A Qualitative Analysis of 3D Display Technology

Nicholas Blackhawk, Shane Nelson, and Mary Scaramuzza

Computer Science

St. Olaf College

1500 St. Olaf Ave Northfield, MN 55057

scaramum@stolaf.edu

Abstract

We employ the perceptually based error metric developed by Ramasubramanian et al. to guide the construction of a realistic 3D environment with the aid of a 3D display. The model we used was generated from a large data set of left and right image pairs of the real world and was then exported to the Irrlicht animation engine, preserving as much of the original lighting data as possible. We rendered the final model on both the Sony 3D TV and the dual projectors. The engine allows the creation of scene cameras controlled by the user, which allowed us to create two perspective cameras and display the frames using side by side split screens. The final analysis of the model is based on the realism of the model and the 3D effect generated by the display.

1. Introduction

The human visual system is imprecise. Thus any model that is photo-realistic only needs to be as precise as the visual system itself. To that end we employ the perceptually based error metric developed by Ramasubramanian *et al.* to guide the display of a 3D environment. With the aid of two 3D displays, we created a 3D environment using the open source graphics engine, Irrlicht. The engine allows the creation of scene cameras controlled by the user. Since the dual projectors, one of our 3D displays, are connected to a single computer through VGA cables, which are converted to use a single graphics card with two DVI outputs, we are able to extend the desktop and overlay one image on top of the other. The slight displacement of the left image from the right image was achieved by placing two cameras in Irrlicht and applying the split screen capabilities of the engine. For our other 3D display, a Sony 3D TV, we used a single HDMI cable powered by a laptop with a dedicated graphics card. Using the split screen input with the displaced cameras, we displayed one half of the input at the full resolution per frame, switching the half that was displayed.

The same software was used on both the TV and the dual projectors. There were settings on the TV to modify displacement, but we found them to be too coarse for our model. The TV has three different 3D display modes: simulated, side-by-side, and over-and-under. The simulated mode allowed the viewer to input a single display, which is not split screened, and allowed the TV to project the image at twice the normal frequency, shifting ever other frame slightly. This created a simulated 3D effect. The side-by-side and over-and-under modes are similar in the respect that they must have two images placed, as described by their name, either side-by-side or over-and-under in a single frame.

The model we used was generated from a large data set of left and right image pairs. These images were processed, and then transformed into a 3D model using Blender. The model is then exported in the Collada file format to the Irrlicht animation engine, preserving as much of the original lighting data as possible. Using OpenGL, we rendered the final model on both the described displays.

The final analysis of the model is based on the realism of the model and the 3D effect generated by the display. A perception based error metric was implemented to assist in the analysis. Through this error metric, the realism of an image can be calculated based on luminance and contrast. The 3D effect of a model is based on the depth represented in an image, the plane of focus, and the lack of ghosting. Ghosting occurs when the image that is meant to be sent to only the left or right eye can be seen by the opposite eye.

2. Framework

2.1 Irrlicht

Irrlicht is an open source 3D graphics engine which provides a robust framework for displaying scenes. The model was based on images captured by our camera team with a stereo-camera set-up and transformed into scenes by our modeling team. The generated models were then imported into Irrlicht using the open source image format Collada. The scene lighting and texture information was included in the Collada file generated by the modeling team. In order to create a photo-realistic model, with respect to lighting and shadows, we would need to constantly update the model's shadows using a global illumination algorithm. However, our hardware was not capable of running this solution in real time. Instead we used standard ambient lighting, which was set low enough to allow the lighting in the Collada files to handle shadows, which became static. In addition, the buffer system of Irrlicht could only handle 65,000 vertices per mesh, so we had to use a dynamic buffer system included in the Irrlicht library and then cut the model into pieces in order to display so many polygons. In previous model tests a mesh cache system was used to retrieve all of the needed object nodes from other file formats, but these were found to lose information at high polygon counts. The standard OpenGL renderer was used without modification. This is another area which could be improved by modifying the way we render the separate cameras and applying the resulting work from the threshold mapping that is done in 3.2.

2.2 Thresholding

Using threshold calculations we were able to analyze the luminance and contrast in a scene. With this analysis, we are able to determine how far off a static image of our model can be from an image taken of the real world. Our implementation used perceptually based thresholds for comparing luminance, which were calculated for each test image from a series of algorithms developed in Ramasubramanian et al.

The first threshold map determined the amount of luminance that could be changed in the scene. This was done by calculating the result of the Threshold vs. Intensity function, or TVI. The TVI function required every pixel's luminance value, which was calculated using a luminance function, which used differing scalars for the red, green, and blue channels. Once the pixels' luminance values were calculated, the TVI function used a weighted bounding box to determine adaptation luminance of a given pixel. To determine the dimensions of the bounding box, we used a formula based on the full view angle for the width and height of the image. The full view angle came from the camera's specific sensor dimensions and its focal length. Then, we averaged all luminance values within the box to determine the center pixel's particular adaptation luminance. With this information, we used a piecewise approximation to determine what the overall luminance change can be at that particular pixel before it is detectable to the human eye.

We then moved on to calculating the change in contrast, also known as the contrast threshold, at some number of associated peak spatial frequencies. This is accomplished by the using a Gaussian pyramid, which is layers of images that have had a low-pass filter applied a number of times to form new, smaller images and thus forming the various levels of the pyramid. The low-pass's mask is generated from a five by five box of pixels with weighted values. Once the mask has been applied to each pixel, the value generated by the mask is set as the new Gaussian's level's corresponding pixel. This is then repeated for each Gaussian level. We generated a total of seven levels.

The Gaussian pyramid was then used to create a Laplacian pyramid. The Laplacian pyramid is made of layers of images formed from this formula:

$$L\_l = G\_l - expand(G\_l + 1)$$

such that l represents a particular level, L represent Laplacian pyramid, G represents Gaussian pyramid, and expand represents the expand function. The expand function uses the higher Gaussian level and expands it to be the same size as the lower level. This requires an application of the mask.

Once we had the Laplacian pyramid, we moved onto constructing a Contrast pyramid. The Contrast pyramid is made up of layers of images formed from the quotient between the relevant levels of the Laplacian and Gaussian pyramids based on the formula:

$$C\_l = L\_l/Expand(G\_l + 2)$$

where C represents the Contrast pyramid.

These pyramids, the Laplacian and the Contrast, were then used to calculate what the peak frequencies and amount of contrast represented at those frequencies. Specifically, each layer of the Laplacian pyramid is associated with a given spatial frequency. To calculate this, we had to run each layer's background luminance, or average luminance of every pixel, with a slowly incrementing frequency through a contrast sensitivity function:

$$S(f, L) = a * f\hat{}(-bf) * \sqrt{1 + .06\hat{}(bf)}$$

such that S is contrast sensitivity, f is frequency, L is luminance, $a = 440(1 + 0.7/L)^{-0.2}$, and $b = 0.3(1 + 100/L)^{0.15}$. As f rises S will also rise, until it hits a plateau, at which point it will descend. Once this plateau is reached, the associated f is the peak spatial frequency for that particular Laplacian level. These peak spatial frequencies are then combined with the Contrast pyramid's levels in order to determine what the spatial frequency is for a certain part of an image.

3. Methods

Through the use of a 3D device, we focus on generating a realistic 3D model. The analysis of the model can be separated into two metrics: realism and 3D effect. The combination of realism in the model and the amount of 3D effect generates the most realistic displays. When considering the amount of separation, we need to consider the effect that is the result of too much separation, which is called ghosting. Ghosting occurs when the left image can be seen by the right eye and the right image by the left eye. Even slight ghosting can cause the user to immediately reject the realism of the model; however, it is not the only effect that can affect the realism of the model.

To implicitly define realism, we will use the slightly vague definition of just how realistic the model looks. Specifically, we consider what the effect of the lighting, contrast, resolution of the display, and perceived realism of objects would be on the realism of the model. Using the image of the object and an image of the screen displaying that same image, we are able to measure any distorting effects of the display. Since the distortion was so great, we did not need to check the differences between the two images, as there were visually distinct was within the thresholds we generated. For the data we collected on the 3D TV, we observed a distinct increase in the overall amount of blue in the images. This method is not without its flaws; careful consideration must be given to the ambient lighting in the room the display is located in. We believe the distinct color change was due to lack of or improper ambient light.

Determining the 3D effect of an image can be more complicated than determining realism, since we have to deal with two images that are combined using the brain and the visual system, which are both slightly different for every person. Over all there are properties of the displayed model that will make it appear better than another. The first part of analyzing the 3D is based on whether or not it is possible to perceive any depth in the scene. Consider a scene with only a wall at a constant depth. Since there are no other objects in the scene, the viewer will not be able to perceive any depth in the scene beyond the depth of the screen. If there are other objects in the scene and the separation between the cameras is correct, the user will be able perceive a depth that is not the depth of the screen, but the depth between the objects.

This perceived depth is associated with the separation between the cameras, which creates a focal plane at some distance from the camera. Everything that is located along that plane in an image will appear to the viewer to be in focus. Also objects that are located near to the plane will also appear in focus to some users as a result of their eye's ability to change the focal length of its lens. All other objects will appear blurred to the user. Considering the normal vision of the user, this is exactly what we would expect. The location of this blurriness in an image can be calculated using the threshold method described in section 3.2.

Detectable ghosting can be subtle or very noticeable to the user. To measure the amount of ghosting, we need to take the left (or right) image and another image with the polarized screen or left side of the shutter glasses. This would give us what the left eye should see and what the left eye actually sees. The difference between these images would give us the amount of ghosting that is seen by the user. Considering our thresholds again, we can say that if this ghosting is under the threshold map, it is not perceivable by users.

The 3D effect of a display is more complex than the realism of an image. When looking at a 3D display, the amount of 3D that is perceived, the 3D effect, is dependent on the user's visual system to generate a single image from the two displayed images. The quantitative measure of such a system would require testing displays with a variety of users. A qualitative measure of this can be explained using much simpler methods.

4. Conclusion

Our project focused on both the rendering and analysis of realistic 3D models. In order to render the 3D models, the graphics engine Irrlicht was used to render in real time. The two types of 3D displays used to display the image were the Sony 3D TV, and the dual projectors. In order to analyze the 3D displayed image, thresholds for both the luminance and contrast were calculated. These were then combined with a qualitative description of the 3D effect of the display to determine the amount of both blurring and ghosting happening in a 3D display, which gave us an accurate measure of how 3D something was. We were successfully able to display the 3D model using both display devices. Qualitatively, we fell short on exactly how accurate the model is to the real world.

References

1. Ramasubramanian, Maehesh, Sumanta N. Pattanaik, and Donald P. Greenberg. "A Perceptually Based Physical Error Metric for Realistic Image Synthesis." MS thesis Cornell University, 1999. Web.

2. Peter J. Burt and Edward H. Adelson. The Laplacian Pyramid as a Compact Image Code. IEEE Transactions on Communications, 31(4):532–540, April 1983.

3. 3. Wenderoth, Peter. "The Contrast Sensitivity Function." Web. Jan 28, 2012.