**ENGI 3051 – Software Engineering Design II**

**Fall 2015**

**Stereoscopic 3D Reconstruction System**

Software Design Document

Contributing Members:  
Slim Babay, Antoine Stavro

Date Submitted: November 08, 2015

# Approval

This document has been read and approved by the following team members responsible for its implementation:

Developers

|  |  |  |
| --- | --- | --- |
| **Print Name** | **Signature** | **Comments** |
| Slim Babay |  |  |

|  |  |  |
| --- | --- | --- |
| **Print Name** | **Signature** | **Comments** |
| Antoine Stavro |  |  |

Client

|  |  |  |
| --- | --- | --- |
| **Print Name** | **Signature** | **Comments** |
| Greg Toombs |  |  |

# Revisions

Table of Contents

[Approval 2](#_Toc434815109)

[Revisions 3](#_Toc434815110)

[Introduction 5](#_Toc434815111)

[Project Overview 5](#_Toc434815112)

[Design overview 5](#_Toc434815113)

[Changes to specification document 5](#_Toc434815114)

[Glossary 6](#_Toc434815115)

[System Architecture Design 8](#_Toc434815116)

[System architecture 8](#_Toc434815117)

[Object Oriented Design 9](#_Toc434815118)

[Class Diagrams 9](#_Toc434815119)

[Class definitions 10](#_Toc434815120)

[Camera Class 10](#_Toc434815121)

[ReconstructionSystem Class 10](#_Toc434815122)

[Important Classes and Functions Imported from the OpenCV Library 11](#_Toc434815123)

[Sequence Diagrams 12](#_Toc434815124)

[System Interface Description (GUI) 14](#_Toc434815125)

[Pseudocode 15](#_Toc434815126)

[Stream Camera Feed 15](#_Toc434815127)

[3D Reconstruction 15](#_Toc434815128)

[Design impacts 17](#_Toc434815129)

[Updated Project Costs 17](#_Toc434815130)

[Privacy 18](#_Toc434815131)

[Security 18](#_Toc434815132)

[Safety 18](#_Toc434815133)

[Societal Impacts 18](#_Toc434815134)

[Economic Impacts 18](#_Toc434815135)

[Project Costs 18](#_Toc434815136)

[Table of Figures 19](#_Toc434815137)

# Introduction

This software design document serves to provide a detailed overview of the software solution being developed for the client Greg Toombs. A 3D capturing and displaying application is to be developed using Java and object-oriented methodologies. The software can be divided into two separate components: a backend that handles the 3-dimensional reconstruction from images acquired by two separate cameras and a frontend web application that provides a graphical user interface and displays the rendering of the 3-dimensional object produced by the backend. Additional detailed step-by-step descriptions of each use case from the previous requirements document are also provided for further explanation.

## Project Overview

The Stereoscopic 3D Reconstruction System will be deployed on a Linux connected to at least two USB cameras. It will allow users to connect through a web application to capture separate images from each camera and reconstruct a 3D model by utilizing the stereoscopic reconstruction functions within the open-source computer vision library, OpenCV. The 3D model produced by the system will be rendered using WebGL, a web graphics library, to allow for seamless rotation and 3D manipulation of the model. The user will also be able to import and export 3D models.

## Design overview

The Design Document will provide a comprehensive explanation of the design of the Stereoscopic 3D Reconstruction system. This will include class diagrams, sequence diagrams, GUI layouts, algorithms descriptions and pseudocode for important classes.

## Changes to specification document

All changes suggested by the client and instructor to sections that also appear in the design document have been updated appropriately.

The Class diagram has been expanded and further elaborated to better explain the software.

## Glossary

Table - Glossary of terms pertaining to the domain of the project

|  |  |
| --- | --- |
| Term | Description |
| Focal length | The optical distance where light rays converge to form a sharp image of an object to the camera sensor at the focal plane in the camera. The focal length is one of the parameters determining the perspective of the image produced by the camera. |
| Perspective | The visual effect or difference in the apparent sizes of objects that are at different distances from the camera. Given 2 images taken at the same location, changing the focal length of lens will also change the angle of view and magnification of the image. |
| Depth of field | The distance that appears to be in focus in front of and behind the focal point of the lens. This is determined by the aperture, lens focal length, and the distance to the subject. |
| Calibration | Function served to form the scene view by projecting 3D points into the image plane using a perspective transformation of a known model. To simplify the process for the end user, most of this will be done during development with additional adjustments performed based on inter-image correspondence matching  Matrices of intrinsic and extrinsic parameters are produced to offset and normalize factors that would account for errors in stereoscopy. |
| Intrinsic parameters | Internal variables dependent of the source of the image. These include varying factors such a focal length, lens distortion and the principal point offset between different cameras. It creates a mapping between the pixel and camera coordinates in the frame. |
| Extrinsic parameters | External variables dependent on the position and rotation of camera. These include difference perspectives of the scene of the difference cameras and are required for stereoscopy. |
| Object rendering | The process or function of generating an image based on geometry of the 3-dimensional model and the viewpoint or perspective from which it is being viewed. |
| Point cloud | A point cloud is a set of data points (X, Y and Z) of a 3D coordinate system used to form the model and their respective colour. Points represent the external surface of the model and are derived from corresponding matches from images and their calculated location in space. |
| Aperture | A hole or an opening which plays a role in determining how much light reaches the image plane. In the context of this project, it is equivalent to the size of the image sensor of the specified camera. |
| Aspect ratio | Description of the proportional relationship between the width of an image and its height. In the context of this project it is a function of the height divided by its width. |
| Epipolar line | A line vector that corresponds points in one image to another. |
| Feature detection | Process or algorithm used for object recognition by developing key points from a 2D image. Key points are usually acquired from high-contrast regions of the image, such as edges, with the aim to produce a collection of scale and rotational invariant features that can be used for their localization. |
| Feature matching | Process or algorithm used to for object recognition to find features that exist in more than one image that accounts for translation, scale, rotation, and other geometric distortions. |
| Disparity map | A representation, often rendered as a greyscale illustration, of the apparent pixel or feature differences between 2 images. |

# System Architecture Design

## System architecture

The diagram shown below (Figure 1) provides an overview of the web application and servlet architecture.

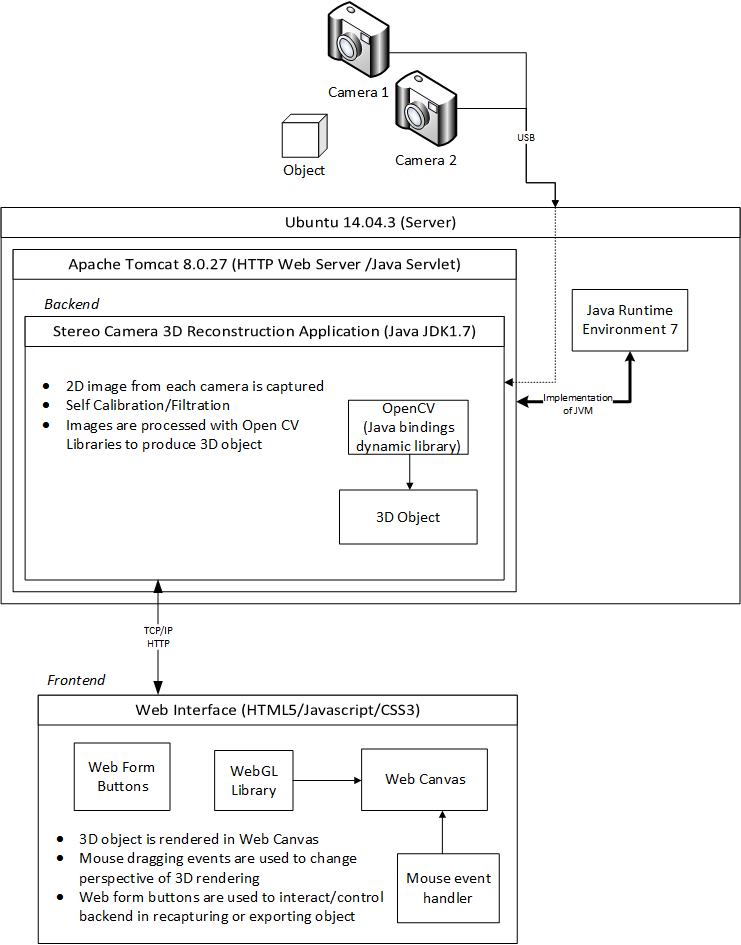


Figure - Stereoscopic 3D Reconstruction System Overview and Architecture

# Object Oriented Design

## Class Diagrams

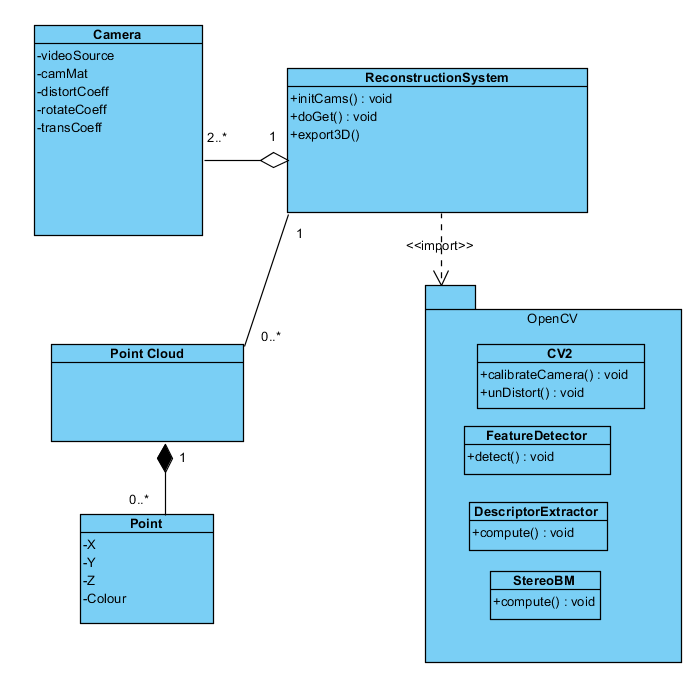


Figure - UML Class Diagram of the Stereoscopic 3D Reconstruction System

## Class definitions

### Camera Class

This class will represent each camera on the system. Defining this class will group relevant information and functions of each camera with its unique intrinsic and extrinsic parameters.

|  |
| --- |
| videoSource : org.opencv.videoio VideoCapture |
| Class used to link software to camera hardware. Methods include retrieve():used to capture a frame image, open()/release():initialize/terminate connection to hardware, and isOpened(): verify connection to camera is ready. |
| camMat: |
| 3x3 matrix containing a set of intrinsic parameters such as the focal length and optical centers of aspecific camera. |
| distortCoeff |
| Set of coefficients used to counter distortion caused by intrinsic factors of the camera. Issues such as bulging (fisheye effect) of the lens and misalignment of the imaging plane and lens need to be rectified. |
| rotateCoeff |
| Set of extrinsic parameters corresponding to rotation vectors to translate coordinates of the camera in 3D space. |
| transCoeff |
| Set of extrinsic parameters corresponding to translation vectors to translate coordinates of the camera in 3D space. |

### ReconstructionSystem Class

This class will be the primary class of the server providing its core functions from initializing the system to communication with clients.

|  |
| --- |
| Cams[] : ListArray<VideoCapture> |
| ListArray to store all the cameras for the system. |
| pointCld[] ListArray<PointCloud> |
| ListArray to store all the pointclouds generated by the system. |
| initCams() : void |
| Method to initialize camera hardware and fill up Cams[] ArrayList. |
| doGet() : void |
| Method to handle get requests sent to server. |
| export3D() :void |
| Converts point cloud into a file that can be sent and saved on the clients computer |

## Important Classes and Functions Imported from the OpenCV Library

|  |
| --- |
| CV2.calibrateCamera() |
| This function is used to determines the camera matrix, distortion coefficients, rotation and translation vectors of each camera. |
| CV2.unDistort() |
| This function undistorts the image taken by the camera using the results from calibration. |
| FeatureDeterctor.detect() |
| This function finds features and key points from each 2D image. |
| DescriptorExtractor.Compute() |
| This function computes the descriptors for image key points in the image. |
| StereoBM.compute() |
| This function computes stereo correspondence using block matching algorithm. |

# Sequence Diagrams

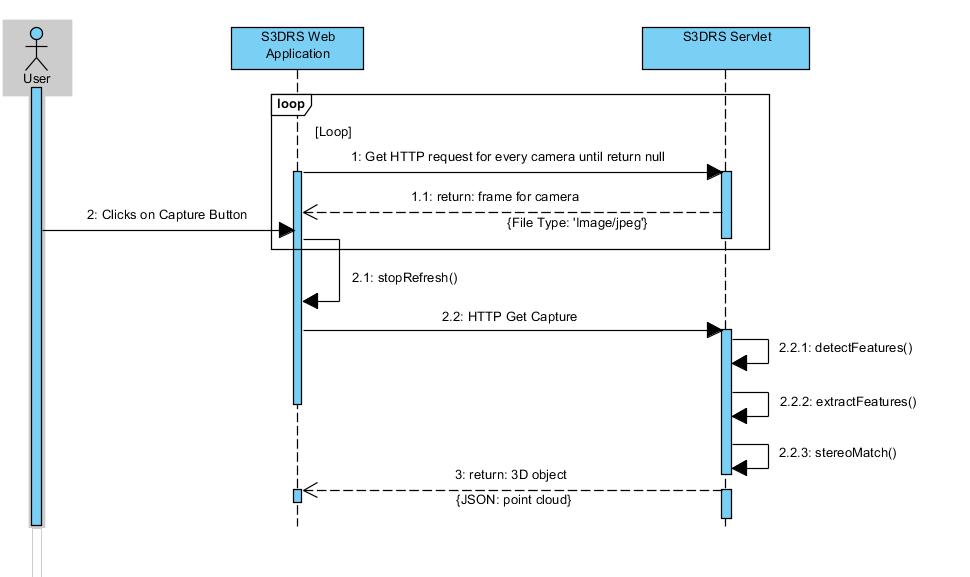


Figure - Capture from Cameras Sequence Diagram

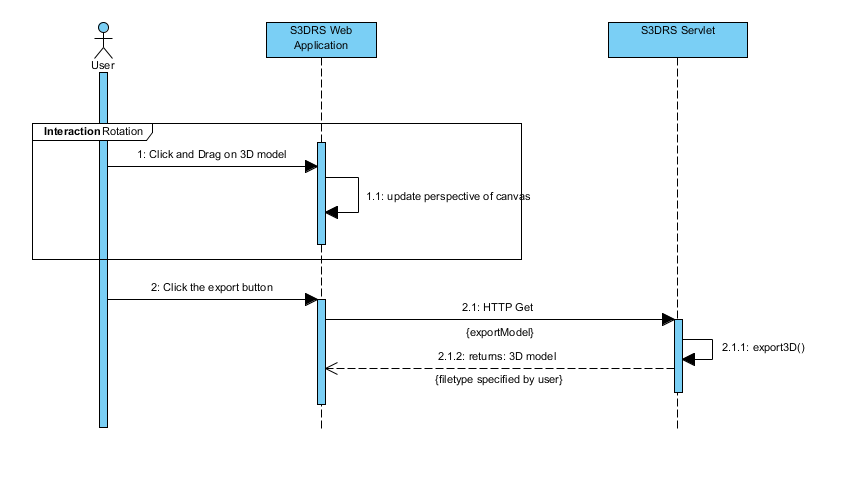


Figure - Export 3D Model Sequence Diagram

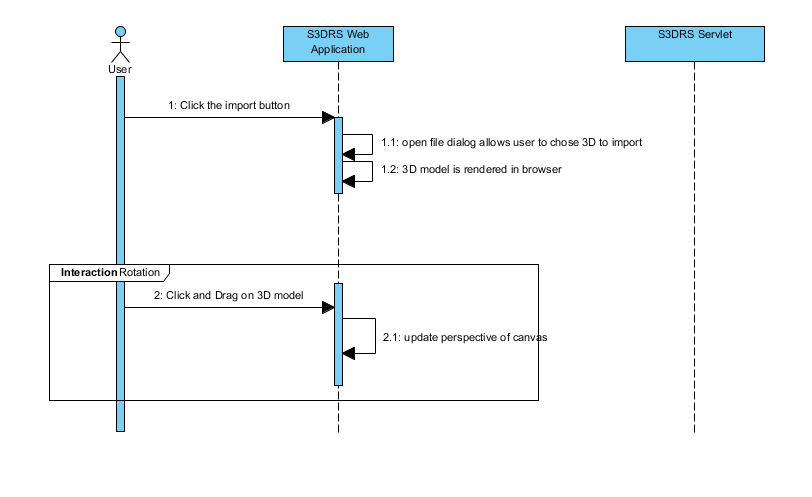


Figure - Import 3D Model Sequence Diagram

# System Interface Description (GUI)

The user will be able to interact with the software through a web interface accessible through the browser chosen by the client as shown below in. The main page will display streaming frames from each camera to allow the user to set up the scene in which they would like to capture.

A button labelled “Capture” will allow the user to capture and produce a 3D model. The 3D model will be reconstructed by the server and sent back to the client to be displayed in the lower section of the page. The streaming images will freeze and the button’s label will be replaced with “Reset” allowing the user to go back to the initial state to recapture their model. Once a model has occupied the 3D canvas area, the user will be able to manipulate its rotation about the scene by clicking and dragging their mouse. The “import” and “export” buttons will also exists next to the 3D canvas area. The “import” button will prompt an open file dialog to allow the user to search for a 3D model file they would like to open and view in the web interface. The “export” button will prompt the server to convert the 3D model into a file that can be downloaded to the clients’ computer.

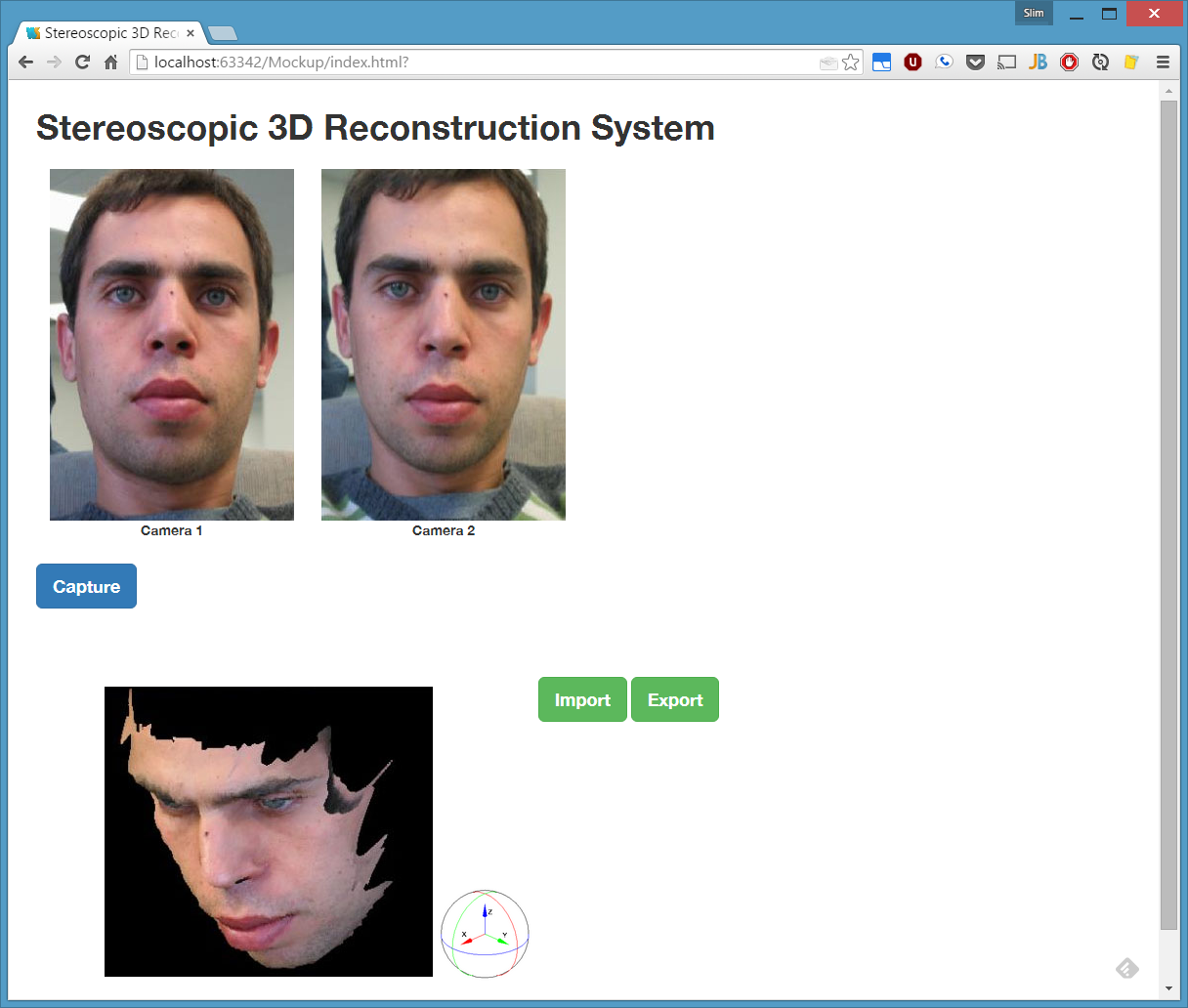


Figure - Web Graphical User Interface of the Stereoscopic 3D Reconstruction System

# Pseudocode

## Stream Camera Feed

//get request “?cam=0”

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

java.io.OutputStream outputStream = response.getOutputStream();

BufferedImage image;

int camreq = Integer.parseInt(request.getParameter("cam"));

if(camreq <= cams.size()) {

Mat f = new Mat();

while (true) {

if (cams.get(camreq).read(f)) {

MatOfByte byteMat = new MatOfByte();

Imgcodecs.imencode(".jpg",f,byteMat);

ImageIO.setUseCache(false);

image = ImageIO.read(new ByteArrayInputStream(byteMat.toArray()));

Graphics2D graphics2D = image.createGraphics();

graphics2D.dispose();

response.setContentType("image/jpeg");

ImageIO.write(image,"jpeg",outputStream);

outputStream.flush();

break;

}

}

}

outputStream.close();

}

## 3D Reconstruction

public class testing {

public static void main(String args[]){

System.loadLibrary(Core.NATIVE\_LIBRARY\_NAME);

ArrayList<VideoCapture> cams = new ArrayList<VideoCapture>();

for (int i = 0;i<10;i++){

VideoCapture c = new VideoCapture(i);

if (c.isOpened())

cams.add(c);

}

System.out.println("Cameras found: " + cams.size());

ArrayList<Mat> frames = new ArrayList<Mat>();

ArrayList<Mat> desc = new ArrayList<Mat>();

ArrayList<MatOfKeyPoint> keypts = new ArrayList<MatOfKeyPoint>();

for(VideoCapture c : cams) {

Mat f = new Mat();

while (true) {

if (c.read(f)) {

System.out.println("Frame Obtained");

System.out.println("Captured Frame Width " + f.width() + " Height " + f.height());

Imgcodecs.imwrite("camera"+cams.indexOf(c)+".jpg", f);

System.out.println("OK");

frames.add(f);

desc.add(new Mat(f.rows(),f.cols(),f.type()));

break;

}

}

}

FeatureDetector detector = FeatureDetector.create(FeatureDetector.FAST);

DescriptorExtractor extractor = DescriptorExtractor.create(DescriptorExtractor.FREAK);

detector.detect(frames,keypts);

extractor.compute(frames,keypts,desc);

StereoBM smatcher = StereoBM.create();

Mat Diffs = new Mat(); //disparity matrix

Smatcher.compute(desc.get(0),desc.get(1),Diffs)

}

}

# Design impacts

## Updated Project Costs

This table represents the level of complexity for each of the components.  The level complexity can be simple, average or complex. The following was derived by estimation and relative comparison to other projects. For example, Inputs and outputs were labelled as complex as they comprise of colour images and colour 3D point clouds.

|  |  |  |
| --- | --- | --- |
| **Component Level of Complexity** | | |
| Input Item | Complex | 6 |
| Output Item | Complex | 7 |
| Inquiry | Average | 4 |
| Master file | Simple | 7 |
| Interface | Average | 7 |
| **UFP** | | **31** |

Figure - Component Complexity Level

TCF is a measure of 14 technical factors with an assigned value from 0-5 to measure its influence.  The highest influence level is 5 and the lowest influence is 0.

|  |  |  |
| --- | --- | --- |
| **Technical Complexity Factor** | | |
| 1. **Data Communication** | Involves transfer of images/3D objects | 4 |
| 1. **Distributed Data Processing** | Majority of data processing occurs on server | 2 |
| 1. **Performance Criteria** | Performance not critical, but synchronized frames are important for moving objects | 3 |
| 1. **Heavily Utilized Hardware** | Prebuilt library function calls should not be too intensive | 2 |
| 1. **High Transaction Rates** | AJAX polling to refresh camera feeds can become an issue with multiple cameras | 4 |
| 1. **Online Data Entry** | Will not be a major factor | 2 |
| 1. **End-User Efficiency** | Will not be a major factor | 2 |
| 1. **Online Updating** | Will not be a major factor | 2 |
| 1. **Complex Computations** | Use of prebuilt library functions | 2 |
| 1. **Reusability** | Intent of software to be very reusable | 4 |
| 1. **Ease of installation** | Software will be bundled into a virtual machine before release | 4 |
| 1. **Ease of Operation** | User should be able to easily use software | 4 |
| 1. **Portability** | If the user can access the server remotely there should notbe an issue using the software | 4 |
| 1. **Maintainability** | The software will self-calibrate but there does not exist a way to push updates after release | 2 |
| **Degree of Influence** | | **41** |

Figure - Technical Complexity Factor

Function Point (FP) is a unit of measure for a software project.

|  |  |  |
| --- | --- | --- |
| TCF | TCF = 0.65 + 0.01\*DI | 1.06 |
| FP | FP = UFP x TCF | 32.86 |

Figure - Function Points

## Privacy

The server will not retain any captured images or models. Any images or models generated are stored in volatile memory on the server, as caching has been disabled, resulting in their loss during any restart or shut down of the server. Due to the nature of interfacing with a browser on the client side, images are temporarily stored but can be deleted at the users’ discretion.

## Security

The client can chose to communicate over HTTPS to almost eliminate security issues of the software.

## Safety

Users will be reminded to pay attention to the USB camera wires as they may pose a tripping hazard that might result in injury. The same wires may also become entangled if the camera system is moving or is around objects that are in motion so it’ll be recommended for stationary use of still objects.

## Societal Impacts

Stereoscopic 3D reconstruction can be implemented in various fields such as autonomous vehicles, robotics, and 3D scanning for manufacturing. While this project is targeting to implement open source stereoscopic 3D reconstruction algorithms, it may provide an easier interface potentially gathering interest and attention in the field of computer vision if it’s shared.

## Economic Impacts

Should this software lead to the development to a commercial product, it might have a beneficial economic impact. Concepts implemented by this project may be useful in areas from manufacturing to automation of mining quarrying and oil/gas extraction. As of 2012, 18.82% of the GDP of Canada is compose by these 2 sectors.

## Project Costs

The software is comprised of open source freeware eliminating all but the hardware costs for the project. As there currently exists no strategy in developing a commercial product from this project, marketing costs and any other associated business related expenses are nonexistent. Consequently, since no product will be sold, revenue generation is also nonexistent. This simplifies the profitability analysis in terms of the value given to the use of the project and availability of the hardware to support it.

# Table of Figures

[Figure 1 - Stereoscopic 3D Reconstruction System Overview and Architecture 8](#_Toc434814407)

[Figure 2 - UML Class Diagram of the Stereoscopic 3D Reconstruction System 9](#_Toc434814408)

[Figure 3 - Capture from Cameras Sequence Diagram 12](#_Toc434814409)

[Figure 4 - Export 3D Model Sequence Diagram 12](#_Toc434814410)

[Figure 5 - Import 3D Model Sequence Diagram 13](#_Toc434814411)

[Figure 6 - Web Graphical User Interface of the Stereoscopic 3D Reconstruction System 14](#_Toc434814412)

[Figure 7 - Component Complexity Level 16](#_Toc434814413)

[Figure 8 - Technical Complexity Factor 16](#_Toc434814414)

[Figure 9 - Function Points 16](#_Toc434814415)

(This page left intentionally blank)