Visualization and Interaction of Neural Networks in Virtual Reality

Juan Llanos

Dr. Gordon Erlebacher

Introduction

Computational neuroscientists are interested in understanding the structure and function of the animal nervous system. Animal nervous systems are comprised of arrangements of connected neurons that transmit signals. These structures are commonly referred as neural networks. However, the connections of neural networks and their behavior are too complex to visualize, which creates an obstacle in our path to understanding them. The goal of this project is to use the Oculus Rift virtual reality headset, and the Leap Motion 3D controller to interact with, and visualize, neural networks.

Implementation

In order to accomplish the goal, the following components must be implemented:

1. a 3D renderer with shading
2. Oculus Rift support
3. Leap motion support, allowing for the grabbing of objects
4. Neural network dynamics

Currently I have implemented a 3D renderer with Phong shading using C++ and the OpenGL graphics and mathematics libraries. I have also implemented support for the Oculus Rift, and the Leap motion. This allows the user to interact with objects in a virtual reality environment through the use of hand gestures interpreted by the Leap Motion.

3D Renderer with Phong Shading

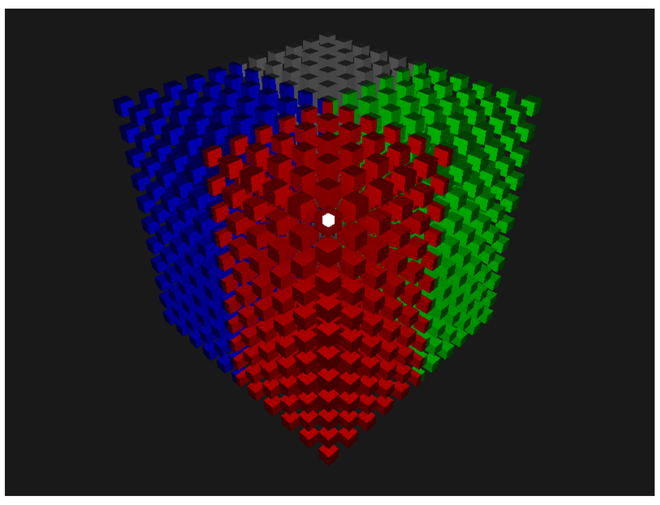


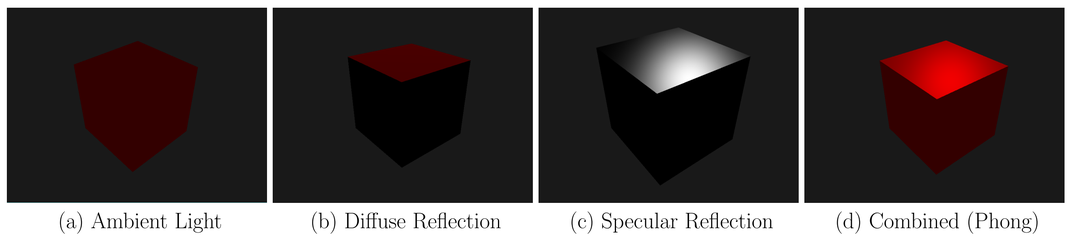
Figure 1: Example of image made by the 3D renderer using Phong shading.

Phong shading is a technique for shading the surfaces of 3D objects in order to simulate the behavior of light in our world. The idea behind the technique is that the color of a point on a surface is described by a combination of ambient light, diffuse reflection, and specular reflection. The total illumination at point p is defined as:

$$I_p = k_a + k_d(\hat{L} \cdot \hat{N}) + k_s(\hat{R} \cdot \hat{V})^\alpha$$

where

- k_a is the ratio of reflection of the ambient light present in all points in the scene
- k_d is the ratio of reflection of the diffuse term of incoming light
- k_s is the ratio of reflection of the specular term of incoming light
- $\hat{L}$ is the vector from p to the light source
- $\hat{N}$ is the normal at p on the surface
- $\hat{R}$ is the direction that a perfectly reflected ray of light would take from p
- $\hat{V}$ is the vector from p to the viewer (or virtual camera)
- α is the shininess constant for the material of the object



(a) Ambient Light (b) Diffuse Reflection (c) Specular Reflection (d) Combined (Phong)

Figure 2: Phong Lighting Model

Oculus Rift Support

In order to render to an Oculus Rift headset, you must perform the following process:

1. Read in location and rotation from the Oculus sensors.
2. Map the location and rotation values from the Oculus coordinate system to the virtual environment coordinate system.
3. Update the location and rotation of the camera view in the virtual environment.
4. Render the camera view from the position of the left eye to the left half of the screen.
5. Translate the camera to the position of the right eye.
6. Render the camera view from the position of the right eye to the right half of the screen.
7. Apply barrel distortion to correct for the pincushion distortion caused by the magnification of the lenses on the Oculus.

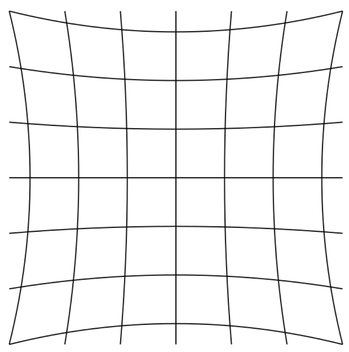


Figure 3: Pincushion Distortion

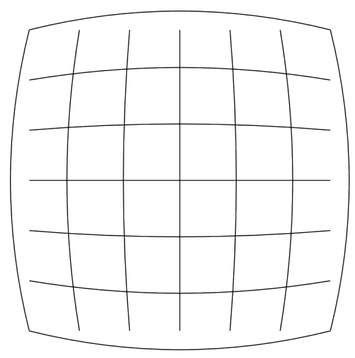


Figure 4: Barrel Distortion

Leap Motion Support

The following steps must be performed to use the leap motion:

1. Read in tracking data from the Leap.
2. Map the location and rotation values from the Leap coordinate system to the virtual environment coordinate system.
3. Use the tracking data to update the location of the hands in the virtual environment.
4. Use the tracking data to detect hand gestures and poses.

A pinch indicates whether the thumb is in proximity of one of the other fingers in the same hand. When a pinch is detected, the nearest object to the pinch is attached to the hand. This allows the user to interact with objects in the environment by pinching to grab and move them.

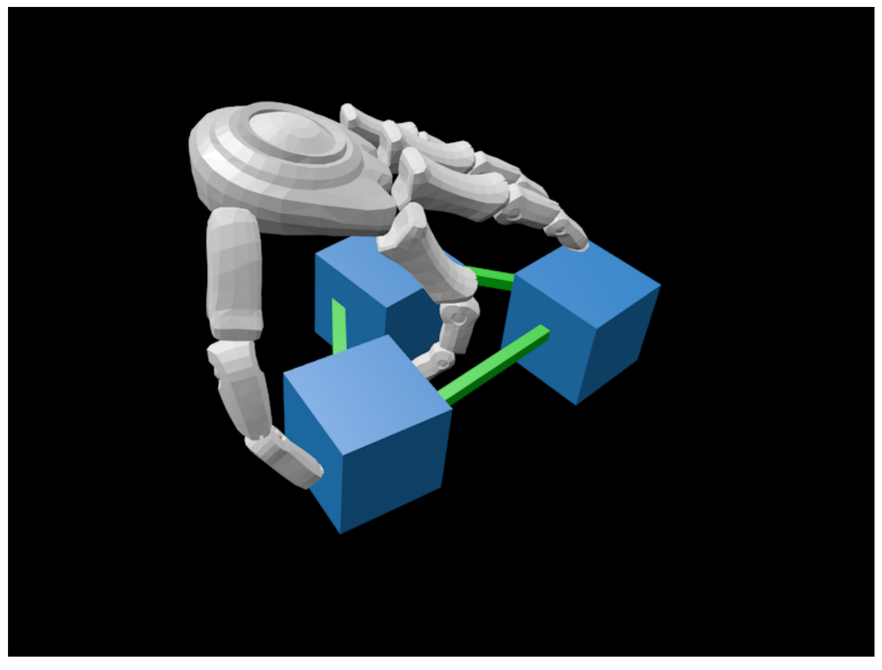


Figure 5: Leap Motion grabbing capabilities

Future Work

Now that I have implemented a virtual environment, the priority of the project will shift towards implementing the dynamics of neural networks. Figure 5 shows a prototype of a small neural network. The neurons will be visualized as simple 3D objects, such as cubes, in order to minimize the amount of faces that need to be rendered and their impact on performance. The connections between neurons will be visualized as tubes, that will glow to show the signals propagating through them. This will allow the user to create the topology of any neural network.