Warping Images to Fit a Cylindrical Virtual Reality System

Project Documentation

Taisuke Yasuda
Max Planck Florida Institute for Neuroscience
August 17, 2014

1 Project Description

Our goal is to create a simple virtual reality system that is suited for imaging the brain activity of mice under a microscope easily. We need a relatively wide field of view in the Virtual Reality setup, as this is what rodents are accustomed to in a real environment. In our Virtual Reality setup, we will use the Unity Game Engine to generate a virtual maze for mice and project it onto a cylindrical screen. An optical sensor detects the mouses movements as it moves on a freely spinning styrofoam ball. The mouse will be placed at the center of the cylindrical screen, and a laser scanning projector will project the Unity maze game from the opposite side of the screen. In order to correctly project the Unity maze game, we must warp its images to account for the abnormal shape of the screen.

2 Unity Game Setup Part 1

In the new Unity Project, we first create a plane that represents the ground in the game, and attach a box collider and a terrain render on it. Then, we place a first person character above the ground. We will attach two additional cameras on the first person character, representing the left and right sides of the image for the mouse. For our setup, both of the attached cameras have a field of view of $75°$ and they are set at $75°$ away from each other. This gives the mouse a cumulative $150°$ field of view. This concludes our initial setup of the Unity Game. Eventually, we will project the images obtained from the two attached cameras onto a warped mesh inside the game, one mesh for each side. The warped meshes will have all the required corrections for the images. The unused main camera will record the two meshes, giving us the final product of the $150°$ field of view that is correctly warped.

3 Preparing the Mesh Part 1

In order to warp the images obtained from the attached cameras in the Unity game, we need a basic square grid mesh to start off with. This can be either created using the Grid Generator in MeshLab or by writing a program that does this. A java code is attached in a separate document which creates an object file that is very similar to the object file created by the MeshLab Grid Generator. The square grid mesh object file must be created in one of these two ways in order to directly use the warping algorithms attached later. When using the MeshLab Grid Generator, the file should be named Gridnxn.obj, where n is the number of vertices on each side of the square grid. For our setup, we used a 200×200 grid for a relatively high resolution image.

4 Warping the Image Part 1

Before we project the game images to the cylindrical screen, we must assign the three dimensional spherical field of view of the virtual first person character to a cylindrical field of view that matches the real cylindrical screen. Therefore, this will be our first warping process. First, we notice that the images obtained from the Unity Game Engine do not appear to have realistic dimensions. This is due to the transformation the image undergoes when the Unity Game Engine projects the three dimensional view of the Virtual Character to the two dimensional monitor. In other words, the Unity Game Engine assigns the spherical field of view of the first person character to a plane. This simplifies our task, as assigning points on a plane to a cylinder is easier than assigning points on a sphere to a cylinder.

5 Warp 1

Figure 1 is a top view of the warp process. The xline represents the image that is obtained from the Unity Camera (assigned to plane) and the x'line represents the image after being warped.

x' is proportional to k x' = k = / x'

is also equal to the inverse tangent of x over the radius of the cylinder = arctan (x / r)

By substituting , we get kx' = arctan (x / r) x' = arctan (x / r) / k

Because k is a constant, we can evaluate k by entering any value for and x'. For any , x' is equal to times the radius. k = / x' = /(r)=1 / r

We also need to determine r. In our case, the square grid is modified to have a maximum x value of 0.5 and the maximum value is 37.5o. Therefore, r = 0.5 / tan(37.5o)

If we substitute k in, we get our final equation x' = rarctan (x / r)

By using this equation, we can map all the x points to the corresponding x' points on the cylinder. However, in projective geometry, a point that moves closer to the point of view becomes larger. When we move the points from x to x', the points move closer to the mouse. Therefore, in order to account for this, we must shrink down the y points. Figure 2a is a diagonal view of the warping process. From the subjects point of view, indicated by a circle in the figure, any two point along a single ray that originates from it appears to be in the same location. Knowing this, we can draw the diagram figure 2b.

First, we can write the following proportions y / (r + d) = y' / r y' = yrr+d

However, from figure 2a, we notice that r + d is equal to r2 + d2. Therefore, we get our final equation y' = yrr2 + d2.

By using the two transformation for the two directions, we can map all the points on the plane to points on the cylinder.

6 Warping the Image Part 2

Now, we are ready to fix the image to compensate for the projection to the cylindrical screen. The projector we will use is a laser scanning projector and it is made to project to a planar screen, rather than a cylindrical screen. This means first we must ensure that the image gets projected to the correct points on the cylindrical screen, then transform those points into the image that the projector will actually project.

7 Warp 2

Figure 3 is the top view of the actual experimental setup. Here, x represents the original image, x' represents the image mapped onto the real screen, and x" represents the image that the projector can project. The projector is placed at a distance L from the screen, the radius of the screen is r (different rvalue from before - measure), and y' is the distance between the plane of projection and the screen.

We will first look at the transformation from x to x'. Both points must appear to be in the same location from the mouses point of view, so we follow the ray the connects x and the center of the screen. Then, we can set up a proportion, similar to how we did earlier. x / r = x' / (r - y')

We will find an expression for y' in terms of x' and r r2 - x'2 = r - y' y' = r - r2 - x'2

We can now plug this into the proportion and manipulate the equation x / r = x' / (r - r + r2 - x'2) x / r = x' / r2 - x'2 x2 / r2 = x'2 / (r2 - x'2) (r2 - x'2) x2 / r2 = x'2 x2- [(x'2) x2 / r2] = x'2 x2 = x'2 + (x'2) x2 / r2 x2 = (x'2)(1+ x2 / r2) x2 / (1+ x2 / r2) = x'2 x / 1+(x2 / r2) = x' x' = (x)1 1+(x2 / r2) (rr) x' = (x)rr2 [1+(x2 / r2)] x' = (x) (rr2+x2)

This gives us a formula which we can use to find the x' value for every x value. To find the x" values, we will first set up another proportion, using the rays that the projector emits x" / L = x' / (L + y')

Our goal is to find an equation that ultimately converts from x to x". However, all the x' values in the proportion above we know that we can substitute x values. We will now do that and manipulate the equation. x" = (x') [L / (L + y')] x" = (x) (rr2+x2) [L / (L + y')] x" = (x) (rr2+x2) [L / (L + r - r2 - x'2)] x" = (x) (rr2+x2) (LL + r - r2 - x'2) x'2= x2 r2r2 + x2 x" = (x) (rr2+x2) (LL + r - r2 - x2 r2r2 + x2)

That is the conversion from x values to x" values. However, we still need to do a similar transformation in the z direction. Figure 4 uses the same prime and double prime notations, as well as the r, d, and L values as Figure 3.

We again start off by setting up a proportion z' / r = z / (r + d)

As with before, we notice that r + d is equivalent to r2 + d2. z' = (z) (rr2+x2)

This completes our transformation from z to z'. Now, we can evaluate z" from z', and later z" from z. We look at the proportion created by the projectors light z" / L = (z') / (L + d) z" = (z') [L / (L + d)]

Then we can again plug in the z' values z" = (z) (rr2+x2) [L / (L + d)] z" = (z) (rr2+x2) [L / (L + r2 + x2 - r)] z" = (z) (rr2+x2) (LL + r2 + x2 - r)

This concludes all the warping we need to do to the mesh.

8 Preparing the Mesh Part 2

With these equations, we can write a program that takes the original square grid object and modifies it. The warping algorithm is attached in a separate document. The result are two object files that contain data for two meshes, the left and right sides, that can be imported into the Unity Game Engine.

9 Unity Game Setup Part 2

When the object files for the two meshes are complete, we copy and paste them to the Assets folder in the Unity game. Then, we can create instances of them inside the game and put them side by side. In order to project the images obtained from the two attached cameras of the first person character on to the meshes, we create two render textures. The two attached cameras will each be assigned a render texture, so that the render textures now contain the information of the characters view in the game. Now, we can attach the render textures to the warped meshes so that images themselves are warped together with the warped meshes. Finally, we obtain the image of the warped images by placing an orthographic view camera in front of the meshes. This image is now ready to project to the cylindrical screen when we compile the game. An infinite range of environments can be constructed in this game now, whether it be a maze, puzzle, or anything.