the sphere) to highlight the effects of tessellation. With no tessellation, the mesh is not sufficiently dense to avoid noticeable artifacts from a non-planar projection. Note too that even though all triangles in the geodesic sphere are the same size, they project to a larger area near the poles (top and bottom of the image). Using static tessellation where each edge of a triangle has 4 inner and outer subdivision levels, the mesh is sufficiently dense near the equator but not near the poles. Using static tessellation where each edge of a triangle has 16 inner and outer subdivision levels, the mesh is sufficiently dense near the poles but unnecessarily over-dense near the equator. With our dynamic tessellation algorithm, the mesh has become sufficiently dense near the poles without becoming over-dense near the equator.

a mesh triangle should be divided into), as well as the inner tessellation level (how many interior rings to add). The TES is used to set the positions and other attributes for all newly created vertices. Algorithm 1 outlines our approach in the TCS for calculating outer and inner tessellation levels to result in negligible artifacts when using an equirectangular projection. Vertex positions and other attributes can directly be calculated using the barycentric coordinates provided by the TES since the goal is simply to shorten projected triangle edge lengths, not to modify the geometry in any other way.

The goal of our algorithm is to refine the input geometry and reduce artifacts from performing a non-planar projection to a negligible level. In an equirectangular projection, a straight edge may need to be rendered as a curve if there is a change in longitude between the endpoints. The greater the longitudinal distance[1] between the two endpoints, the more segments the edge will likely need to approximate the curve. While changes in latitude will not introduce curves in an equirectangular projected image, it will cause issues with attributes such as normals and texture coordinates since the projection is non-linear. Therefore, changes in latitude are still taken into consideration, but with half the weight of changes in longitude. Due to the potential curvature of a projected edge, a central point may be located at a higher or lower latitude than either endpoint. We therefore also check the latitude of the midpoint of an edge when calculating the change in latitude. While not perfect, this method provides a decent approximation while remaining computationally efficient.

Also worth noting is that the non-linear effects of both changes in longitude and latitude are magnified the further away from the equator the edge is. Therefore, we introduce a pole multiplier to increase the tessellation levels as the edge gets near to the north or south pole regions. Fig. 3 shows an equirectangular projection of a geodesic sphere mesh with the camera at its center. Note that our dynamic tessellation algorithm successfully results in high tessellation near the pole regions without over tessellating the majority of the mesh.

The number of inner tessellation levels is set based on the maximum outer tessellation level for any triangle edge. Equation 4 shows the piecewise function used to calculate the inner tessellation level. If any edge requires significant tessellation (more than 5 segments), then the inner tessellation will be set to the maximum tessellation level minus 2. If all edges can remain a single segment or only need to be split up into two segments, then the inner tessellation will be set to equal the maximum tessellation level. If neither cases are true, the inner tessellation level is capped at 3 to ensure sufficient inner tessellation when an edge requires a medium number of segments (between 3 and 5).

$$tess\_level_{inner} = \begin{cases} max\_tess - 2 & max\_tess > 5 \\ 3 & 3 \leq max\_tess \leq 5 \\ max\_tess & max\_tess < 3 \end{cases} \quad (4)$$

3.2 Discontinuities

There are two situations where simply projecting the vertices of a triangle using Equations 1-3, will cause issues. The first is when the projected triangle crosses the back of the projection sphere. The second is when the projected triangle overlaps the north or south pole. Issues will occur in either case because the triangle overlaps a discontinuity in the projection – a continuous area in three-dimensional world space that projects to more than one non-continuous area in two-dimensional image space. Take, for example, a triangle whose vertices project to $(-170°, 45°)$, $(-170°, 35°)$, and $(165°, 40°)$. In this situation, half of the triangle should be drawn on the left of the image and the other half on the right, but if the triangle is rendered by simply connecting the projected locations of the vertices, it will instead result in a single triangle that covers nearly the whole width of the image.

Since multiple triangles may need to be drawn on the projected image in order to represent a single input triangle from the refined mesh, we have implemented the equirectangular projection and discontinuity handling using geometry shaders in OpenGL 4.1. Geometry shaders have the ability to take an input primitive and output 0 or more primitives in its place. Our real-time 360° surround-view panorama geometry shader takes refined triangles from the tessellation shaders in three-dimensional world space as input and outputs between 1 and 10 triangles in two-dimensional image space.

[1] Change in longitude between two points, taking into account that $-180°$ and $180°$ are the same location. Example: $\text{lonDist}(-170°, 165°) = 25°$.

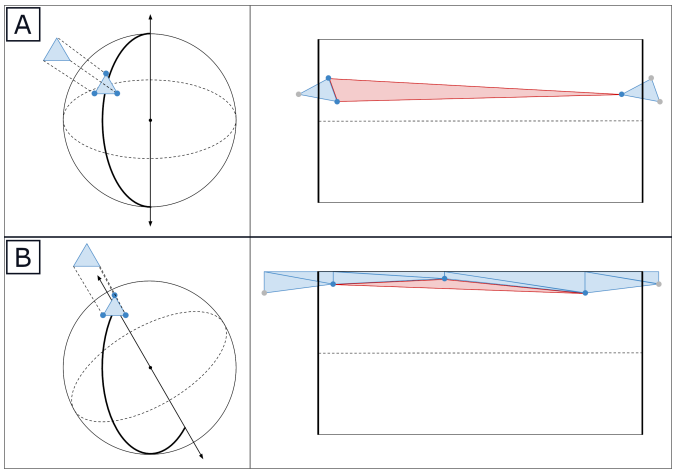


Fig. 4. Discontinuity handling with the left part of each panel depicting the three-dimensional geometry with the projection sphere, and the right part of the panel depicting the two-dimensional equirectangular image. Panel A shows a projected triangle that crosses the back of the projection sphere, and our solution for properly handling this type discontinuity. Panel B shows a projected triangle that covers the north pole of the projection sphere, and our solution for properly handling this type of discontinuity. Both panels also show the incorrect result (red triangle) if discontinuities are not handled and projected vertices are simply connected.

3.2.1 Discontinuity at Back of Projection Sphere

The longitudinal distance between two projected points cannot be greater than 180°. Therefore, if the difference between the minimum projected longitude and maximum projected longitude for the three vertices of a triangle is greater than 180°, then the triangle must cross the back of the projection sphere (where $-180°$ is the same as 180°). In such cases, part of the triangle should be rendered on the left side of the image and the other part of the triangle should be rendered on the right side. Rather than calculating where the discontinuity intersects the triangle, as done by Ardouin et al. [1], we propose to simply draw the whole triangle twice (once on the left side and once on the right side), letting hardware perform its standard clipping to remove unnecessary geometry. To first render the triangle on the left side, we subtract 360° from any vertex's longitude who current value is greater than 0, thus moving it outside the left edge of the image. Then, to render the triangle on the right side, we add 360° to the longitude of all vertices, thus moving vertices moved for the first triangle back to their original location and moving the rest outside the right edge of the image. Fig. 4 Panel A illustrates this process.

3.2.2 Discontinuity at North and South Pole

A discontinuity exists at the poles (where latitude = $-90°$ or 90°) since all possible longitude values correspond to the same point on the projection sphere, but different locations in the equirectangular image. In order to determine whether or not a triangle covers the north or south pole region, barycentric coordinates are calculated for the origin with the two-dimensional triangle created by using only the xz components of each vertex's position in relation to the virtual camera position. If all barycentric weights are greater than or equal to 0, then the triangle does cover one of the two poles. There are three distinct ways that a triangle could be covering a pole: 1) the pole could align exactly with one of the triangle's vertices, 2) the pole could fall exactly on an edge between two of the triangle's vertices, or 3) the pole could fall in the middle of all the three of the triangle's vertices.

In case 1, where the pole aligns exactly with one vertex, the first step is to determine the two vertices that do not project onto a pole (vertices with barycentric weights greater than 0) – call these vertices a and b. A triangle strip with points $(a_{lon}, 90°)$, (a_{lon}, a_{lat}), $(b_{lon}, 90°)$, (b_{lon}, b_{lat}) is used to create two triangles connecting a and b straight up to the top of the image (or down to the bottom with pole latitude $-90°$). Note that attributes, such as normals and texture coordinates, from the vertex that projects to the pole should be used for both vertices at the top/bottom of the image.

In case 2, where the pole falls exactly on an edge, the first step is to calculate the position and all other vertex attributes for the pole intersection location, which can be done by using the barycentric weights with traditional three-dimensional linear interpolation. The next step is to determine the the vertex opposite of the edge that projects onto a pole (vertex with barycentric weight equal to 0) – call this vertex opp, and the other two a and b. A triangle strip with points $(a_{lon}, 90°)$, (a_{lon}, a_{lat}), $(opp_{lon}, 90°)$, (opp_{lon}, opp_{lat}), $(b_{lon}, 90°)$, (b_{lon}, b_{lat}) is used to create four triangles connecting a, opp, and b straight up to the top of the image (or down to the bottom with pole latitude $-90°$).

In case 3, where the pole falls in the middle of the triangle, again the first step is to calculate the position and all other vertex attributes for the pole intersection location using the barycentric weights with traditional three-dimensional linear interpolation. The next step is to order the triangle vertices such that they are in ascending longitudinal order (using longitudes in the range $[-360°, 0°]$) – call these sorted vertices a, b, and c respectively. A triangle strip with points $(a_{lon}, 90°)$, (a_{lon}, a_{lat}), $(b_{lon}, 90°)$, (b_{lon}, b_{lat}), $(c_{lon}, 90°)$, (c_{lon}, c_{lat}) is used to create four triangles connecting a, b, and c straight up to the top of the image (or down to the bottom with pole latitude $-90°$).

Note, in all three cases the triangle may also cross the back of the projection sphere. In this situation, both fixes are applied. For case 3, it is guaranteed to cross the back of the projection sphere since the triangle covers all longitude values. Therefore, there should be one long triangle strip with 10 triangles, rather than separate strips on each side of the image. Fig. 4 Panel B illustrates this process for case 3 (only 8 triangles shown since the final two get completely clipped).

4 Omnidirectional Stereo

ODS images are made from two 360° surround-view panoramas, representing the views from each eye. For traditional S3D rendering, the virtual camera is simply moved left/right half the desired interocular distance and uses an off-axis view frustum. However, since left and right are not fixed directions in a 360° surround-view panorama, this approach does not work. Instead, left and right directions can uniquely be determined for each vertex in the scene's meshes. The assumption is made that the view direction is exactly facing the vertex and the positive y-axis is up. Using these two assumptions, the view right direction for a particular vertex can be calculated using Equation 5.

$$\begin{aligned} forward &= vertex - camera \\ up &= \text{vec3}(0,1,0) \\ right &= \text{normalize}(forward \times up) \end{aligned} \tag{5}$$

The $right$ vector can then be scaled by $(-0.5 \cdot interocular\_distance)$ to get the camera offset for the left eye, or $(0.5 \cdot interocular\_distance)$ to get the camera offset for the right eye. This camera offset is used to translate the camera's location prior to performing the equirectangular projection. Using our approach, this process takes place in the geometry shader for each tessellated vertex.

4.1 Pole Merging

As previously stated, creating traditional ODS images will result in non-perfect stereo for the peripheral vision. When looking nearly straight up or straight down, these imperfections can cause noticeable binocular misalignment resulting in visual discomfort. Therefore we have implemented pole merging to gradually reduce the interocular distance near the pole regions. Our pole merging technique contains a number of tunable parameters to control the angle of inclination where the interocular distance begins to reduce and how quickly it will go to 0. While pole merging will reduce the stereo depth cues available to viewers in part of the ODS image, we argue that it does not negatively affect the overall S3D experience since the stereo is also imperfect without pole merging.

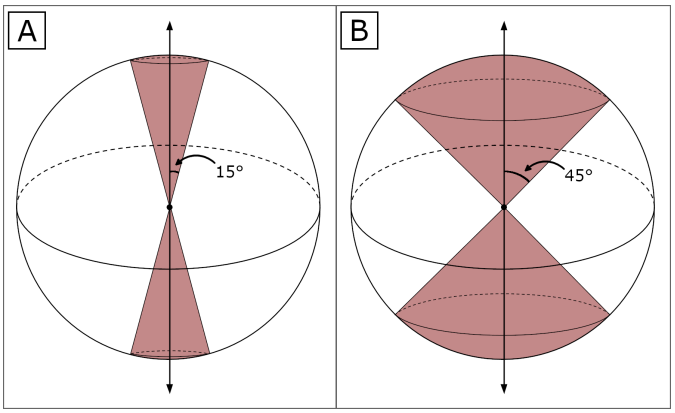


Fig. 5. Diagram of the ODS projection sphere, highlighting the region where pole merging gradually takes place. Panel A shows pole merging that starts 15° away from each pole and ends at the pole. Panel B shows pole merging that starts 45° away from each pole and ends at the pole.

There are three parameters to control the severity of the pole merging: 1) start angle, 2) end angle, 3) interpolation exponent. The start and end angles specify the region in relation to the poles where the interocular distance is gradually reduces to 0. The interpolation exponent controls how quickly the interocular distance will reduce within the merging region. Fig. 5 illustrates basic pole merging regions starting at 15° away from each pole in Panel A, and 45° away from each pole in Panel B. The region of the projection sphere that is between the equator and the start angle will maintain regular interocular distance. The region of the projection sphere that is between the end angle and the pole will have an interocular distance of 0. All three parameters can be set independently for each pole. Equation 6 shows the formula used to calculate the adjusted interocular distance based on a vertex's projected latitude.

$$
\begin{aligned}
inclination &= \text{abs}\,(latitude) \\
merge_{start} &= 90^\circ - start\_angle \\
merge_{end} &= 90^\circ - end\_angle \\
a &= \left(1 - \text{clamp}\left(\frac{inclination - merge_{start}}{merge_{end} - merge_{start}}, 0, 1\right)\right)^{exp} \\
adjusted\_dist &= a \cdot interocular\_dist
\end{aligned} \tag{6}
$$

One notable limitation of ODS images is that the interpupillary distance (IPD) must be determined at render time, and therefore cannot be customized for individual users. However, non-anatomically correct IPD has been shown to not affect depth perception or visual fatigue when using the adult mean value of 6.3cm [3].

5 Performance Evaluation

In order to evaluate our work and showcase its ability to handle differing rendering techniques, we developed two sample scenes. The first uses data from a LiDAR scan to render over 5 million points as imposter spheres (camera-facing rectangles shaded to look like spheres) with two point light sources. The second includes a number of three-dimensional OBJ models with textures made from a total of more than 620 thousand triangles, as well as a skybox for the background. This scene is illuminated by 18 point lights and 3 spot lights. Both scenes use Phong shading to calculate illumination per pixel.

To evaluate the performance of our dynamic tessellation algorithm, we compared its frame rate against using no tessellation and static tessellation with 4 and 16 segments for all triangles. All tests were run on a Linux workstation with an NVIDIA GeForce RTX 2080 GPU. Results are summarized in Table 1.

Table 1. Frames per second for rendering a 3840×1920 360° surround-view panorama for the LiDAR scene and the OBJ model scene.

	No Tess.	**Static (4)**	**Static (16)**	**Dynamic**
LiDAR	32.5 fps	2.7 fps	0.2 fps	25.5 fps
OBJ Models	154.0 fps	38.2 fps	4.7 fps	145.0 fps

Our dynamic tessellation algorithm resulted in a nominal decrease in frame rate as compared to not performing any tessellation whatsoever. The dynamic tessellation also significantly outperformed static tessellation using either 4 or 16 segments. Since our dynamic tessellation algorithm is based on projected edge length, only those triangles that cover a significant portion of the image will require tessellation. Therefore, many triangles (especially those far away from the camera) are likely to require little or no tessellation. However, we cannot skip the tessellation stage altogether, since there are other triangles that may cover a significant portion of the projected image which would result in noticeable artifacts without tessellation.

Also worth noting is that both scenes were able to achieve real-time frame rates (> 25 fps). Therefore our technique for generating 360° surround-view panoramas is well suited for integration into real-time rendering applications.

6 User Study

We designed and conducted a user study to evaluate the effects of pole merging on visual comfort and stereo depth perception when viewing ODS images in a VR headset. We developed a WebXR application using Babylon.js [2] to view ODS images so that it would run on any HMD with a compatible browser. Although the three parameters that control pole merging exist in a continuous domain, we selected a discrete set for users to compare. Images were generated with the pole merge start angle varying from 0° to 90° (where 0° equates to no pole merging, and 90° begins the pole merging at the equator) in 15° increments. For each start angle, images were generated using linear and cubic interpolation for the adjusted interocular distance (exponent equal to 1 and 3 respectively). The pole merge end angle was held at 0° (at the pole) for all images.

Image sets were created for both the LiDAR point cloud and the OBJ model scenes. Participants viewed a total of 13 images for each scene. When the pole merge start angle was 0°, no pole merging took place. Therefore, this resulted in a traditional ODS image without pole merging for both the linear and cubic interpolation image sets. There were six other pole merge start angles tested (15°, 30°, 45°, 60°, 75°, and 90°) for each of the two interpolation exponents.

Our study was designed to qualitatively evaluate both visual comfort and stereo depth perception. The user study was broken into two parts. For part 1, participants compared each ODS image to the monoscopic

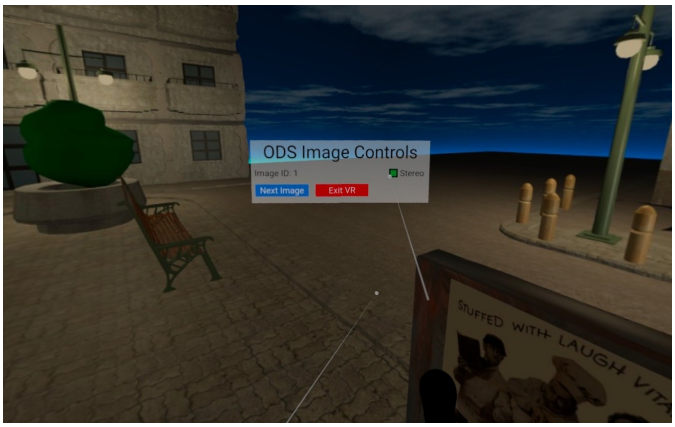


Fig. 6. Immersive study application for viewing ODS images. Participants switched between images using the virtual user interface (figure depicts interface used during part 1 of the study).

Fig. 7. Equirectangular projection of a nighttime scene in a town square with a car driving by. Green and red spheres were added as targets to direct participant view when evaluating visual comfort and stereo depth perception.

360° panorama individually. Participants were asked to rate their visual comfort and improvement in depth perception when viewing each ODS image. For part 2, participants were able to dynamically switch between all versions of the ODS image. They were then asked to select the image that was most comfortable while still maintaining quality stereo depth for both the linear and cubic sets. Fig. 6 shows a view from inside a VR headset, depicting how participants would view an ODS image and interact with a user interface to partake in the study. For both part 1 and part 2, bright colored spheres were added to each scene as targets to direct a participant's view to certain regions of interest.

In order to evaluate visual comfort when viewing areas near the poles, participants were instructed to center a particular target in the center of the HMD screen (blue sphere with a latitude of -73.5° for the LiDAR point cloud scene, and the moon on the skybox of the OBJ model scene which had a latitude of +74.0°). Participants were then directed to focus on other nearby targets (bright green spheres with similar latitudes but different longitudes) by moving only their eyes so as to keep the original target centered on the HMD screen. When focusing these green spheres located near the north or south pole that were displayed in the periphery of the HMD screen, participants were asked to rank their level of visual comfort on a 5-point scale (1: very uncomfortable, 2: uncomfortable, 3: neutral, 4: comfortable, 5: very comfortable).

In order to evaluate stereo depth perception, three red targets were placed in each scene near specific areas where stereo depth cues were particularly helpful for understanding the space (e.g. power lines that run in front of a tree in the LiDAR point cloud scene, and a nearby movie poster in the OBJ model scene). Participants were asked to dynamically toggle between the ODS and monoscopic versions of the 360° panorama and qualitatively rate their improvement in depth perception when viewing objects in the ODS image located near the red targets on a 4-point scale (1: no improvement, 2: slight improvement, 3: moderate improvement, 4: great improvement). Fig. 7 shows the OBJ model scene with the targets used for the study.

Twelve participants were recruited for the user study – 8 males and 4 females, with ages ranging from 23 - 65 (average age of 43.7). The study was conducted remotely, with each participant using their own HMD. Therefore VR hardware was not standardized for this study. 9 participants used an Oculus Quest, 1 used an HTC Vive Pro, 1 used an HP Windows Mixed Reality Headset, and 1 used a Google Daydream.

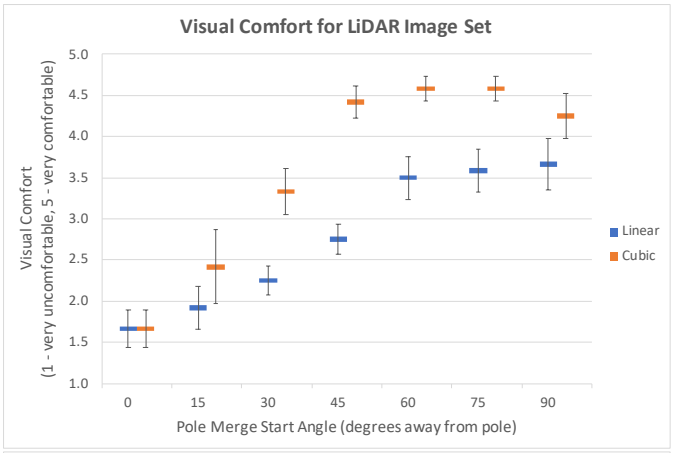
6.1 Visual Comfort Results

For each image that a participant viewed during part 1 of the study, they were asked to rate their level of visual fatigue / discomfort on a scale of 1-5 (with 1 being very uncomfortable and 5 being very comfortable). Results are illustrated in Fig. 8 and show a substantial improvement in participant visual comfort as the pole merge start angle increases using either linear or cubic interpolation for the adjusted interocular distance. We also note that there appears to be a point where user comfort plateaus, indicating that after some threshold there is no benefit from starting the pole merging further from the poles.

In the LiDAR point cloud scene, cubic interpolation of the adjusted interocular distance resulted in a more comfortable viewing experience than using linear interpolation. The plateau for cubic interpolation is reached at a value of approximately 4.5 (between comfortable and very comfortable), whereas the platuea for the linear interpolation is reached at a value of approximately 3.5 (between neutral and comfortable). Also the plateau using cubic interpolation was reached at a smaller pole merge start angle than the plateau when using linear interpolation. Both, however are vast improvements over the traditional ODS image, which had an average user comfort level of 1.7 (between uncomfortable and very uncomfortable).

In the OBJ model scene, cubic and linear interpolation of the adjusted interocular distance resulted in much more similar levels of visual comfort. Both plateaued at approximately 4.0 (comfortable), though the images using cubic interpolation reached that plateau at a lower pole merge start angle. Again, ODS images using both pole merge interpolation schemes resulted in improved visual comfort levels as compared to the traditional ODS image, which had an average value of 2.6 (between neutral and uncomfortable).

For each scene, we performed a single factor ANOVA test to compare the seven images that used linear interpolation to gradually reduce the interocular distance, and a second single factor ANOVA test to compare the seven images that used cubic interpolation to to gradually reduce the interocular distance. When viewing the LiDAR point cloud scene, there was a significant effect of pole merge start angle with linear interpolation on a participant's visual comfort at the $p < 0.05$ level for the seven conditions [$F(6,77) = 11.87$, $p = 2.18 \times 10^{-9}$]. There was also a significant effect of pole merge start angle with cubic interpolation on a participant's visual comfort at the $p < 0.05$ level for the seven conditions [$F(6,77) = 19.43$, $p = 1.16 \times 10^{-13}$].



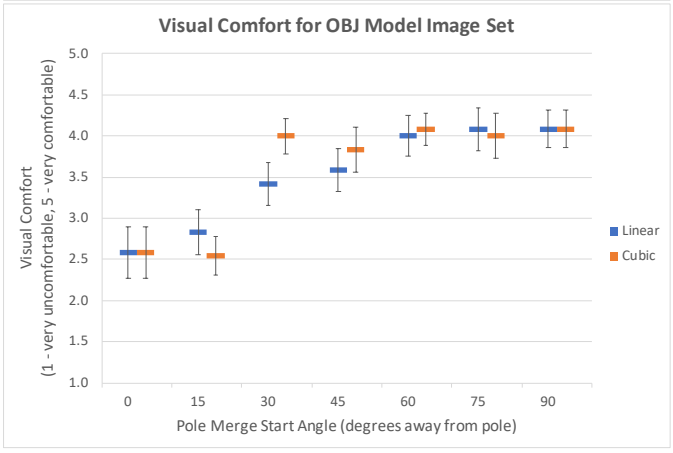
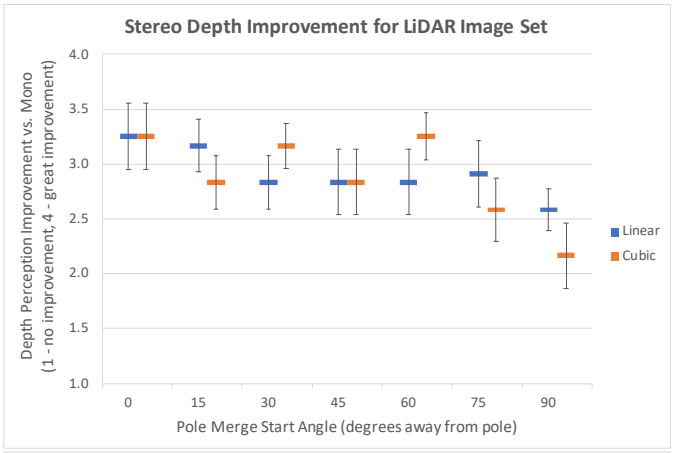


Fig. 8. User study results for visual comfort with various severities of pole merging. The use of pole merging showed a substantial improvement in visual comfort when viewing objects in the peripheral vision near the poles for both image sets.



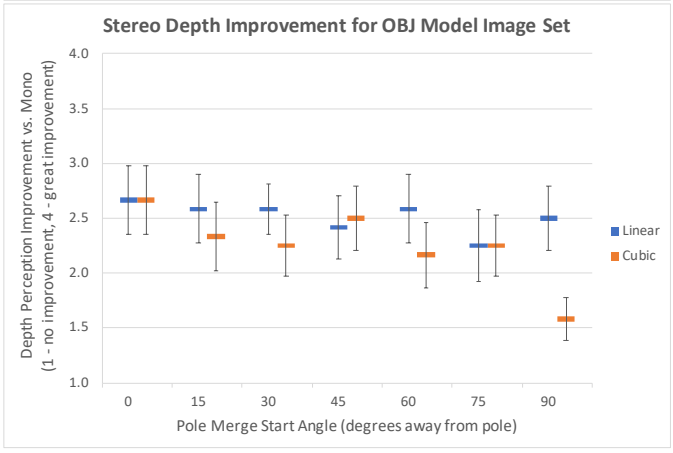


Fig. 9. User study results for stereo depth perception with various severities of pole merging. In all but the most extreme cases, the use of pole merging did not show a substantial decrease in stereo depth perception for both image sets.

Similarly, when viewing the scene with OBJ models, there was a significant effect of pole merge start angle with linear interpolation on a participant's visual comfort at the $p < 0.05$ level for the seven conditions $[F(6,77) = 5.34,\ p = 1.19 \times 10^{-4}]$. There was also a significant effect of pole merge start angle with cubic interpolation on a participant's visual comfort at the $p < 0.05$ level for the seven conditions $[F(6,76) = 7.70,\ p = 1.76 \times 10^{-6}]$. Note that one participant did not provide a visual comfort level response for the image using cubic interpolation with a pole merge start angle of 15°.

These results show that the use of pole merging can significantly improve a user's visual comfort levels when viewing ODS images in a VR headset. Comfortable viewing tended to occur with pole merge start angles beginning in the $[30°, 60°]$ range.

6.2 Stereo Depth Perception Results

For each image that a participant viewed during part 1 of the study, they were also asked to rate their improvement in depth perception when toggling between the ODS image and a monoscopic 360° panorama of the scene on a scale of 1-4 (with 1 being no improvement and 4 being great improvement). Results are illustrated in Fig. 9 and show similar levels of improved depth perception in all but the most extreme pole merge start angles.

In the LiDAR point cloud scene, improvement in stereo depth perception with linear interpolation of the adjusted interocular distance consistently scored and average value near 3.0 (moderate improvement). For pole merge start angles $[0°, 15°]$ the average ratings were slightly above 3.0, and for pole merge start angles $[30°, 90°]$ the average ratings were slightly below 3.0. When using cubic interpolation of the adjusted interocular distance, improvement in stereo depth perception fluctuated slightly as it also hovered around a value of 3.0 for pole merge start angles $[0°, 75°]$. At the most severe pole merge start angle (90°) however, the stereo depth improvement dropped to an average value of 2.2 (just above slight improvement).

In the OBJ model scene, improvement in stereo depth perception with linear interpolation of the adjusted interocular distance consistently scored an average value near 2.5 (between slight and moderate improvement). Only minor fluctuations existed between all pole merge start angles. When using cubic interpolation of the adjusted interocular distance, improvement in stereo depth perception hovered slightly lower – around a value of 2.3 for pole merge start angles $[0°, 75°]$. Again, a drop-off in stereo depth improvement was reported at the most severe pole merge start angle, where the stereo depth improvement scored an average value of 1.6 (between no improvement and slight improvement) when the start angle was 90°.

For each scene, we performed a single factor ANOVA test to compare the seven images that used linear interpolation to gradually reduce the interocular distance, and a second single factor ANOVA test to compare the seven images that used cubic interpolation to to gradually reduce the interocular distance. When viewing the LiDAR point cloud scene, there was no significant effect of pole merge start angle with linear interpolation on the improvement of a participant's stereo depth perception at the $p < 0.05$ level for the seven conditions $[F(6,77) = 0.74,\ p = 0.62]$. There was however a significant effect of pole merge start angle with cubic interpolation on the improvement of a participant's stereo depth perception at the $p < 0.05$ level for the seven conditions $[F(6,77) = 2.23,\ p = 0.049]$. We therefore followed up by performing a two-tailed t-test to determine the largest pole merge start angle using cubic interpolation that did not show a significant

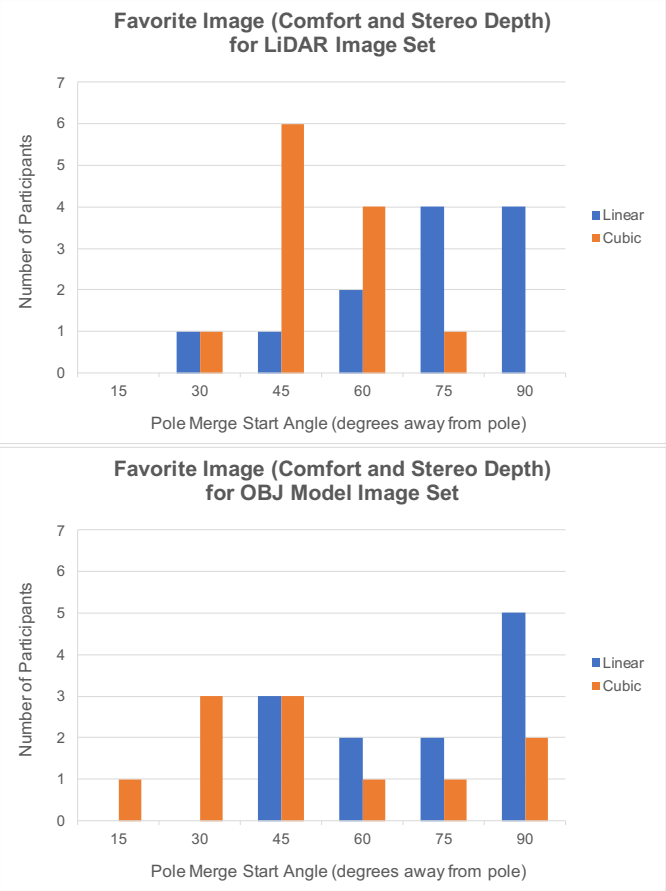


Fig. 10. User study results for favorite image (most comfortable while still maintaining quality stereo depth) with various severities of pole merging.

difference in improved stereo depth perception when compared against the traditional ODS image (pole merge start angle = 0°). We found that a pole merge start angle of 75° resulted in no significant difference at the $p < 0.05$ level [$p = 0.13$].

Similarly, when viewing the scene with OBJ models, there was no significant effect of pole merge start angle with linear interpolation on the improvement of a participant's stereo depth perception at the $p < 0.05$ level for the seven conditions [$F(6,77) = 0.22$, $p = 0.97$]. There was also no significant effect of pole merge start angle with cubic interpolation on the improvement of a participant's stereo depth perception at the $p < 0.05$ level for the seven conditions [$F(6,77) = 1.46$, $p = 0.20$]. This was slightly surprising since there is a noticeably lower average when the pole merge start angle was 90°. A larger sample size would help determine whether this decrease in stereo depth perception actually is significant.

These results show that, in all but the most extreme cases, the use of pole merging does not negatively impact a user's stereo depth perception when viewing ODS images in a VR headset. The improvement in depth perception offered by the ODS image compared to the monoscopic 360° panorama remained fairly constant except when the pole merge start angle began at the equator (90° from the poles) and cubic interpolation of the adjusted interocular distance was used.

6.3 Dynamic Image Comparison Results

For part 2 of the study, users were able to dynamically switch between the ODS images with each pole merge start angle (0° to 90°) for each interpolation exponent (linear and cubic). Users were asked to select the most comfortable image that still maintained quality stereo depth and mark it as their favorite for each interpolation exponent. Results are illustrated in Fig. 10 and mostly align with results of the visual comfort and stereo depth perception tasks. The only somewhat surprising result was the number of participants who selected 90°, even though some fall-off in stereo depth perception seems to occur based on the results of rating each image separately.

Participants were then asked to choose which of their two favorite images was the most comfortable while still maintaining quality stereo depth (linear, cubic, or no difference). For the LiDAR point cloud scene, 4 participants selected linear, 7 selected cubic, and 1 selected no difference. For the OBJ model scene, 4 participants selected linear, 3 selected cubic, and 4 selected no difference, and 1 did not respond. In total, there did not seem to be a strong preference for one interpolation exponent vs. the other. However, we do note that using cubic interpolation seems to result in comfortable viewing at lower pole merge start angles, meaning that a larger portion of the ODS image maintained proper interocular distance.

7 Conclusion

We have presented a technique for rendering omnidirectional stereo images for interactive viewing in virtual reality environments. Our technique expands upon previous geometry-based approaches for creating 360° surround-view images, as we provide solutions for dynamically tessellating input meshes and handling cases when triangles cross a discontinuity in the equirectangular projection. We also provide a method for generating omnidirectional stereo image pairs that can leverage tunable pole merging parameters to reduce the binocular misalignment that can become pronounced in the peripheral vision when looking nearly straight up or straight down. An implementation of our technique using GLSL shaders is available at `https://github.com/tmarrinan/omnistereo`.

Performance results of two fairly complex sample scenes demonstrate that our dynamic tessellation algorithm has minimal computational overhead while ensuring that all triangles project to a sufficiently small area to reduce visual artifacts to a negligible level. Thus, our technique is well suited for real-time rendering applications. We envision our real-time ODS rendering being useful for applications such as recording video game play that can later be watched while viewing in any direction, or in situ large-scale scientific visualization where high-performance computing resources can quickly render ODS images, leaving the majority of compute cycles available for the simulation.

We also conducted a user study to gain insight on the impact pole merging has on visual comfort and stereo depth perception when viewing ODS images in a HMD. Our results show that pole merging can greatly reduce the visual discomfort associated with binocular misalignment in the north and south pole regions of the image. Our results also show that pole merging does not significantly reduce the improvement in depth perception that viewing an ODS image provides as compared to a monoscopic 360° panorama, except in the most extreme cases (when the interocular distance begins to reduce near the equator).

While fairly generalizable, there are a couple of limitations with our work. First, our real-time geometry-based equirectangular projection technique relies on the use of tessellation and geometry shaders. Therefore OpenGL 4.0 / Direct3D 11 or higher is required. Additionally, if an application already uses the tessellation or geometry shaders for another purpose, it may be more difficult to integrate ODS rendering. We also would like to point out that results from our user study are limited in scope as well. Only a discrete set of pole merge start angles using two interocular distance interpolation exponents were tested. Additionally, stereo depth perception was only qualitatively measured.

In the future we would like to leverage our ODS rendering technique for real-time streaming, allowing many users to view a live stream in VR with each user able to control their own view. We also think it would be valuable to conduct a follow-up user study where users can control all the pole merge parameters dynamically, and depth estimation of targets is measured quantitatively in addition to visual comfort.

Acknowledgments

We would like to thank the participants of the user study for their time and energy spent helping us better understand the effects of pole merging when viewing ODS images in a VR headset.

REFERENCES

[1] J. Ardouin, A. Lécuyer, M. Marchal, and E. Marchand. Stereoscopic rendering of virtual environments with wide field-of-views up to 360°. In *2014 IEEE Virtual Reality (VR)*, pp. 3–8, 2014.

[2] Babylon.js. Welcome to Babylon.js. `https://www.babylonjs.com/`. Accessed: 2020-09-06.

[3] S. Best. Perceptual and oculomotor implications of interpupillary distance settings on a head-mounted virtual display. In *Proceedings of the IEEE 1996 National Aerospace and Electronics Conference NAECON 1996*, vol. 1, pp. 429–434 vol.1, 1996.

[4] Blender 2.83 Manual. Rendering >> render output >> stereoscopy >> usage. `https://docs.blender.org/manual/en/latest/render/output/stereoscopy/usage.html#stereo-3d-camera`. Accessed: 2020-08-20.

[5] D. Bodington, J. Thatte, and M. Hu. Rendering of stereoscopic 360° views from spherical image pairs. Technical report, Department of Electrical Engineering, Stanford University, 2015.

[6] K. R. Boff and J. E. Lincoln. Engineering data compendium: Human perception and performance. Technical report, Armstrong Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, OH, 1988.

[7] P. Bourke. Synthetic stereoscopic panoramic images. In H. Zha, Z. Pan, H. Thwaites, A. C. Addison, and M. Forte, eds., *Interactive Technologies and Sociotechnical Systems (VSMM 2006)*, vol. 4270, pp. 147–155. Springer, Berlin, Heidelberg, 2006.

[8] Chaos Group. VRayStereoscopic. `https://docs.chaosgroup.com/display/VMAX/VRayStereoscopic`. Accessed: 2020-08-20.

[9] R. G. de A. Azevedo, N. Birkbeck, F. De Simone, I. Janatra, B. Adsumilli, and P. Frossard. Visual distortions in 360° videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(8):2524–2537, 2020.

[10] A. Gaggioli and R. Breining. Perception and cognition in immersive virtual reality. In *Emerging Communication: Studies on New Technologies and Practices in Communication*. IOS Press, 2001.

[11] Google Inc. Rendering omni-directional stereo content. `https://developers.google.com/vr/jump/rendering-ods-content.pdf`. Accessed: 2020-08-20.

[12] H. Grasberger. Introduction to stereo rendering. Student Project, 2008.

[13] J. Heller and T. Pajdla. Stereographic rectification of omnidirectional stereo pairs. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1414–1421, 2009.

[14] Y. Lee, Y. Kim, H. Kang, and S. Lee. Binocular depth perception of stereoscopic 3d line drawings. In *Proceedings of the ACM Symposium on Applied Perception*, SAP '13, p. 31–34. Association for Computing Machinery, New York, NY, USA, 2013.

[15] H. Lorenz and J. Döllner. Dynamic mesh refinement on GPU using geometry shaders. In *Proceedings of the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pp. 97–104, 2008.

[16] K. Petkov, C. Papadopoulos, M. Zhang, A. E. Kaufman, and X. Gu. Interactive visibility retargeting in vr using conformal visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1027–1040, 2012.

[17] L. B. Rosenberg. The effect of interocular distance upon operator performance using stereoscopic displays to perform virtual depth tasks. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, pp. 27–32, 1993.

[18] A. Simon and S. Beckhaus. Omnidirectional stereo surround for panoramic virtual environments. In *ACM SIGGRAPH 2003 Sketches & Applications*, SIGGRAPH '03, p. 1. Association for Computing Machinery, New York, NY, USA, 2003.

[19] M. Trapp, H. Lorenz, and J. Döllner. Interactive stereo rendering for non-planar projections of 3d virtual environments. In *2009 International Conference on Computer Graphics Theory and Applications (GRAPP)*, pp. 199–204, 2009.

[20] C. W. Tyler, L. T. Likova, K. Atanassov, V. Ramachandra, and S. Goma. 3D discomfort from vertical and torsional disparities in natural images. In B. E. Rogowitz, T. N. Pappas, and H. de Ridder, eds., *Human Vision and Electronic Imaging XVII*, vol. 8291, pp. 212–220. International Society for Optics and Photonics, SPIE, 2012.

[21] Yonggao Yang, J. X. Chen, and M. Beheshti. Nonlinear perspective projections and magic lenses: 3D view deformation. *IEEE Computer Graphics and Applications*, 25(1):76–84, 2005.