**Chapter 1 Introduction**

This dissertation proposes the use of image based rendering with PCA to reconstruct the original image from the training data. This method is intended to save the rendering time and requirement of devices for complex rendering, such as volume rendering.

* 1. **Motivation**

As it still cost a lot to render the volume data directly on standard hardware in real time, image based rendering could be an approach to speed up the process and directly show the result of volume rendering.

* 1. **Methodology**

The goal is to use PCA to reconstruct original images and use multithread programming to speed up the process of construction, so that we can test if the image based rendering for volume rendering could be used in real world application.

* 1. **Contribution**

Based on the original paper [1], I re-implement the PCA function in C++ and speed up the reconstruction process with OpenCL module in OpenCV. According to the implementation, experiments were created to test the quality of reconstruction images, the memory required for reconstruction, the frame rate for real-time application and the relationship between these three variables.

* 1. **Summary of Chapters**

This dissertation is structured as follows:

* Chapter2 gives an overview of previous works done related to imaged based rendering, principle component analysis, volume rendering, and parallel programming. A detailed overview is given to main works related to this project, IBR (image based rendering) with PCA (principle component analysis) on volume rendering data.
* Chapter3 presents the design and implementations of the preprocessing, the CPU reconstruction, and the method of speeding up the reconstruction by using parallel programming.
* Chapter4 details the result gained from experiment. A discussion relating to the results is also presented.
* Chapter5 summarizes the project and provide a discussion of future work.

**Chapter 2 Background**

**2.1 Image based rendering**

**2.2 Principle component analysis**

**2.3 Volume rendering**

**2.4 Parallel programming**

**Chapter 3 Design and implementation**

**3.1 PCA image based rendering**

In this section, I’ll explain the implementation of the PCA used on training images to reconstruct the original image. Eigen library [4] was used here for matrix calculation and a PCA tutorial [3] was referenced.

**3.1.1 Putting the image pixel value into the matrix**

The raw rbg image has three channels and the specific location’s pixel value is obtained by the corresponding width and height location. For OpenGL, this can be made by using SOIL library. Then, the image pixel value can be accessed by a char array. For example, to use 20\*20 pixels cell-based PCA method, I need to create a matrix with 36 rows and 20\*20 columns to put 20\*20 pixels value into one row for 36 training images. And for each channel, there’re 225 matrixes for saving the whole images information.

. . .

**3.1.2 Subtract the mean**

To get the variance for each pixel, calculate average value in column, get the mean matrix, and use the original matrix to subtract this mean matrix.

**3.1.2 Calculate the covariance matrix**

Covariance is a similar way like standard deviation and variance to perform “how much the dimensions vary from the mean with respect to each other” [3]. It is always measured between two-dimensional data. For example, two vectors X and Y with 10 elements in each vector, where is the mean of vector X, is the mean of vector Y.

Covariance matrix is to measure multi-dimensional vector variance from their mean. For a matrix with n rows. The covariance matrix is as below,

where DataAdjust is the matrix with each row vector subtract the mean vector, DataAdjust\_Transpose is the transpose of the DataAdjust. For the soldier head example I used previously, the covariance matrix is calculated as below:

**3.2 Small training data with CPU**

To build up a prototype of the reconstruction and save the preprocessing time, I used 36 training images, at a resolution of 300\*300 pixels. These images were saved screenshots by rotate the camera around a soldier [2] model’s head horizontally for 360 degrees (10 spacing for the azimuthal angle).

Reference:

[1] Alakkari, S., & Dingliana, J. (2016, September). Volume visualization using principal component analysis. In Proceedings of the Eurographics Workshop on Visual Computing for Biology and Medicine (pp. 53-57). Eurographics Association.

[2] Soldier model link: <https://free3d.com/3d-model/chris-15987.html>

[3] Smith, L. I. (2002). A tutorial on principal components analysis. *Cornell University, USA*, *51*(52), 65.

[4] Eigen library: http://eigen.tuxfamily.org/index.php?title=Main\_Page