10-24-2013 12:00 AM

Web-based Simulation and Training Environment for Laparoscopic Camera Calibration

Kartik Thakore
The University of Western Ontario

Supervisor
Hanif Ladak
The University of Western Ontario

Graduate Program in Biomedical Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science
© Kartik Thakore 2013

WEB-BASED SIMULATION AND TRAINING ENVIRONMENT FOR LAPAROSCOPIC CAMERA CALIBRATION

Thesis format: Monograph

by

Kartik Thakore

Graduate Program in Biomedical Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Engineering Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

Abstract

Endoscopic cameras are increasingly employed in image-guidance procedures, where the video images must be registered to data from other modalities. However, such cameras are susceptible to distortions, requiring calibration before images can be used for registration, tracking and 3D reconstruction. Camera calibration is learned in a laboratory setting, where configuring and adjusting the physical setup is tedious and not necessarily conducive to learning. A centralized resource that utilizes 3D interactive components needs to be available for training on camera calibration. In this project, a web-based training environment for camera calibration is implemented called SimCAM. SimCAM was developed using the Web Graphics Library (WebGL), Open Computer Vision (OpenCV) library, and custom software components. WebGL and OpenCV were used to simulate camera distortions and the calibration task. The main contributions include the implementation and validation of SimCAM. SimCAM was validated with a content validity study, where it was found to be useful as an introduction to camera calibration. Future work involves improving the supporting material and implementing more features, such as uncertainty propogation.

Keywords: WebGL; OpenCV; Camera Calibration; Endoscopy; Laparoscopic; Training; calibration; Web-based training; 3D interactive environments

Acknowledgments

I would like to thank my advisors, Dr. Ladak and Dr. Peters for their continued patience throughout my program. I would also like to thank both Dr. Ladak and Dr. Peters for their critique of my work and writing. I have learned an incredible amount, with regards to writing, presenting and analysis for which I am eternally grateful. Additionally I would like to thank Alireza Rohani, Arefin Shamsil, Caiwen Huang, Diego Cantor, Dr. Agrawal, Dr. Barron, Dr. Bureau, Dr. Chen, Dr. Eagleson, Dr. Katchabaw, Dr. Parraga, and Uditha Jayarathne, for their support, advice and valuable feedback. Also I would like to thank my family for their continued support.

Finally I would like to thank my dear friend Stacy Roberson, without her this thesis would not have been possible.

Table of Contents

List of Tables

List of Figures

List of Appendices

List of Abbreviations and Acronyms

<<GET>> - HTTP GET request sent from the client to the server

<<POST>> - HTTP POST request sent from the client to the server

<<WS>> - HTTP WebSocket request that initiates a WebSocket between the client and server

2D – two-dimensional

3D - three-dimensional

API – Application programming interface

Bootstrap – HTML and CSS framework

C programming language – High level programming language with efficient data structures, CPU and memory usage

CPAN - Comprehensive Perl Archive Network

CPU - central processing unit

CRUD - basic create, read, update and delete operations on the database

CT - computed tomography

DBI - Database interface

DBIx::Class – Database interface extension library that marshals database tables into class

GPU - Graphics Processing Unit

GUI - Graphical User Interface

HTML5 – Hyper Text Markup Language 5th edition

JSON - JavaScript Object Notation

MATLAB - high-level technical computing language and interactive environment

MIS - Minimally invasive surgeries

MRI - magnetic resonance imaging

MVC - model view controller architecture

MVP – model view presenter paradigm

OpenCV – Open Computer Vision library

OpenGL ES - a cross-platform API for full-function 2D and 3D graphics on embedded systems

ORM - object relational mapping

PET - positron emission tomography

RDBMS - Relational data base management system

REST - Representational state transfer

SimCAM – Name of system created and tested, sort for simulation camera

STD – Standard deviation

SQL - Structured Query Language

SQLite – Server less SQL database

Three.JS – JavaScript library that extends WebGL

WebBrowser – Software package that abstracts web browsers used by users

WebGL – 3D library access to the GPU via the user's web browser enabling

List of Symbols

A - Camera matrix that describes the transformation between P and p

a – Aspect ratio, a ratio between the height and width of the image plane

(c_x, c_y) - Principal point

f – Focal Length in camera matrix A

$k_1, k_2, k_3 -$ 3 coefficients for radial distortion correction equation

P - 3D world point (X, Y, Z)

p – Image point (u, v) or (x, y)

p_1, p_2 – 2 coefficients that describe tangential distortion correction equation

r - The distance of an image point from the center or principal

$[R|t]$ - Homogenous matrix of the camera's rotation (R) and translation (t) relative to the origin point of the world

s - Arbitrary scalar scale factor for perspective camera model

(u, v) or (x, y) – 2D position coordinates in image

$(x_{corrected}, y_{corrected})$ – X and Y positions after correction applied to original positions

1 Introduction

1.1 Motivation

Minimally invasive surgeries (MIS) help to increase patient recovery and decrease complications, and have, in large measure, been enabled by the decreasing size of instruments and laparoscopic cameras. Images from laparoscopic cameras must be highly accurate and distortion free because they are often registered with images from other sources such as pre-operative computed tomography (CT) or magnetic resonance imaging (MRI) [1] [2]. Laparoscopic cameras can be effectively modeled as an ideal pinhole camera [3]. However, the precision and accuracy of MIS applications are affected by distortions because the optical system associated with a typical endoscope is not ideal [4]. Distortions are especially exacerbated as the camera is made smaller for minimally-invasive procedures, which can directly affect application results [5]. As a consequence, it is necessary to perform a calibration procedure to correct these distortions [6]. In addition to this distortion correction, a further calibration step must be performed to relate the pixels in the image to the three-dimensional space within which the image resides [7]. Most camera calibrations enable the intrinsic (distortion and focal length) and extrinsic (pose) parameters to be performed in a single calibration procedure [8].

The most common method of achieving such a camera calibration is to capture images of a known grid pattern from multiple viewpoints. These images are then used to estimate the intrinsic parameters of the camera. Additional steps are then performed to estimate the transformation matrix that relates the 3D coordinate of the camera to those of the real world, also known as the extrinsic parameters. These camera parameters are described more in detail in section 1.2 [8].

Currently there are several camera calibration algorithms and software packages. Regardless of the algorithm however, calibration grid images remain a crucial component. Especially in medical imaging the OpenCV software package is used to perform camera calibration. Camera calibration is described in detail in section 1.3.

Camera calibration is a valuable technique and most medical imaging graduate students learn it through trial and error. There is an existing software alternative called Metrovisionlab which is used to teach industrial camera calibration with a course [9]. Web based training for similar techniques have been found to be effective [10]. The motivation for the current work is that no self-directed 3D interactive camera calibration environment is available for graduate students online.

1.2 Cameras

Laparoscopic cameras, and cameras in general, are devices that record light scattered from objects in the real word and capture them to a screen. In medical imaging, laparoscopic cameras provide visual inputs that can be used to drive a variety of applications. Medical imaging tasks such as registration, tracking and visualization are a few of the applications that depend on camera calibration ([1], [2], [4]).

For example, medical registration applications involve alignment of features and images captured from cameras and other modalities [1]. Medical imaging modalities used to acquire images of the body include radiography (X-ray, Computer Tomography), magnetic resonance imaging (MRI), and positron emission tomography (PET), with laparoscopic cameras also falling into this category ([11], [12]).

1.2.1 Components of a Camera

Cameras are composed of 2 major components, the lens and the recording medium shown in Figure 1-1 [13]. The lens focuses the incoming light onto a medium where the light is recorded. Characteristics of these two components play a part in distortions that is discussed later. These components can be modeled simply as a pinhole camera [14] (Figure 1-2). In this model the lens is the pinhole and the back of the box is the recording medium. The pinhole camera model provides a mathematical basis for camera calibration and is used throughout this thesis.

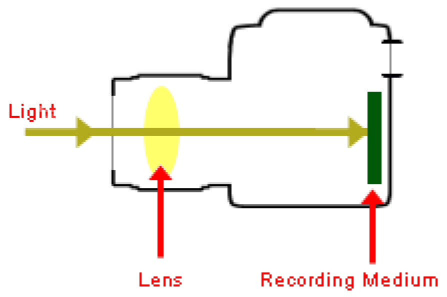


Figure 1-1: Simple Camera Components: Light enters the camera and is focused through a lens system on the recording medium, which stores the image.

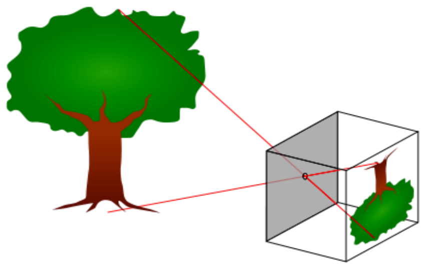


Figure 1-2: Camera image acquisition where a pinhole in a box acts as a lens for the light which is projected on the back of the box – source: Wikimedia commons.

1.2.2 Pinhole Camera Model

The pinhole camera is a common model that describes the image acquisition process of a camera. This model is used extensively in this research and is the basis for camera calibration. A part of camera calibration involves acquiring two sets of parameters described in the pinhole camera model, which describes both intrinsic and extrinsic camera parameters. The intrinsic camera parameters describe internal features of a camera such as focal length and distortions, while the extrinsic parameters describe the external features of the camera such as the orientation and position of the camera [15].

Perspective projection model

In the pinhole camera model, the ideal relationship between the 3D world point $P = (X, Y, Z)$ and the camera's capture point $p = (u, v)$ is modeled as a perspective projection. In the perspective projection model (Figure 1-3) the optical axis is collinear with the Z axis. The optical centre of the camera is defined to be positioned at the origin of the 3D (world) coordinate system, and where the image plane contains the projection of the 3D word point P onto the 2D image point (p). The focal length of the camera is the distance between the optical centre and the image plane. Finally the centre of the image plane is the principal point that decides the offset of the image captured [16].

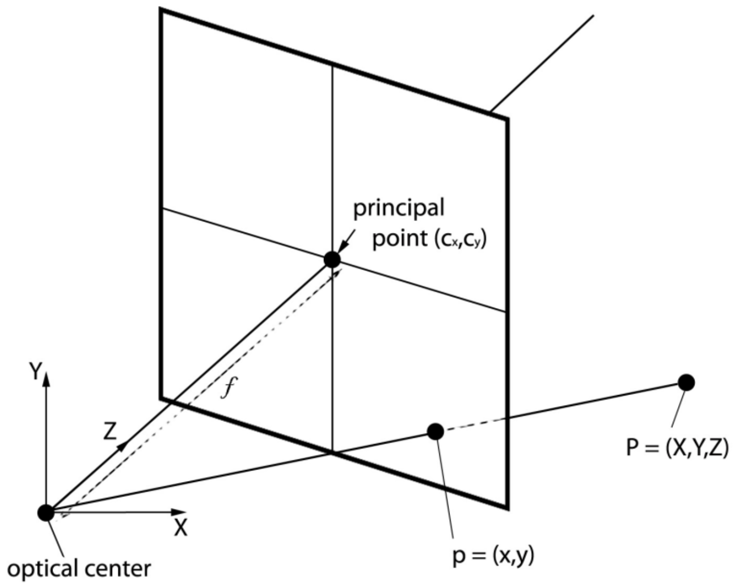


Figure 1-3: Depiction of a pinhole camera in the perspective projection model **[14]** that projects 3D world points P to 2D image points p, where f is the focal length.

The model the projection of a 3D world point $P = (X, Y, Z)$ to the 2D image point $p = (u, v)$ can be written as a relationship (1-1).

$$u = \frac{Xf}{Z}, v = \frac{Yf}{Z} \qquad \textbf{(1-1)}$$

This relationship can be reformulated in matrix notation as denoted in (1-2).

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad \textbf{(1-2)}$$

Equation (1-1) can be amended (Equation 1-3) to account for the principal point (c_x, c_y), which is the offset of the image plane capture.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad \textbf{(1-3)}$$

This principal point is usually at the centre of the image plane. For example if the image plane is 600 pixels in width and 300 pixels in height, the principal point would be (300, 150) pixels. The transformation matrix ($\begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$) is called the camera matrix (A in Equation (1-4)).

$$A \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad \textbf{(1-4)}$$

The camera matrix can also be adapted (1-5) to account for additional parameter such as the aspect ratio (a, the ratio between the height and width) of the image plane.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} af_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad \textbf{(1-5)}$$

1.2.3 Intrinsic Parameters

Linear Intrinsic Parameters

The components of the camera matrix (A in Equation (1-4)) are the linear intrinsic parameters of the pinhole camera model. The linear intrinsic parameters are required to perform correction of cameras. However, real cameras are not perfectly modeled by the ideal pinhole model.

An exception to the pinhole camera model in real lens systems is distortion. Two such distortions are radial and tangential, described below. These distortions are corrected in the camera calibration process, but first we need to define the equations that can appropriately characterize these distortion. The coefficients used in the family of equations that correct both radial and tangential distortions make up the non-linear intrinsic parameters. There are 5 coefficients in total, that describe radial distortions (k_1, k_2, k_3 in Equation (1-6)) (3 parameters) and tangential distortions (p_1, p_2 in Equation (1-8)) (2 parameters) as described below. In the camera calibration process an algorithm, usually Zhang's [8], estimates these parameters to achieve distortion correction.

Radial Distortions

Radial distortions typically exhibit radial symmetry, and are caused by the inability of a typical lens to accurately model the characteristics of a pinhole camera (Figure 1-4). Radial distortions are mathematically modeled and corrected with the following equation:

$$\begin{bmatrix} x_{corrected} \\ y_{corrected} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \qquad \textbf{(1-6)}$$

where $\boldsymbol{x_{corrected}}, \boldsymbol{y_{corrected}}$ are the corrected $\boldsymbol{x}, \boldsymbol{y}$ points in an image after the image has been process with equation (1-6) to remove radial distortions.

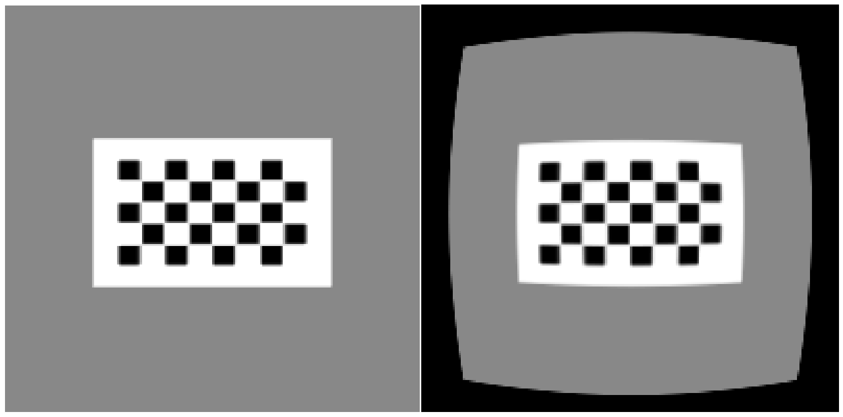


Figure 1-4: Example of radial distortion, applied on the left image. The left image is also the ideal correction of the radially distorted image.

In Equation (1-6), r is defined as the distance of an image point from the centre (Equation (1-7)).

$$r^2 = x^2 + y^2 \tag{1-7}$$

The radial distortion correction equation lends three coefficients (k_1, k_2, k_3 Equation (1-6)) to the non-linear intrinsic parameters. Figure 1-6 shows an example of tangential distortion that can be corrected with the ideal coefficients values (-10, -100, -1000). The corrected image is shown in Figure 1-5.

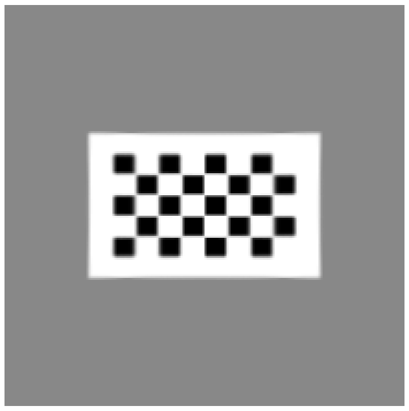


Figure 1-5: The distorted image corrected with coefficients (-10, -100, -1000) for the radial distortion correction equation applied.

Tangential Distortions

Tangential distortions (Figure 1-6) occur along the X or Y axis and are caused by misalignment between the camera's lens and the recording medium (referring back to Figure 1-1) upon which the image is projected.

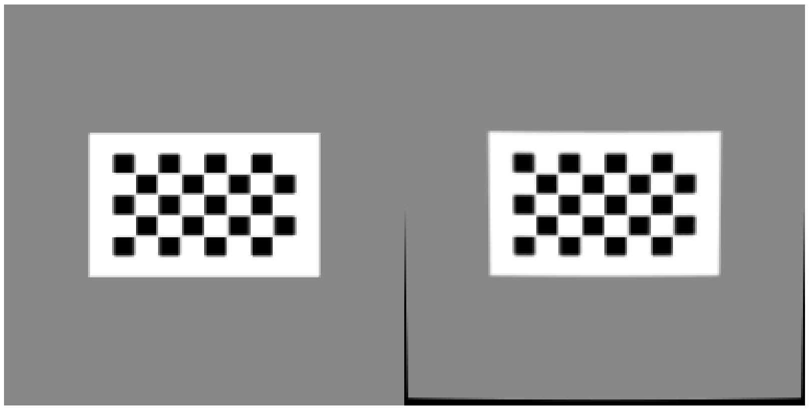


Figure 1-6: Example of tangential distortion, applied on the left image. The left image is also the ideal correction of the tangentially distorted image.

Tangential distortions can be mathematically modeled and corrected with the following equation:

$$\begin{bmatrix} x_{corrected} \\ y_{corrected} \end{bmatrix} = \begin{bmatrix} x + (2p_1 y + p_2(r^2 + 2x^2)) \\ y + p_1(r^2 + 2y^2) + 2p_2 x \end{bmatrix} \qquad \textbf{(1-8)}$$

In Equation (1-8), again r is defined as the distance of an image point from the center (Equation (1-7)). The tangential distortion correction equation lends two coefficients (p_1, p_2 in Equation (1-8)) to the non-linear intrinsic parameters. Figure 1-6 shows an example of tangential distortion that can be corrected with the coefficients values (-0.1, 0). Figure 1-7 shows the corrected image, where $p_1 = -0.1$ and $p_2 = 0$.

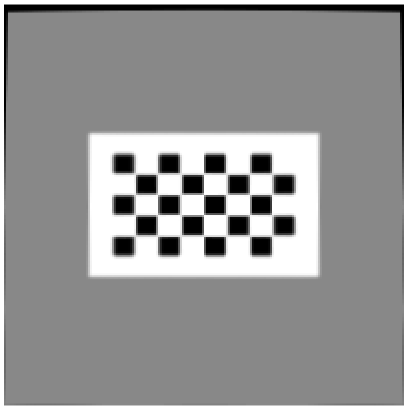


Figure 1-7: The distorted image corrected with coefficients (-0.1, 0) for the tangential distortion correction equation applied.

1.2.4 Extrinsic Parameters

The final set of parameters that describe a cameras' pose, are the extrinsic parameters, which is a homogenous matrix of the camera's rotation and translation relative to the origin point of the world coordinate system ($[R|t]$ Equation (1-9)).

$$[R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \tag{1-9}$$

The pose matrix (Equation (1-10) can then be applied to the perspective camera matrix Equation (1-4), to achieve the expanded camera transformation (Equation (1-10)).

$$s * \ A * \ [R|t] * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{1-10}$$

1.3 Camera Calibration

Camera calibration involves estimating the intrinsic and extrinsic parameters of the camera as discussed in earlier sections [14]. These parameters are required to determine the camera's position and orientation (pose) from which an image is acquired. Specifically, Zhang's algorithm [8], well established for camera calibration, employ a calibration rig, consisting of a planar checkerboard pattern as shown in Figure 1-8 [17]. By taking many images of the pattern with the camera, a set of feature points can be extracted (Figure 1-8). The feature points and their locations are recorded to create a projected image coordinate equation. Every captured image has a unique set of these equations and a closed form solution can be formed which can be used to estimate further parameters. The extrinsic parameters are estimated based on the camera position relative to an arbitrary origin position for the real world coordinates (P). With respect to correcting lens distortion, there are several algorithms (Zhang [8], R. Tsai [18], etc.) that maybe used to accomplish this, but the practice and process is usually similar. With several image captures, the feature points can be used to create a system of equations relating 2D image positions to 3D world positions. Solving this system of provides the intrinsic parameters (especially the camera matrix Equation (1-4)), discussed earlier, that define the relationship between the world coordinate systems to the image plane.

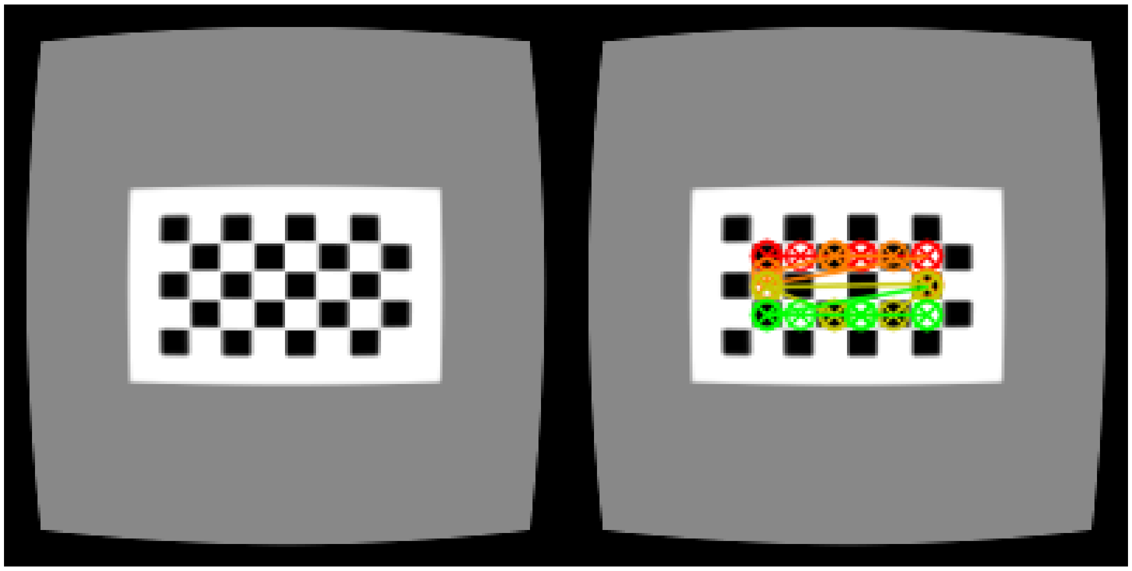


Figure 1-8: Image on the right shows extracted features from the corners of the calibration grid from the left image.

1.3.1 Calibration Software

Several algorithms are available that perform camera calibration which are usually based on Zhang's calibration algorithm which is used extensively in this thesis [8]. An implementation of Zhang's is available with the open-source Open Computer Vision (OpenCV[1]) software library [19].

1.4 Available Training for Camera Calibration

Camera calibration is taught in computer vision courses, however it is a minor component and practical application of the technique may not be covered. A simulation software package called Metrovisionlab [9] is available and used for training camera calibration in an industrial computer vision class. Metrovisionlab, available as an offline MATLAB module. Metrovisionlab simulates several aspects of the camera calibration process, but doesn't provide a 3D environment. The Metrovisionlab simulator is used in two courses at the Department of Design and Manufacturing Engineering at the Universidad de Zaragoza (Zaragoza, Spain), where it is used to teach students in 4 year of a manufacturing engineering course and thus requires a higher level of understanding to use. Metrovisionlab focuses on explaining how various parameters affect the image taken by a camera, which makes the user interface complex to use.

1.5 Objectives

The objective of this project is to make a web based 3D interactive environment for training in camera calibration, and to validate the content and efficacy of the created system by having experts and non-experts review it.

[1] http://opencv.org/

2 Methodology

The prototyping software engineering methodology was used to guide the project initialization, design, implementation and testing. This methodology is notable for reducing development time of small scale interactive software systems ([20], [21]). Initially information was gathered and was used to specify the implementation that was to be built. The specifications were then used to create two plans namely, the system architecture and the design. These plans were then used to drive the implementation progress and subsequent testing. An implementation called SimCAM was created based on the system architecture and design defined below.

2.1 System Users

SimCAM is designed for users who are learning camera calibration for medical imaging and who are expected to have prior training in linear algebra, computer vision or computer graphics. Users are also expected to have some experience using 3D environments and rich interactive web pages.

2.2 Gathering Specifications

In interviews, user stories were gathered from end users and subject matter experts. The user stories are simple statements of features and behaviours the users expected from the system. Some early prototypes were generated to gauge the feasibility, user experience and to further clarify the specifications. After several initial prototypes were implemented, a list of specifications was developed for the foundation of SimCAM.

2.3 Specifications

The specifications are broken down into two types. Functional specifications describe specific features of the system and its components. On the other hand non-functional specifications describe the operation of a system rather than specific behaviours. The following lists are the specifications of SimCAM.

2.3.1 Functional Specifications

1. Camera simulation
    1.1. User can manipulate cameras' camera matrix
    1.2. User can manipulate cameras' extrinsic parameters
    1.3. User can manipulate cameras' distortion parameters
2. Camera calibration simulation
    2.1. User can select position and orientation of the calibration grid
    2.2. User can capture calibration grid placements
    2.3. User can get calibration parameters for each calibration grid placement
    2.4. User can see difference image of calibrated image vs. undistorted image
    2.5. Calibration will be based on OpenCV implementation of Zhang's algorithm
3. Webcam calibration
    3.1. User can acquire images from their web camera
    3.2. User can capture calibration grid placements from images
    3.3. User can get calibration parameters for each calibration grid placement
    3.4. User can see difference image of distorted image vs. calibrated image
4. Training and Validation
    4.1. Users will be presented background material and quizzes to review material
        4.1.1. A tutorial format will be used with several milestones
    4.2. User metrics will be stored
        4.2.1. Time spent per milestone of the tutorial
        4.2.2. Quiz results
        4.2.3. Answers to content validity questionnaires (defined in Appendix A)

2.3.2 Non-functional Specifications

1. Users will interact with the system through a website
2. System will provide near real-time feedback on camera distortions
    2.1. Image frame rate should be approximately 20 milliseconds

The plan for implementing the functional specifications is defined in the system design; while the plan for implementing the non-functional specifications is defined in the system architecture.

2.4 Implementation

SimCAM was implemented as a web application that provided milestones that are composed of training tutorials, quizzes and interactive components (as depicted in Figure 2-1). SimCAM's architecture was designed as a web application due to the constraints and specifications distilled from the user stories. The architecture of the system is divided into two major components: the server and the client. The server (application server) serves content and resources to users over the web to the user's web browser. The server is responsible for data storage, content generation, computer vision and image processing. The client will process and render the content generated to the web browser. Additional processing such as calibration and distortions are handled by the computer vision and image processing components in the server.

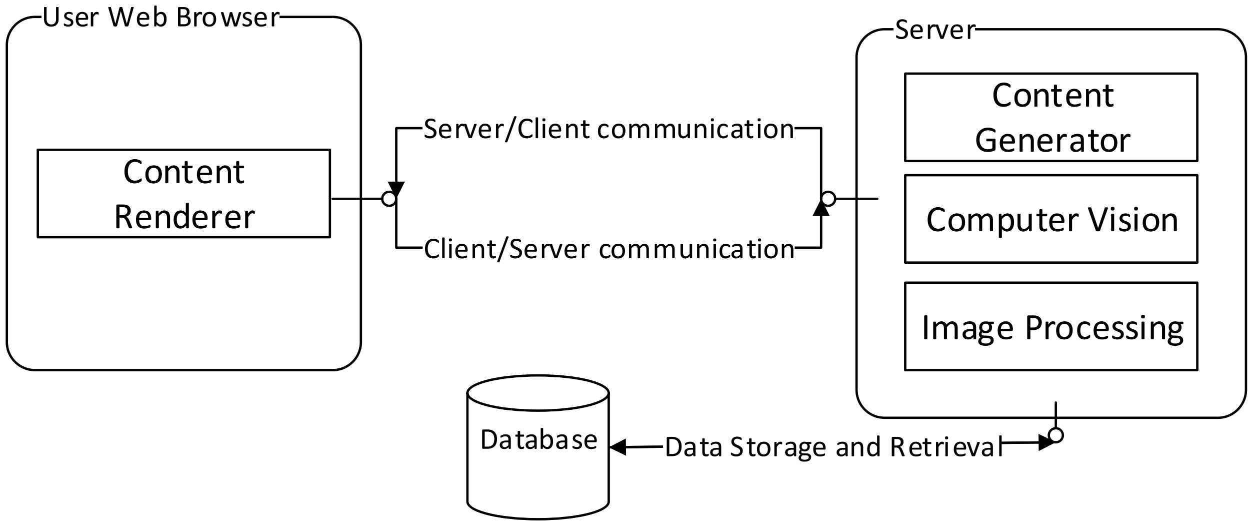


Figure 2-1: System architecture overview of SimCAM, showing the client (code deployed on the user's Web Browser), server and database component.

2.4.1 Programming Languages & Libraries

SimCAM utilizes several programming languages and libraries for the implementation. The SimCAM database was built on SQLite [22], a software library that implements a self-contained, server less SQL (Structured Query Language [23]) database engine. Perl [24], a scripting language, was used to implement SimCAM's application server. Computer Vision components were developed in the C programming language [25], using the OpenCV [26] library. Finally the JavaScript language [27] and HTML5 (HyperText Markup Language) [28], were used in the client, which is responsible for rendering content (such as the 3D simulation environment, distortions, charts, etc.). HTML5 also provided a communication protocol called WebSockets [29]. WebSockets is a persistent connection between the user's web browser and the server. WebSockets was used since there needs to be image data communicated to the server for calibration and image processing. With most broadband Internet connections the communication of image data was near real time.

Database

SQLite [30] was appropriate for SimCAM because only a small amount of data (2 tables) are stored. An alternative to SQLite would be to use another RDBMS (relational database management system), which could handle larger datasets, volume of usage and concurrency. The RDBMS features are currently not required for SimCAM's data usage or specifications and would add another system on the server that would need configuration and management. SQLite, however, uses the same interface (SQL) as other RDBMSs, and migration would not be hindered. Additionally SimCAM uses a database interface (DBI [31]) to abstract the database away from the application.

Perl DBI [31] is an interface implemented in the Perl programming language which standardizes database communications for programmers. Perl DBI has been maintained since 1992, and allows for near database independent SQL code. SimCAM uses Perl DBI for communication with SQLite indirectly via the DBIx::Class, an object relational mapping (ORM) library.

Object relational mapping (ORM) is a technique that converts data between databases and object oriented constructs (usually classes). DBIx::Class generates boilerplate (inconsequential and formulaic) code for basic create, read, update and delete (CRUD) operations on the database. Usage of DBIx::Class [32] increases development speed, abstraction and portability, which facilitates prototyping and faster iteration. Although, DBIx::Class abstractions may increase in complexity for performing comprehensive queries and operations (which can be easily avoided by stored procedures and views on the database), DBIx::Class was appropriate for SimCAM as the queries and operations needed are not more complex than CRUD operations.

Perl

Perl is notable for its ease of integration with several systems, protocols and interfaces. Perl also has a rich ecosystem of reusable code (called modules) available on the Comprehensive Perl Archive Network (CPAN) [33]. CPAN currently provides 123,825 modules that are available on over 271 servers worldwide. CPAN modules are well tested and provided several reusable components that SimCAM is built with (for example DBIx::Class which was mentioned earlier). Perl also provides great flexibility of language and paradigms that is excellent for prototyping applications. However, this flexibility at times allows the user to develop complex components (anti-patterns) that are difficult to migrate and extend. Perl anti-patterns can be avoided by following best practices and testing. The Perl language was also selected for the SimCAM project, for the Mojolicious web framework ([34]) module available on CPAN.

Mojolicious is a real-time Perl web framework with features such as Representational state transfer (REST described in [35]), JavaScript Object Notation (JSON) ([36], [37]) and receiving WebSockets. Mojolicious uses declarative language that emphasizes expression of the server logic, which make defining web services simple. As an example of this declarative language the following script (Figure 2-2) serves the text 'Hello World'.

```
Use Mojolicious::Lite;

Get ‘/’ => {text => ‘Hello World!’};

App->start
```

Figure 2-2: Mojolicious descriptive languages for web services. This example shows a simple <<GET>> request to the root URL path that returns the text ‘Hello World!’.

The declarative language again makes writing complex web services simple and easier to prototype. Mojolicious trades powerful features and abstraction of logic with performance. In SimCAM, Mojolicious’ performance weaknesses are addressed by performing expensive computation either on the client machine or in a memory/CPU efficient language like C programming language.

C programming language

The C programming language is a low level language (relative to Perl) that is notable for efficiency of data structures. In SimCAM the C language was used to implement computer vision features with the OpenCV library. An alternative to C would have been to use Python another scripting language similar to Perl. Usage of Python OpenCV would increase the memory and CPU overhead unnecessarily.

JavaScript & WebGL

JavaScript is a dynamic scripting language available on modern web browsers, providing features such as WebGL ([28], [38]). WebGL was appropriate for the frame rate specifications for data rendering. WebGL allows web browsers access to the Graphics Processing Unit (GPU), a specialized electronic circuit for graphics processing. WebGL is based on OpenGL ES 2.0, a graphics library that is an industry standard. Using WebGL, JavaScript is able to render 3D content on the user’s system without requiring processing from the server.

Because WebGL is based on OpenGL ES 2.0, and is low level, this would increase the development overhead for implementing features. A higher level library called Three.js ([39], [40]) was used to meet the user specifications. Three.js created an object oriented interface to WebGL in JavaScript. Three.JS provided classes and several utilities that address the overhead issue, while still being a cross-browser light weight library. An example of a Three.JS class, is the camera class making it easy to create perspective cameras and implement interaction for updating camera parameters. Three.js can also render scene data to several targets, one of which is a HTML5 [41] canvas tag. The advantage of rendering to a HTML5 canvas is that the final 2D rendered scene from a camera can be extracted and sent to the server for additional processing.

Since the client was responsible for several views, a model view controller (MVC) architecture was selected for the implementation. The Backbone.JS JavaScript library was used as the MVC framework for SimCAM. Backbone.JS [42] is based on the model-view-presenter paradigm [43] (a derivative of MVC), where a presenter is responsible for keeping various views synchronized with the model data on the application. Backbone.JS performs synchronization by utilizing events that are compartmentalized to views, collections, models and routers. Backbone.JS provides much needed structure for SimCAM's interactive user interface and is light weight and compatible across several browsers and operation systems. Alternatives to Backbone.JS (such as Ember.js, Angular.js, etc.) provide more features and abstractions but are less stable.

HTML5

Hyper Text Markup Language was specifically used for the availability of the canvas tag. The canvas tag provides a view-port to a variety of content, especially WebGL content. An alternative would be to use Flash, Java or Unity3D platforms to render the content. Both Flash and Java applications would require an extra interface to transmit content back and forth between the client and server. By using WebGL and HTML5 canvas we avoid this interface by using the existing server interface that is used to render the rest of the data. The advantage of having access to additional 3D rendering features has been traded-off by using HTML5, but the 3D content in SimCAM is fairly simple and does not need those features.

Additionally a library called Bootstrap [44] was used to provide extra features such as scaffolding and a cleaner design. Bootstrap required the client to process additional design components.

However, the additional processing load is negligible and in return the HTML5 content is well structured and rendered, increasing the maintainability and extensibility of the project.

2.4.2 Architecture & Design

The architecture and design describe engineering decisions made prior and during the construction of SimCAM.

Interfaces

Figure 2-3 shows the various classes, libraries and interfaces that make up SimCAM. On the client side, WebBrowser (the WebBrowser is a software package referring to the users' installed web browser), runs the JavaScript SimCAM.js component and libraries. SimCAM.js uses SimCAM::Model sub-packages to request objects as REST requests. The SimCAM::View sub-packages renders various views that depend on Three.JS and WebGL to render content. The views are also responsible for managing and responding to events on the web browsers.

The SimCAM server module was written in Perl, and services the client which runs as JavaScript in a web browser. Communication is done using the HTTP 1.1 protocol. The server running the service is available at http://simcam.imaging.robarts.ca. The Perl server module connects to both the database and to a custom package of OpenCV, which handle the data storage and computer vision components, respectively.

More specifically, the SimCAM.pm Perl module serves and services the SimCAM.JS application on the WebBrowser. SimCAM.PM contains the SimCAMService, where the SimCAMService depends on SimCAM::Schema class, which maps the database to class instances, via the DBIx::Class ORM. SimCAMService also provides the SimCAM::API which exposes access to SimCAM_CV. And SimCAM_CV is a package of custom OpenCV compiled resources that perform computer vision processing. Figure 2-3 shows the architecture in more detail as a class diagram.

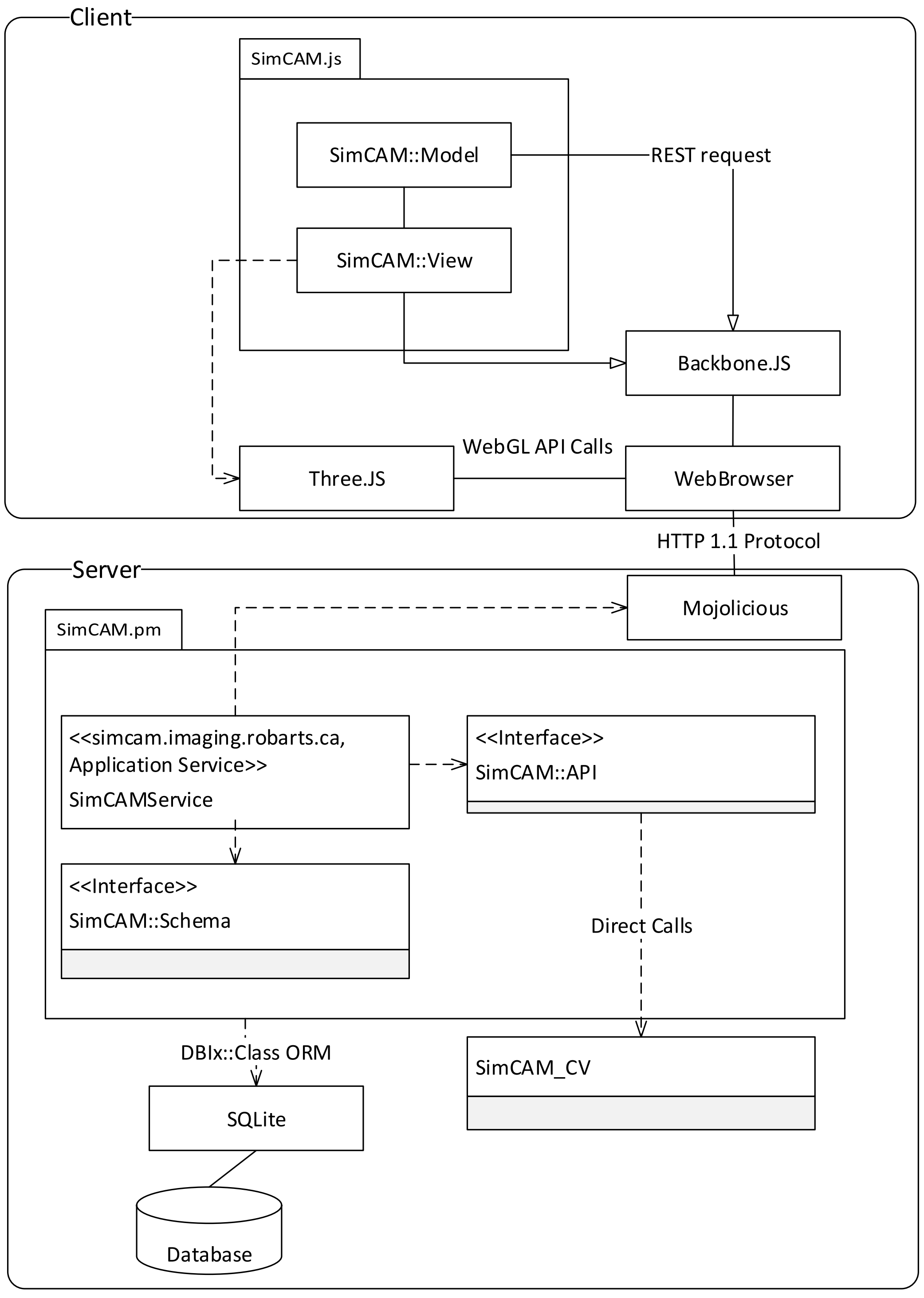


Figure 2-3: SimCAM Architecture that shows specific components responsible for interfaces and communication between the client and the server.

The SimCAMService (depicted in Figure 2-4) defines a service that provides a REST API to front end clients. The resources provided by SimCAMService are sessions, images and computer vision algorithms. Resources are available as URL (uniform resource locator) paths that extend the main service domains (called resources):

- http://simcam.imaging.robarts.ca/ Service
    - /session/ - SessionResource
    - /api/image - ImageResource
    - /api/ - CVResource

Each resource can be accessed with “verbs” such as:

- <<GET>> /api/session/start/1
    - Gets the content for the first session
- <<POST>> /api/image
    - Post new images from the client to the server
- <<WS>> /api/calibration
    - Start a Websocket (bi-directional communication) between the client and server for calibration of image data

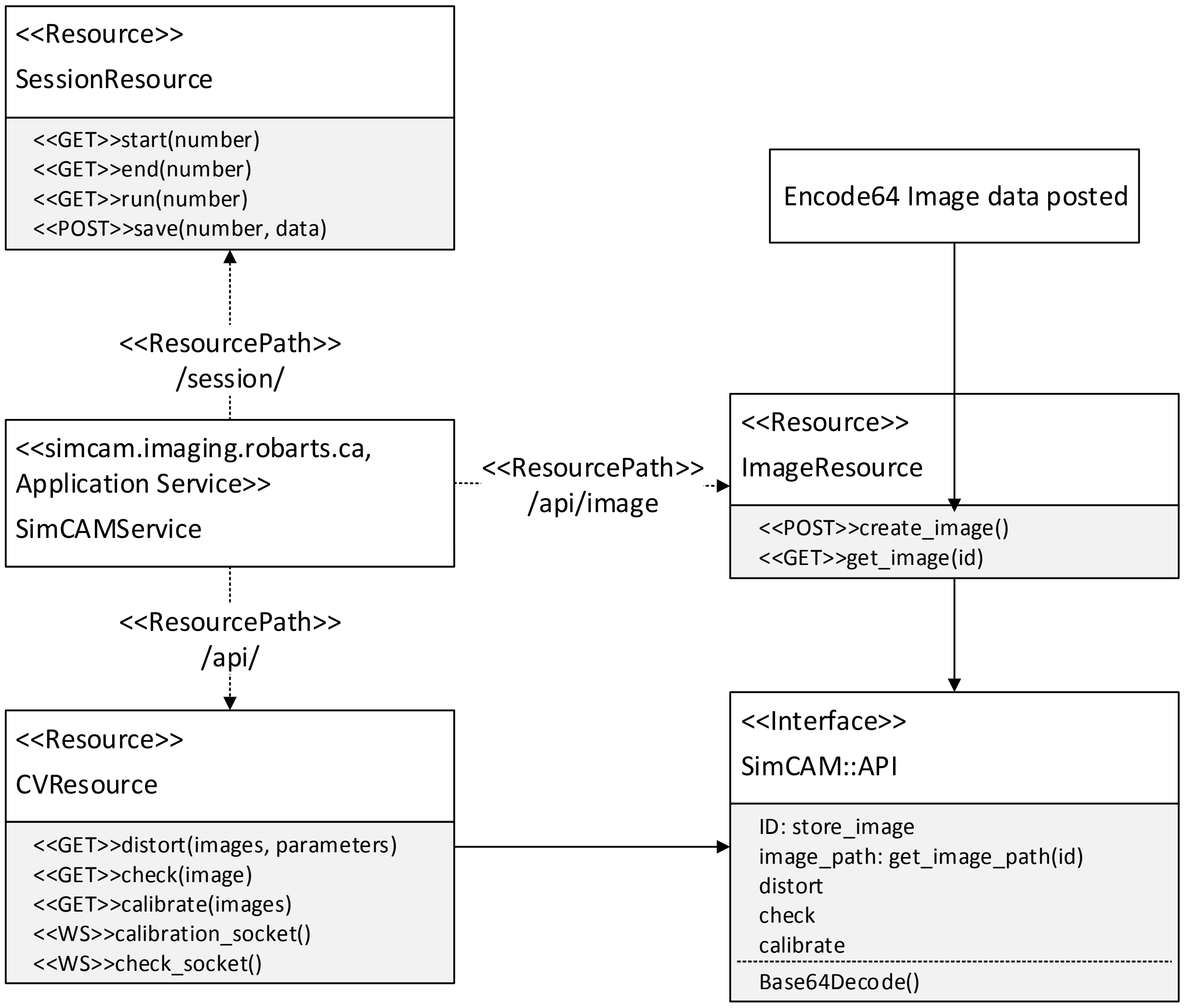


Figure 2-4: Class diagram describing the REST services implementation for SimCAMService.

Application Initialization

The process of initializing the application (Figure 2-5), describes the steps involved in starting SimCM. SimCam is initiated when the user requests the http://simcam.imaging.robarts.ca URL from their WebBrowser. The WebBrowser performs a <<GET>> request to receive the Home Page. Users then submit their email addresses on the Home Page’s login form. In response to the email submission the server provides a unique identification token (called the cookie), which will be stored on the WebBrowser. Meanwhile the email and cookie are stored in the database via SimCAM::Schema ORM interface. Once the user is identified, the correct session resources and the client application are delivered to the WebBrowser, where the session is rendered (triggering the onpageload event). The SimCAM::Router::App JavaScript class is then initialized, which renders appropriate content for the user to see. The SimCAM::Router::App initializes also Views (such as SimCAM::MainView), while binding events from the WebBrowser, that can update the WebBrowser content and URL asynchronously allow for bi-directional interaction with the user.

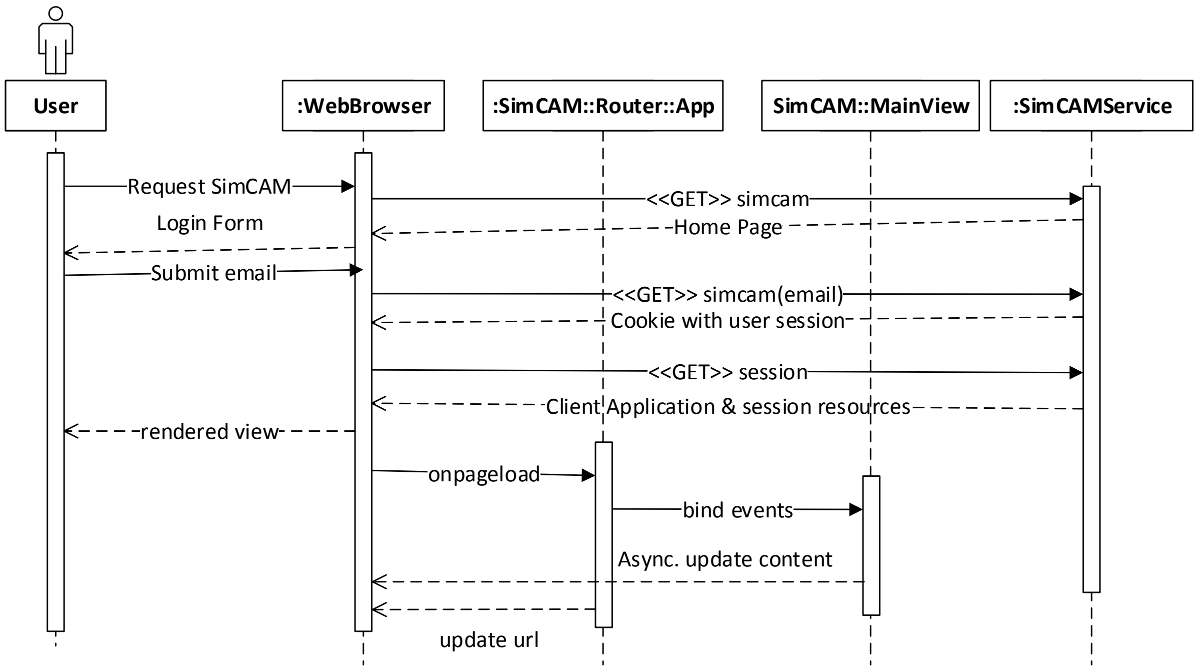


Figure 2-5: Initialization sequence of the client (SimCAM::Router::App) on the WebBrowser. The User triggers the process by requesting the service URL through the WebBrowser. The request triggers the SimCAMService on the server to provide the client application payload, which is then initialized on the WebBrowser when the page is loaded.

Session Resource

The session resource (shown below) is responsible for recording the results of the milestones and quizzes. SimCAM::Schema::Session watches a directory on the server for milestone template files. The template files are presented to users as completed tutorial content with a screen. This request is initiated with <<GET>> SessionResource::start request and if valid redirected to the <<GET>> SessionResource::run request, where the server responds with a specific milestone (associated with an id number). Once the milestone is rendered, the quiz at the end of the milestone can be filled out by the user. When the user clicks submit the data are posted as <<POST>>

SessionResource::save requests. The data from the request are store using the SimCAM::Schema as a SimCAM::Schema::Session in the database. Each SimCAM::Schema::User is provided with cookies that are deployed with SessionResource::start responses.

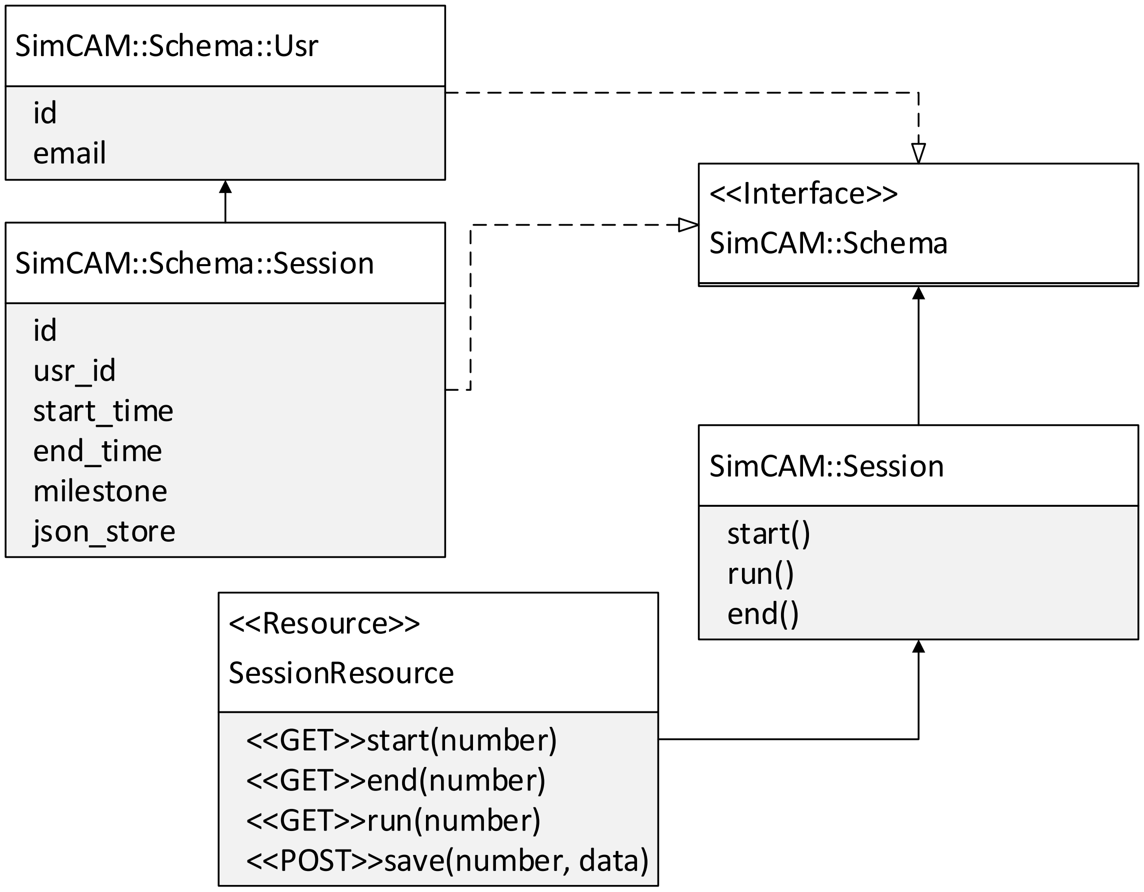


Figure 2-6: Class diagram describing session resource provide by SimCAM::Service.

Image/CV Resource

The image resource (shown in Figure 2-7) is responsible for storing and serving the image and image data resources. The ImageResource accepts a <<POST>> request of base64 [45] (a lossless encoding format) for image data and generates a server version of the image on the file system.The ImageResource then serves the image to <<GET>> requests to the resource URL. The client is able to construct encode64 image data using the HTML5 canvas tag. The computer vision resource is provided by CVResource on the /api/ URL. The CVResource provides responses to <<GET>> requests for distort, check for, calibrate (single image) and calibration (several image) operations.

CVResponse also provides sockets for bi-directional data transfer with WebSockets for <<WS>> requests. All computer vision operations are programs that run on the local version of the image (captured from ImageResource). Once these operations are completed the results are stored in a public folder and the paths sent via JSON in responses.

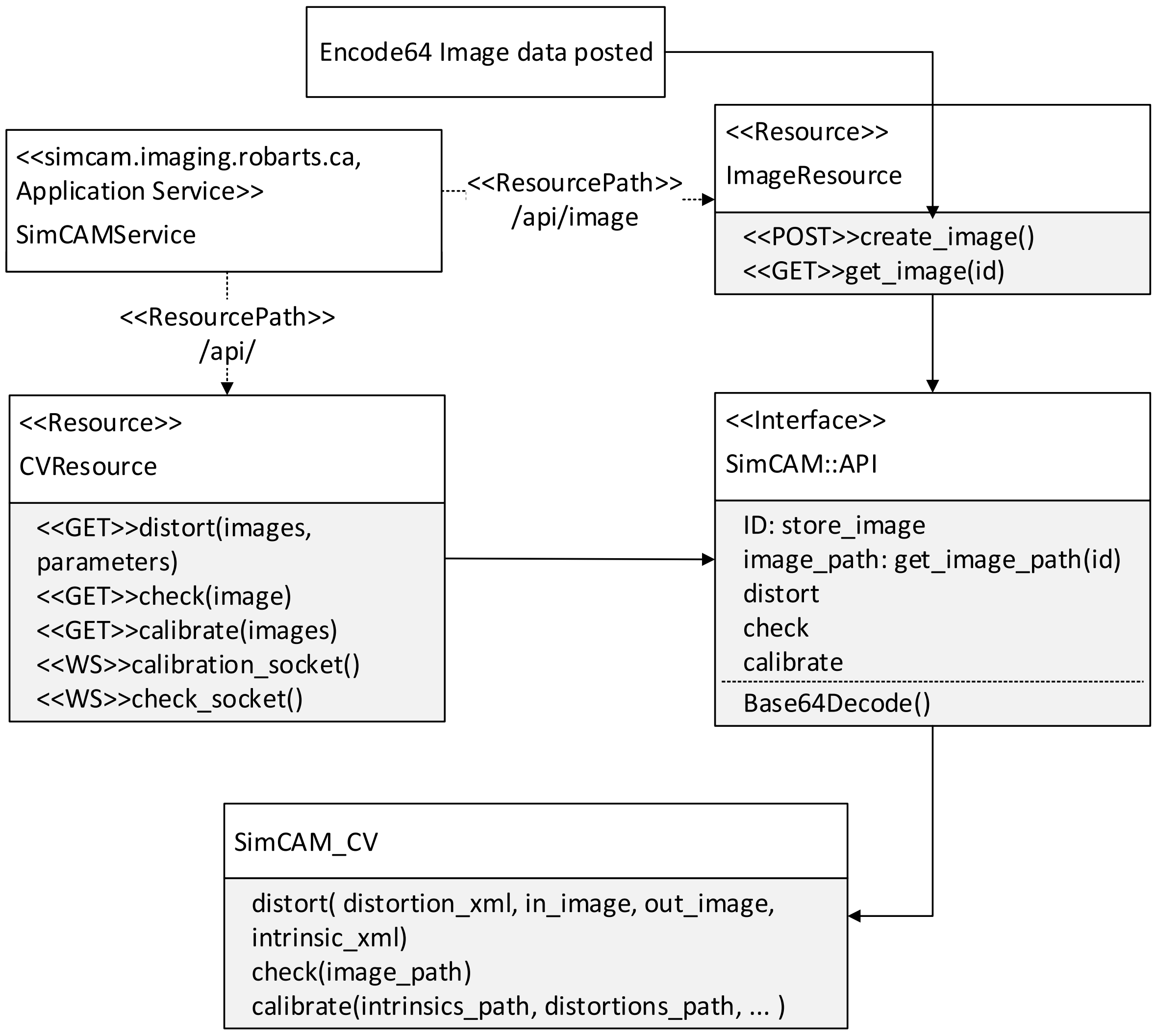

Encode64 Image data posted
<<simcam.imaging.robarts.ca, Application Service>>
SimCAMService
<<ResourcePath>> /api/image
<<Resource>>
ImageResource
<<POST>>create_image()
<<GET>>get_image(id)
<<ResourcePath>> /api/
<<Resource>>
CVResource
<<GET>>distort(images, parameters)
<<GET>>check(image)
<<GET>>calibrate(images)
<<WS>>calibration_socket()
<<WS>>check_socket()
<<Interface>>
SimCAM::API
ID: store_image
image_path: get_image_path(id)
distort
check
calibrate
Base64Decode()
SimCAM_CV
distort(distortion_xml, in_image, out_image, intrinsic_xml)
check(image_path)
calibrate(intrinsics_path, distortions_path, ...)

Figure 2-7: Class diagram describing the image and computer vision resource in SimCAMService.

Asynchronous Client/Server data interface

The SimCAM JavaScript application binds directly to resources on the SimCAMService, using the SimCAM::Model::Generic. SimCAM::Model::Generic is an implementation of Backbone::Model, where a URL resource can be defined on initialization. Once the client is initialized, the SimCAM::Model::Generic instances can request and update asynchronously with the SimCAMService (via the Backbone::Model shown in Figure 2-8). The advantage of using asynchronous connection is that the frontend client does not have to wait for the response from the server. Backbone.JS additionally provides a Backbone::Collection class that can be used to implement a collection of models, which are used for the image captures and calibrations taken by the user.

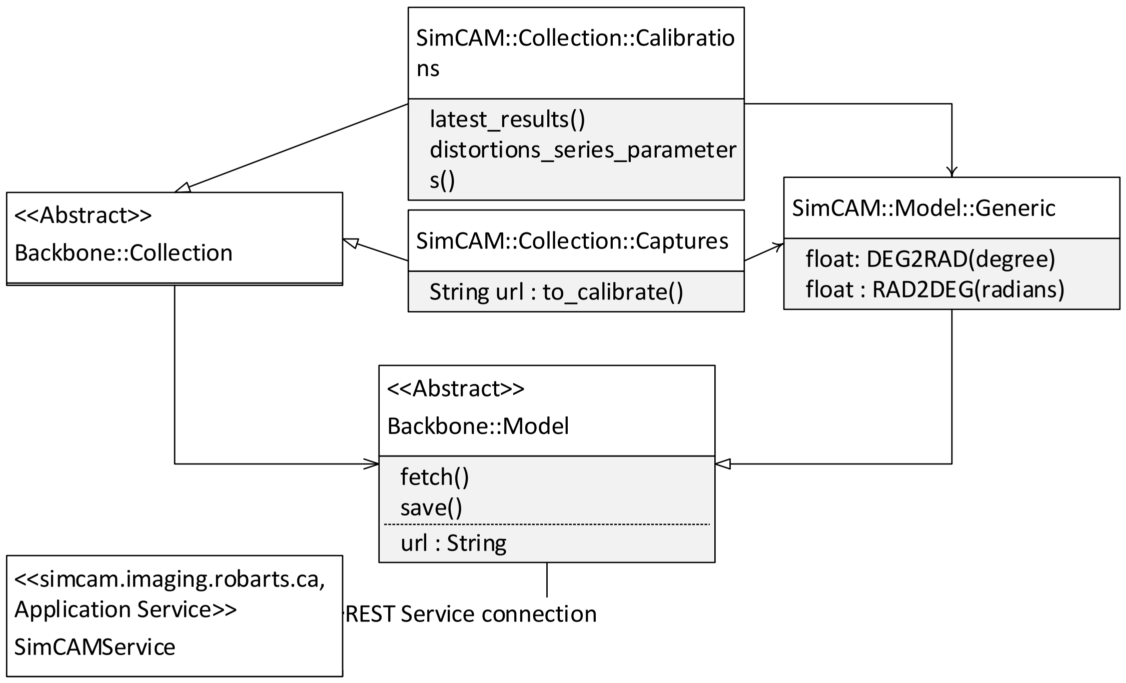

SimCAM::Collection::Calibrations
latest_results()
distortions_series_parameters()
<<Abstract>> Backbone::Collection
SimCAM::Collection::Captures
String url : to_calibrate()
SimCAM::Model::Generic
float: DEG2RAD(degree)
float : RAD2DEG(radians)
<<Abstract>> Backbone::Model
fetch()
save()
url : String
<<simcam.imaging.robarts.ca, Application Service>> SimCAMService
REST Service connection

Figure 2-8: Class diagram describing the interface between the client and service via REST and Backbone::Model and its children in the SimCAM::Collection and SimCAM::Model namespace.

Graphical User Interface

SimCAM was designed around a tutorial format, using user stories and feedback to do paper prototyping. Paper prototyping is a technique to quickly design graphical user interfaces [45]. The user logins in with their email address, which allows the system to track and store their results. The user is presented with background materials that guide them through 6 milestones.

1. Determine if the user's browser supports required technology
2. Teach user about the camera matrix and pose parameters using the pinhole camera model
3. Radial and tangential distortion tutorial and simulator
4. Camera calibration background and simulation environment
5. Web camera calibration simulator
6. Content validity questionnaire

After paper prototyping the following screen templates were generated. The graphical user interface associated with each milestone is designed to follow the simple template shown in Figure 2-9 and Figure 2-10.

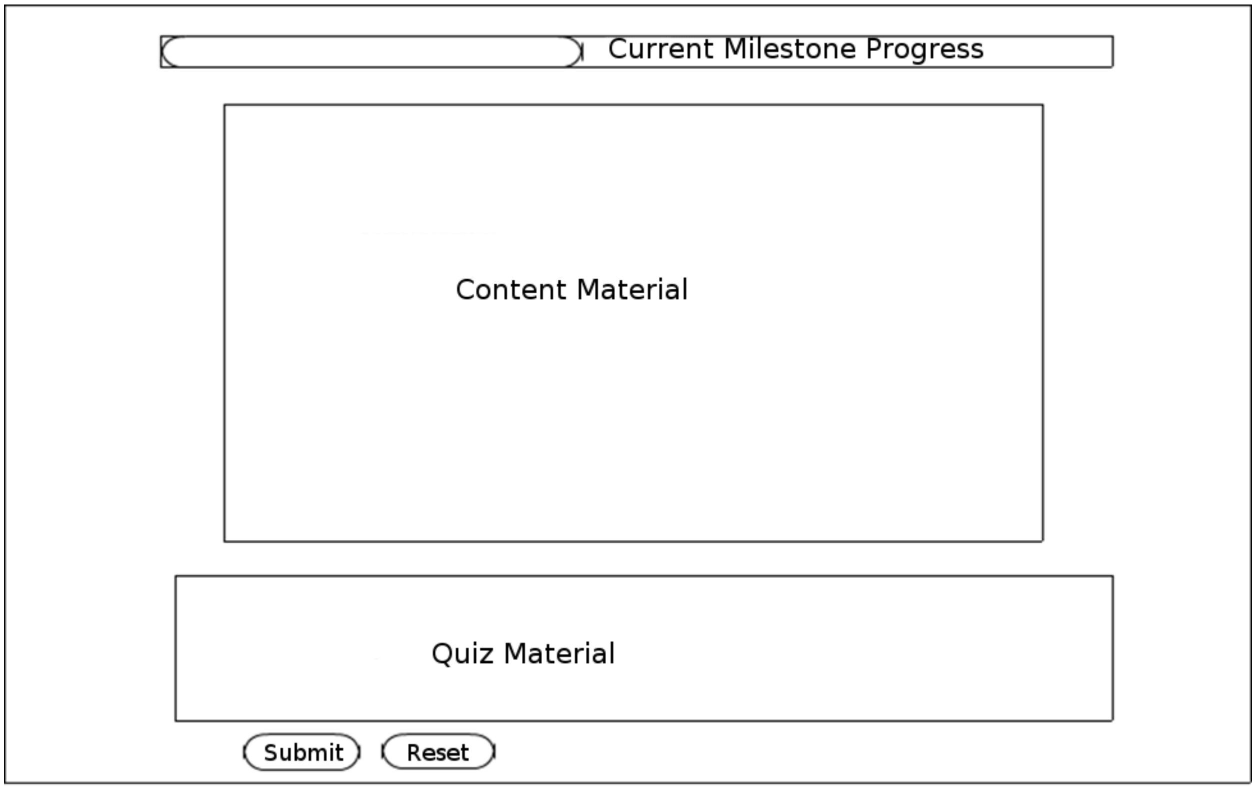


Figure 2-9: Paper prototype of basic user interface template for each milestone. The milestone progress bar shows how far along the user is in the study. The content material is shown in the center with tutorial and interactive content. Finally a quiz is presented at the bottom.

Another important GUI designed was the 3D workspace environment shown below.

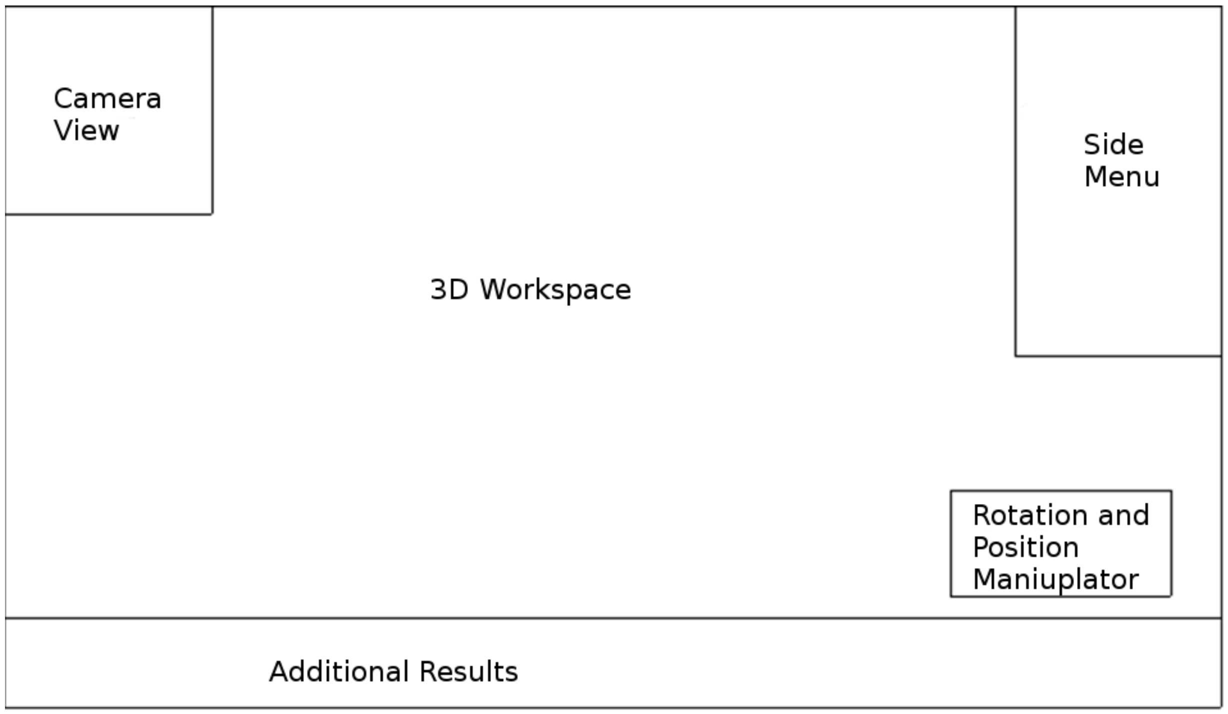


Figure 2-10: Paper prototyping of GUI Design for 3D interactive components. The camera view shows the current camera's view. The 3D workspace is where the user interacts with the camera and calibration grid. The side menu provides buttons for camera calibration results and capturing images. The captured images are shown in the additional results area. Finally the rotation/position manipulator shows the rotation and position of the current select object that can also be edited.

Since this was a pilot project, most of the design was created during the implementation period. This was accomplished by performing several iterations of implementation periods, where each iteration focused on a feature or specification.

SimCAM::View Classes

The relationship between the SimCAM::View classes, SimCAM::Router and Backbone are shown in Figure 2-11. The SimCAM::View are responsible for the following user interfaces:

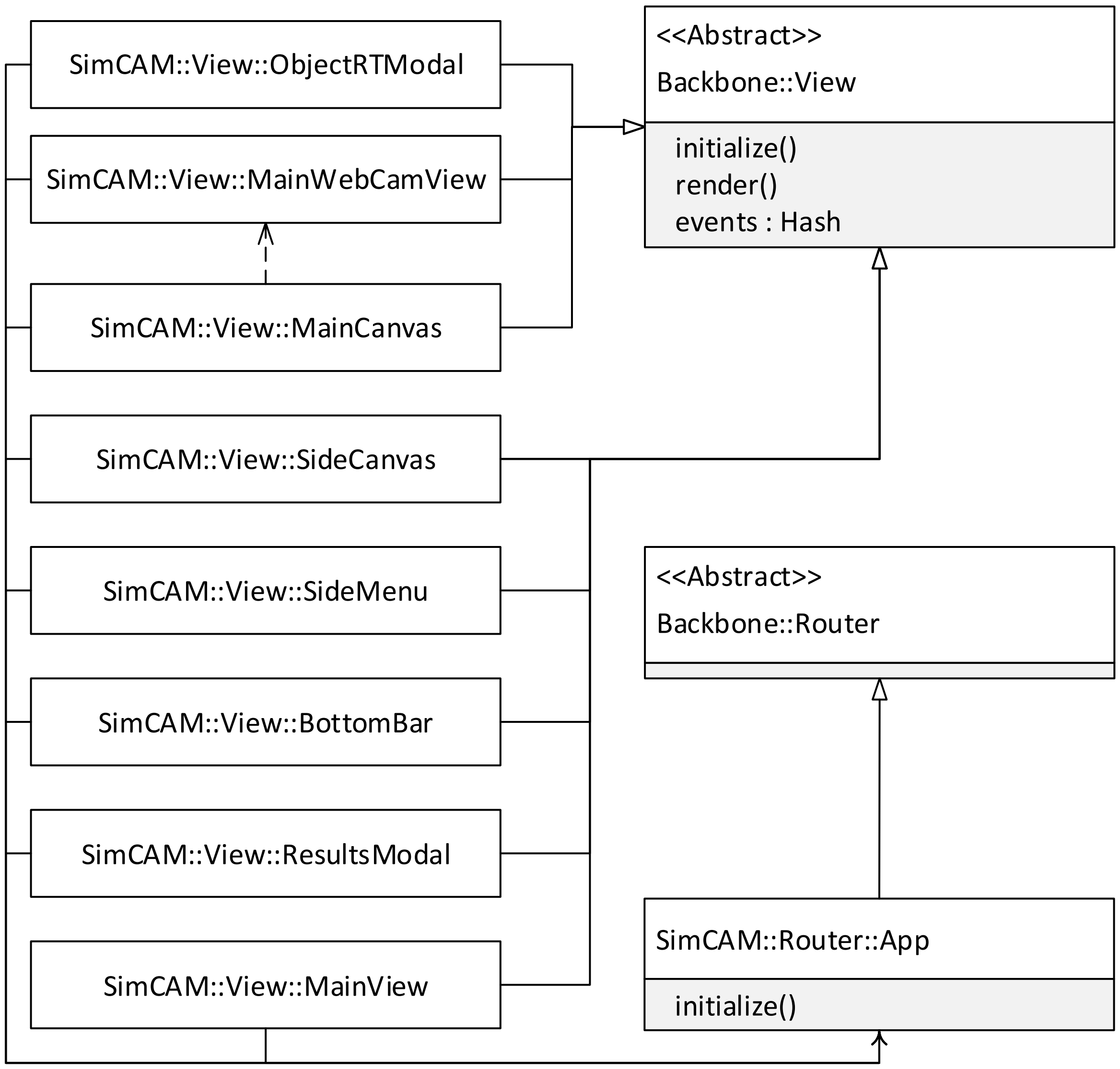


Figure 2-11: Class diagram describing the class hierarchy for views in SimCAM. All SimCAM::Views are implementations of Backbone::View and initiated by the SimCAM::Router::App.

- **SimCAM::View::MainView :** The main view is the first view to be initialized by the author and renders the structure as laid out in Figure 2-3. Creating the elements for the other views to attach too.

- **SimCAM::View::MainCanvas :** The 3D workspace and user inputs with the mouse in those areas are managed by the main canvas view. In the webcam interactive view the main canvas loads the webcam view (SimCAM::View::MainWebCamView).

- **SimCAM::View::SideCanvas :** The camera view element in Figure 2-9 is managed by the SimCAM::View::SideCanvas. This view watches for all changes on the MainCanvas to update the camera view.

- **SimCAM::View::SideMenu :** The side menu view swaps several templates depending on the parameters passed by SimCAM::Router::App. It shows the various menus for the distortion, calibration and webcam calibration milestones.

- **SimCAM::View::BottomBar :** The bottom bar holds additional results such as the captured images in the calibration and webcam calibration milestones.

- **SimCAM::View::ObjectRTModal :** The ObjectRTModal shows the position and rotation of the object currently clicked on in the main canvas.

- **SimCAM::View::ResultsModal :** The results modal shows calibration results, charts and efficacy.

SimCAM Graphical User Components

SimCAM's GUI can be broken down by logical sets of features that cover the functional specifications. The components that were implemented can be broken down into simulation, interactive, feedback, tutorial and validation components (Figure 2-12).

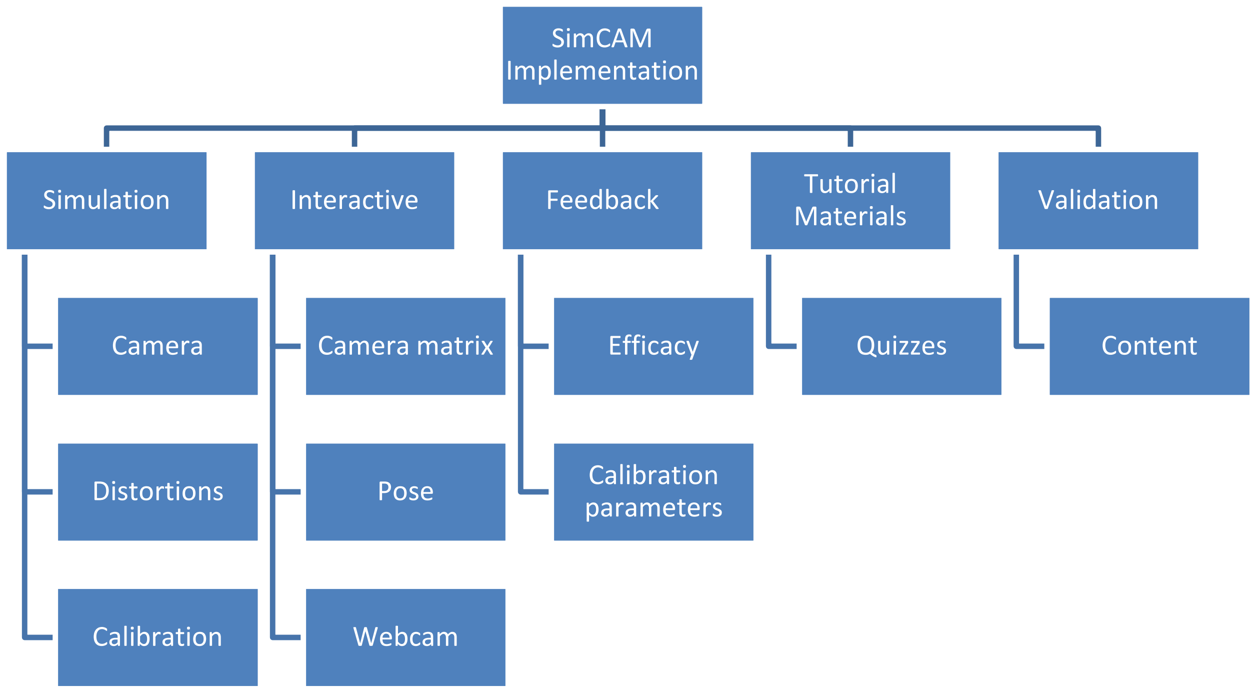


Figure 2-12: Hierarchy of implemented components of SimCAM: Simulation, Interactive, Feedback, Tutorial Materials and Validation computers

Camera

The camera was implemented using WebGL, which supports perspective cameras that can also be manipulated by users in the interface show in Figure 2-13.

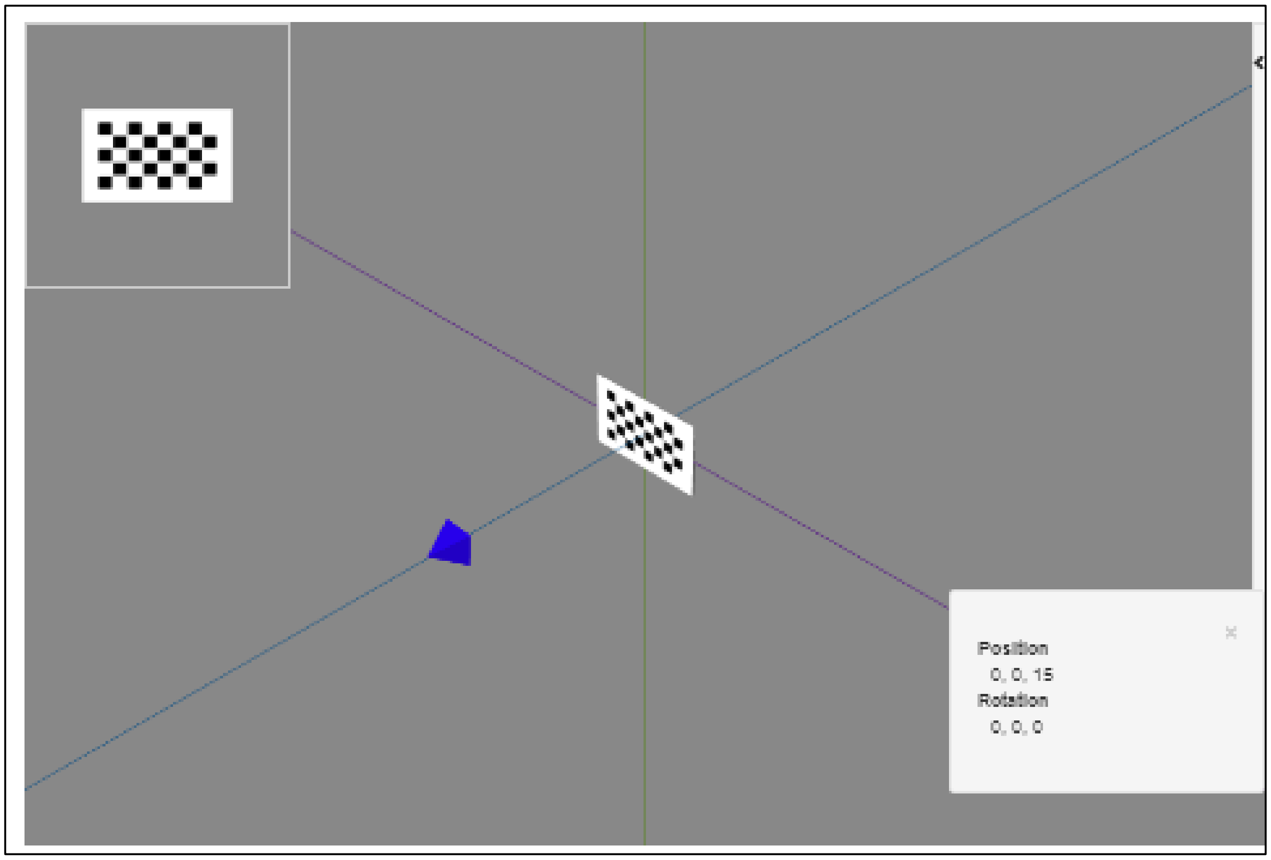
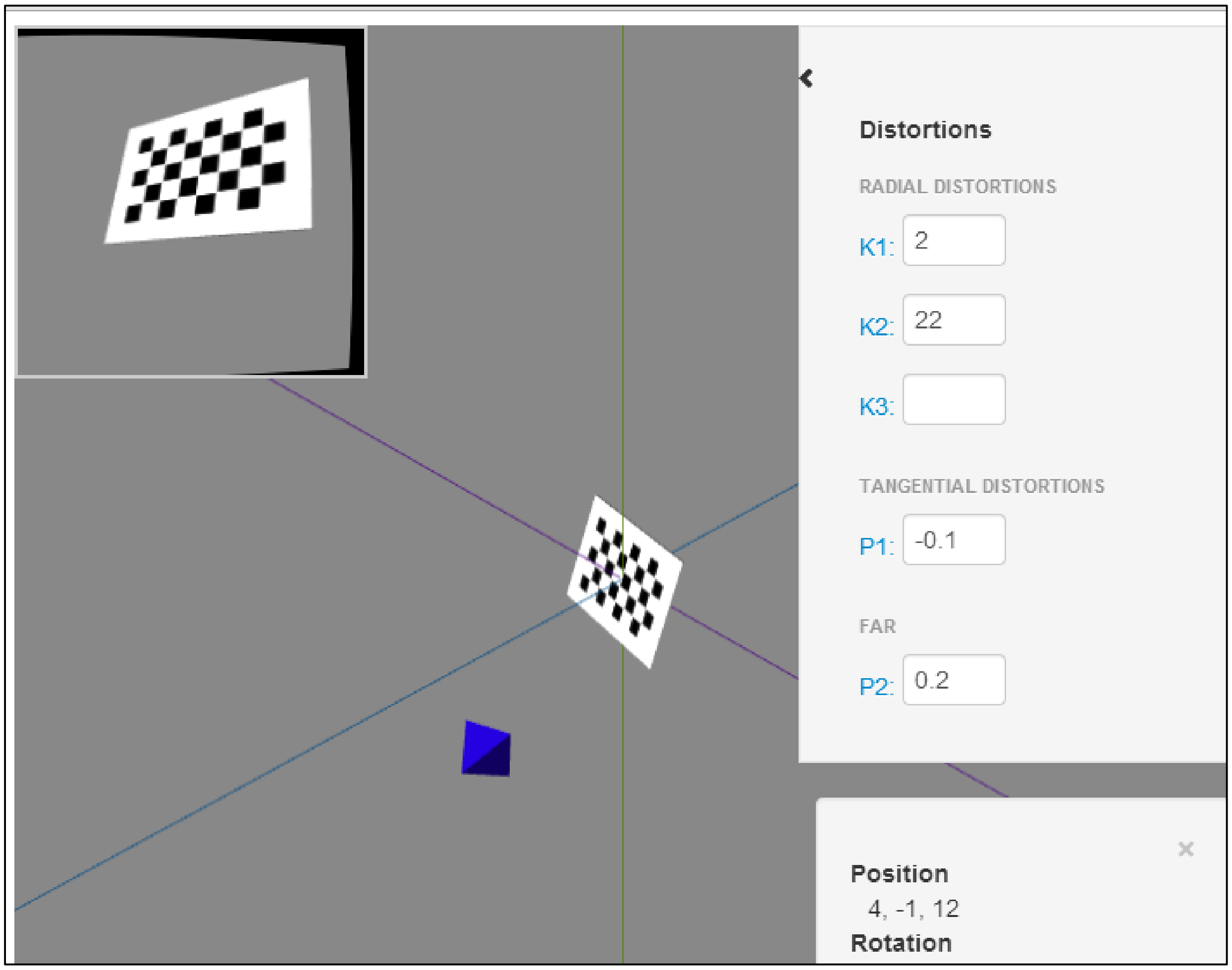


Figure 2-13: 3D interactive camera implementation, with the 3D workspace, camera view and the rotation/translation manipulator.

Controls were implemented in the 3D workspace (blue pyramid) which can be moved and rotated using the mouse. Additional controls are also provided in the bottom right widget where rotation and positions can be typed in. All the manipulations change the camera view in the top right. This is also where distortions are simulated.

Distortions

Distortions were implemented on the server side and powered by the OpenCV library (Figure 2-14). The user would be presented with several controls in the side menu (right) to change the distortion parameters. As the parameters are changed the user can see the effect in the top right.


Distortions

RADIAL DISTORTIONS

K1: 2

K2: 22

K3:

TANGENTIAL DISTORTIONS

P1: -0.1

FAR

P2: 0.2

Position

4, -1, 12

Rotation

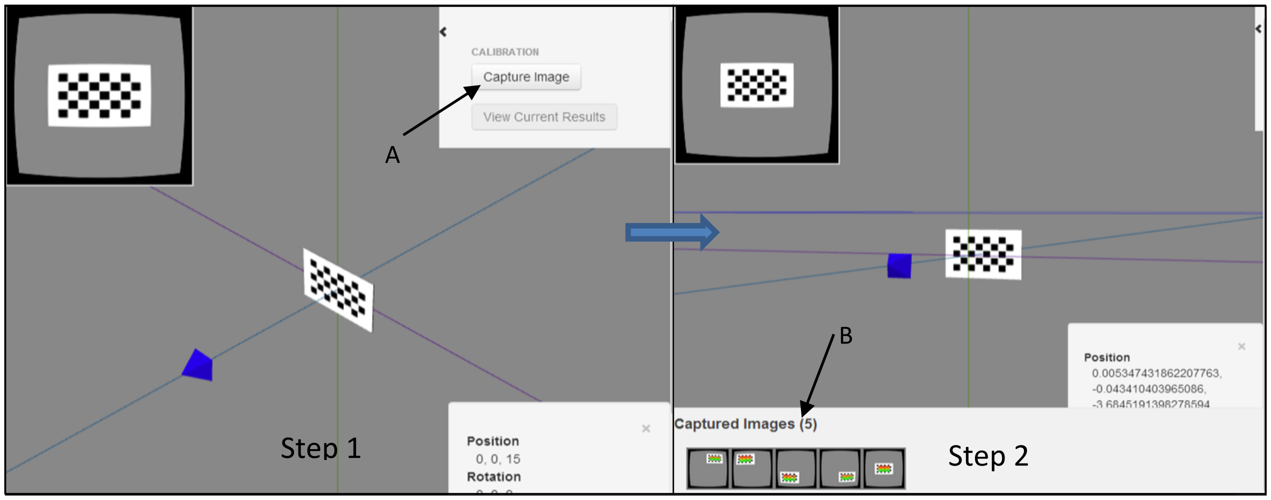
Figure 2-14: Distortion simulation with controls on the side menu where the user can adjust the parameters for radial and tangential distortions.

Image Data Serializing and Communication

Implementation of distortions on SimCAM's camera required a method to send the 3D data to the server. Image data were captured from the camera view and serialized (compressed into a text string). The serialized image data were encoded on the client side and sent via WebSockets to the server. A custom server module was made to decode the serialized image and pipe it to the OpenCV components. Once the distorted image was created it was made available on the server as an image. The client then finished the communication loop by showing the distorted image instead of the original image.

Calibration

In the camera calibration interactive component, the user was provided a random simulated camera with distortions. The camera is presented with a calibration grid (checkerboard pattern). The user is then able to move and position the calibration grid (using either the mouse or the widget), and capturing images by clicking the button labeled A in Figure 2-15. The captured images is validated by checking if the calibration grid is present in them, and shown in the bottom panel (B in Figure 2-15). The valid images are then used to perform a calibration attempt. Once a calibration attempt is done the results button is enabled in the side menu. Clicking this opens the calibration dialog which provides feedback about the current calibration attempt and compares it to previous calibration attempt. This is described in further detail in the feedback section (Figure 2-19, Figure 2-20, and Figure 2-21).



Position
0, 0, 15
Rotation

Position
0.005347431862207763,
-0.043410403965086,
-3.6845191398278594

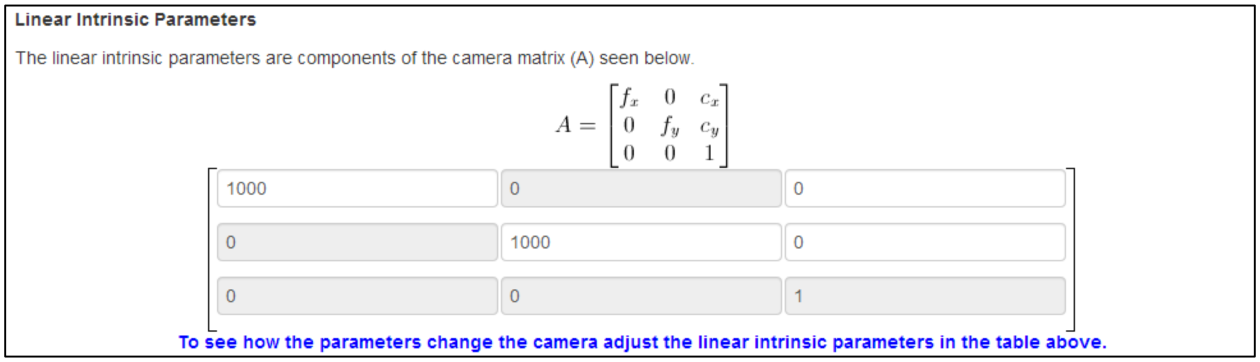
Figure 2-15: User interaction with calibration environment. (Step 1, A) Capture image with camera. Images appear in (Step 2, B).

Interactive Components

Several embedded components were developed that allowed users to interact and learn about various parameters.

Camera Matrix

The widget shown in Figure 2-16 is embedded in the 2nd milestone of the tutorial and allows users to change the camera matrix in the 3D environment. This widget was implemented by exploiting the OpenGL matrix pipeline. In this pipeline, OpenGL can push matrices that will affect the rendering of the camera view.



Linear Intrinsic Parameters

The linear intrinsic parameters are components of the camera matrix (A) seen below.

$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

1000	0	0
0	1000	0
0	0	1

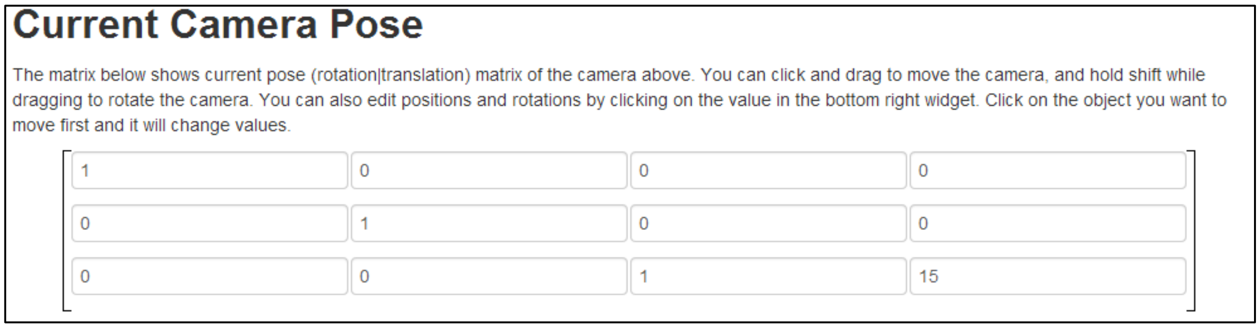
To see how the parameters change the camera adjust the linear intrinsic parameters in the table above.

Figure 2-16: Camera matrix manipulator widget. Users can change the focal length and principal points of the camera in the 3D environment above.

The widget shown in Figure 2-16 is embedded in the 2nd milestone of the tutorial and allows users to change the camera matrix in the 3D environment. This widget was implemented by exploiting the OpenGL matrix pipeline. In this pipeline, OpenGL can push matrices that will affect the rendering of the camera view.

Camera Pose

With this embedded widget (Figure 2-17) users can see how the camera pose matrix is affect when they move the camera in the environment. Also users can directly change the pose matrix here and see the result in the environment. This widget is also implemented by exploiting the OpenGL matrix pipeline.


Current Camera Pose

The matrix below shows current pose (rotation|translation) matrix of the camera above. You can click and drag to move the camera, and hold shift while dragging to rotate the camera. You can also edit positions and rotations by clicking on the value in the bottom right widget. Click on the object you want to move first and it will change values.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 15 \end{bmatrix}$$

Figure 2-17: Camera pose manipulator. The user can adjust the rotation and translation matrix directly to the 3D environment in the same milestone.

Webcam

Users are also able to use a calibration grid to calibrate their webcam (Figure 2-18). Once several images are captured, users can see the calibration parameters, charted values as described in the feedback section (Figure 2-19, Figure 2-20, and Figure 2-21). This component was implemented using HTML5 video and webcam components.

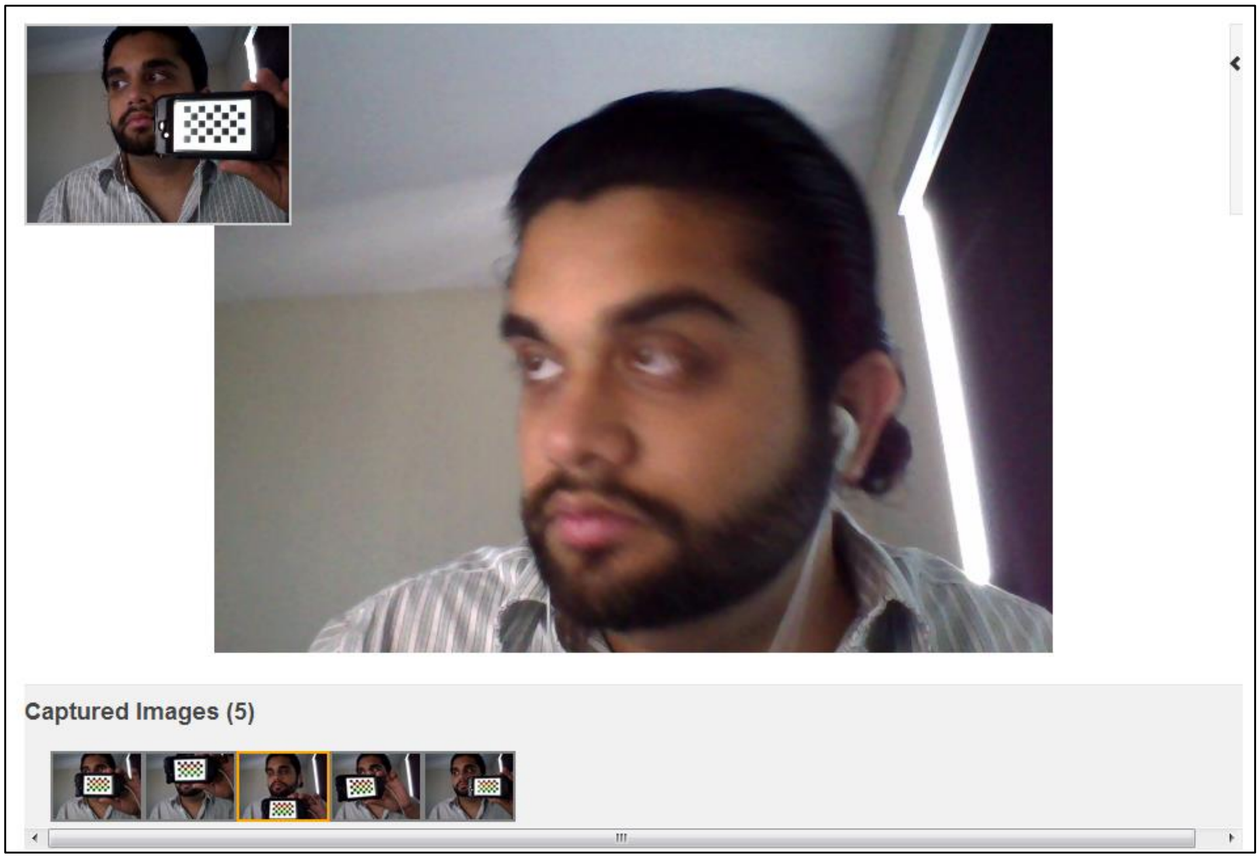
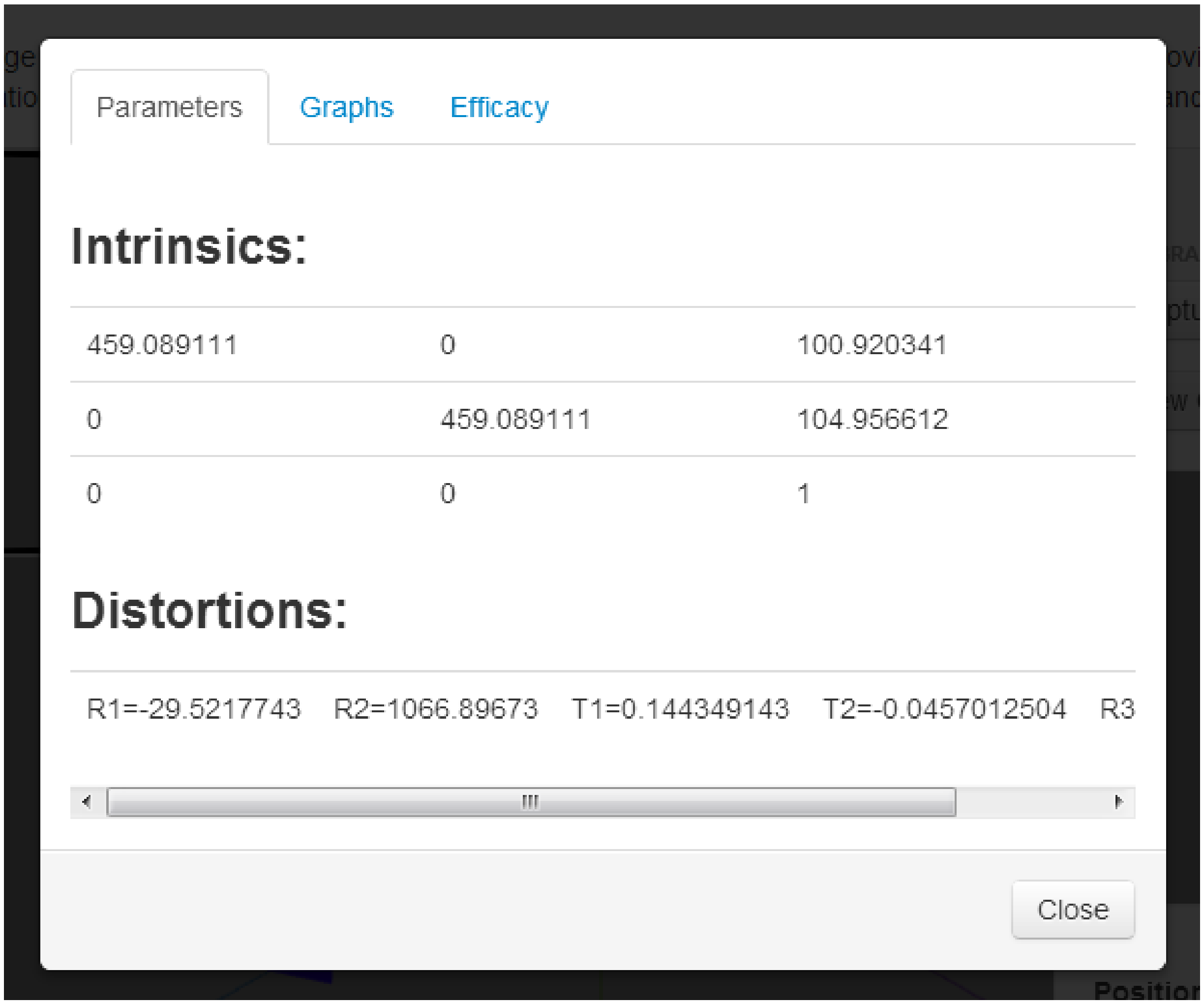


Figure 2-18: Interactive webcam component that allows user to calibrate their web camera. They use their phones or printed paper to display an 8x5 calibration on their web camera. By clicking the capture button similar to the calibration environment in Figure 2-15. The captured images are shown in the bar on the bottom and the calibrated results in the results dialog (Figure 2-19)

Feedback

When the user finishes a camera capture and subsequent calibration, a series of feedback screens are provided. These feedback components are available in popup dialog boxes (Figure 2-19) that becomes available by pressing the Results button in the side.



Intrinsics:

459.089111	0	100.920341
0	459.089111	104.956612
0	0	1

Distortions:

R1=-29.5217743 R2=1066.89673 T1=0.144349143 T2=-0.0457012504 R3

Figure 2-19: Feedback popup dialog which shows the calibration parameters, graphs (Figure 2-20) and efficacy (Figure 2-21) of the current calibration attempts

Efficacy

Difference images of the undistorted image and the corrected image show how well a camera was calibrated. An ideal difference, were the image was perfectly calibrated, would show as a completely black image (no difference). Thus by reviewing the difference image users were provided quick feedback on the efficacy of each of their calibration attempts. These difference image are available in the post calibration dialog (Figure 2-20).

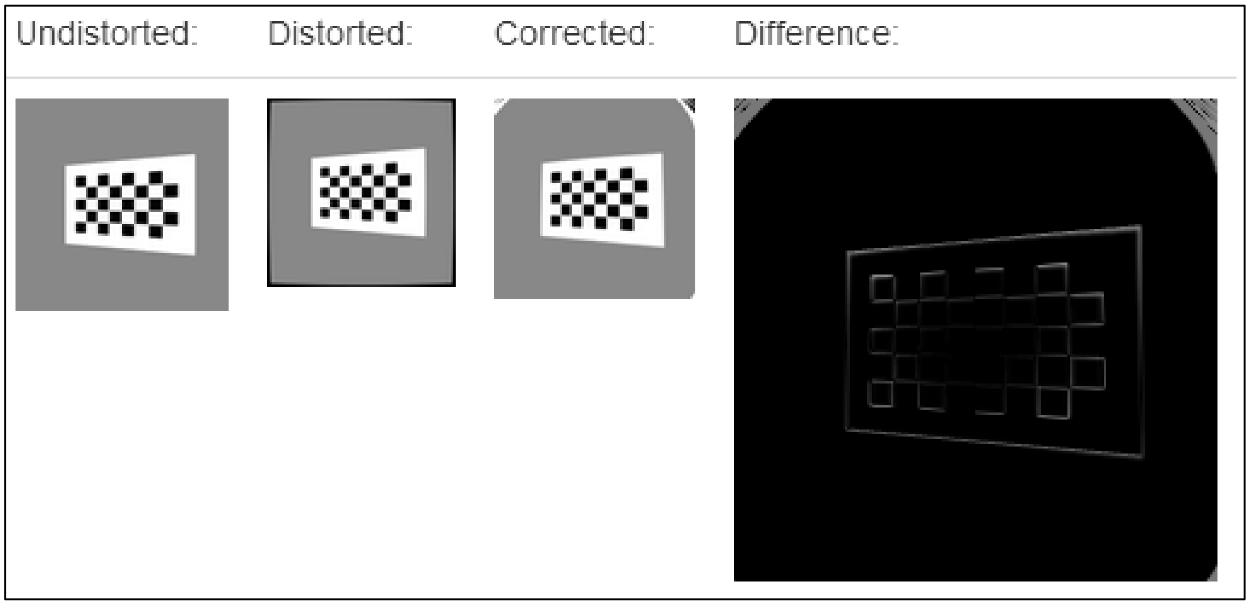


Figure 2-20: Example of differencing the corrected and undistorted images. In the simulation environment the undistorted images are simulated camera views before the simulated distortion is applied. The distorted images are with simulated distortions applied, and the corrected images are with the calibration parameters used to correct the distorted image. The final efficacy of the calibration is based on the difference image of corrected and undistorted image.

Charting calibration parameters

The current calibration parameters are available on the first tab of the popup dialog (Figure 2-19) and also as an interactive line graph chart (Figure 2-21). This line graph shows all the calibration parameters for each calibration attempt, and shows how each parameter value eventually converges.

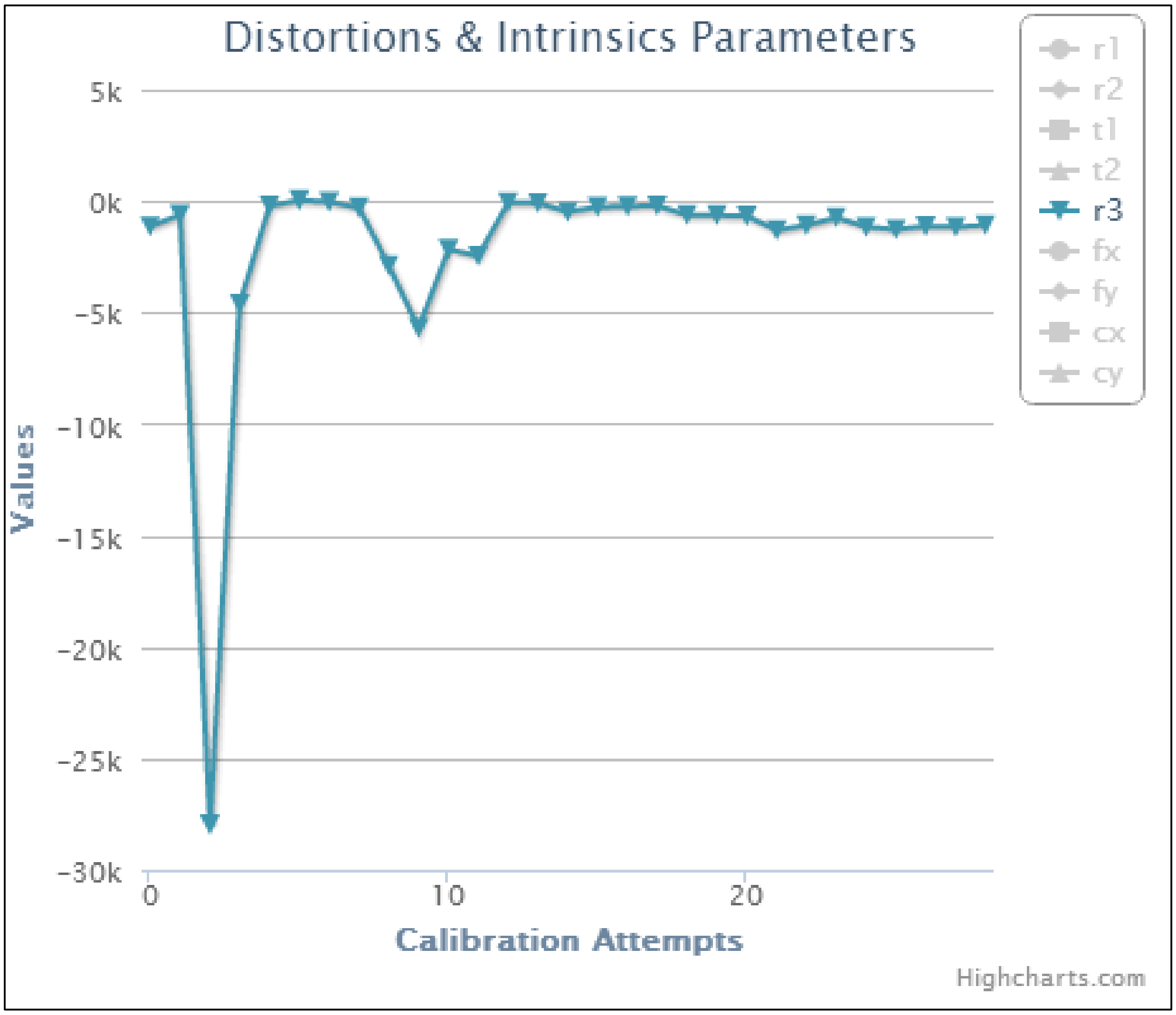


Figure 2-21: Example of charted calibration parameters. Clicking on the legend values enables each series

Content Validity

Content validity measure the completeness of the simulator in comparison to the natural processes it is simulating. Content validity is assessed by asking experts experienced with camera calibration to rate the simulator for its ability to perform the intended functions (such as training of camera calibration). The content validity study was implemented as a questionnaire that was presented to users. The questionnaire (described below in section 0) captured results that were stored in a database (described in Chapter 3).

Testing & Validation

In SimCAM two types of testing were completed: software testing and content validity. SimCAM's software was tested with manual integration test cases and automated unit test cases that tested major views of the system. The user acceptance testing (UAT) was performed in the format of a content validation questionnaire. The content validity study was used to gauge if SimCAM covers the required camera calibration subject, verify design decisions and the explore value of certain features. The questionnaire (Appendix A) has six sections.

- The first section captures demographics of the participant
- The second section covers questions about specific content in the simulator (discussed in the implementation section 2.4)
- The third section covers yes and no questions that focused on determining if the participant agreed with the specific direction of SimCAM
- The fourth section determines if SimCAM is appropriately difficulty
- Finally free form comments are captured

The results of the content validity are discussed in the Results chapter.

Participants

The SimCAM content validity questionnaire targeted both expert and non-expert (who would be end-users of the systems) participants [46]. Subject matter expert participants were included to determine if enough content was covered and valid to act as a training resource. Non-experts, on the other hand were included as they would provide the end-user perspective of the system. End-user perspective is critical because they would not know common assumptions (that experts would know) in the camera calibration material.

Quiz Results

Throughout each milestone users were quizzed with several multiple choice questions reviewing the material just presented. The quizzes were made to be a quick review, to check if the user was able to understand the material. It was expected that the expert users should have a higher score on average than the non-experts, helping to further differentiate the two groups.

Demographics

The content validity study also collected the following demographics, which were used to differentiate the expert user from the non-expert:

1. What is your area of study/program?

2. How many hours have you been awake?

3. What is your experience in number of years with computer vision/graphics?

4. What is your experience in number of years with linear algebra?

5. If you have done camera calibration in the past, about how many different times have you done it?

Questions 1 and 3 to 5 were included to determine the experts in the pool of users. Question 2 was included to exclude unexpectedly bad quiz results, for example, if the user was awake for too long.

Content Validity Questionnaire

On the last milestone, users were given 4 sections (sections A-D) that generated results to validate the content of SimCAM. Section A presented users with 12 questions (Appendix A) and users were asked to respond by circling a number on a 5-point Likert scale where:

- 1= Strongly Agree
- 2= Agree
- 3= Neutral
- 4= Disagree
- 5= Strongly Disagree

The Likert scale is a psychometric scale used frequently in studies that are dependent on questionnaires. The scale runs from an extreme to another opposite extreme, with a neutral point in between. This scale is a tool to measure attitudes [47], and used in this study to measure attitudes about the content of SimCAM.

Specific Content Validity Questions

Users were asked the 12 questions to measure the attitudes using the Likert scale:

Questions 1 and 2 covered the motivation of SimCAM. Users were asked if they agreed that having a self-directed and structured tutorial for learning camera calibration is valuable. Question 3 asked if SimCAM was a good introduction for camera calibration.

Question 4 also asked if SimCAM improved understanding of camera calibration.

Question 5 to question 10 asked participants if the various implemented components listed below were valuable for learning camera calibration.

5. Simulation of pin-hole camera model, camera matrix and pose matrix (Figure 2-13, Figure 2-16, and Figure 2-17)
6. Simulation of the radial and tangential distortions (Figure 2-14)
7. Simulation of the calibration task (Figure 2-15)
8. Feedback provided with the line graphs (Figure 2-21)
9. Feedback provided with difference images (Figure 2-20)
10. Webcam calibration component (Figure 2-18)

Finally, the last two questions asked if participants would continue to use SimCAM and if they would recommend it to beginners.

In Section B, the participant was asked 4 (Yes/No) questions:

1) Is simulating individual aspects (e.g. rotation matrix, distortion parameters, etc.) of the camera calibration task valuable for training?

2) Is simulation of the overall calibration task in an ideal environment important for camera calibration training?

3) Should software programming using the OpenCV library (used for calibrating images) be included in the tutorial for camera calibration training?

4) Is the amount of simulation you were exposed to in the environment sufficient for an introduction to camera calibration?

The first two questions were included to capture users' opinions on the motivation behind SimCAM. Since OpenCV is a commonly used library for camera calibration, question 3 was included to determine if expert users feel a software programming tutorial is needed. Question 4 was presented to determine if user's felt there was enough fidelity in the simulator.

Section C asked expert users to compare the difficulty of the simulator and a real camera calibration procedure. The difficulty of the simulator should be low enough to allow non-experts an acceptable learning curve, but high enough to represent common challenges in camera calibration.

Section D was used to qualify certain questions in Section A based on if the simulator and web-cam technology worked as expected.

Free form comments were recorded from users, which let them provide additional insight and critique of the content.

Analysis

To begin analysis of the results, the experts and non-expert samples needed to be separated. Differentiating the non-expert and expert users was accomplished using the demographics and quiz. The questionnaire results were analyzed to determine the overall content validity of the system. The specific content questions in Section A were scaled (Figure below) to quantify users' agreement to the question.

1= Strongly Agree (100%)

2= Agree (75%)

3= Neutral (50%)

4= Disagree (25%)

5= Strongly Disagree (0%)

Once quantified an overall agreement score is generated for the expert and non-expert users. An arbitrary agreement of over 75% was considered as valid content.

Since Section B consisted of Yes/No answers, their statistical mode were used to determine if the experts and non-experts users' agreed with the question.

Section C results were used to determine if the simulator was too difficult for users to use. Should the Section A results indicated a low content validity, the Section C results can be used to determine if the cause was due to too high of a learning curve. Section D was also used to qualify the Section A results, where questions where dependent on technology working.

3 Results

3.1 Introduction

SimCAM was made available online[2] and users finished quizzes and the content validity study. Overall, 13 users who meet the criteria as participants completed the study. The first type of data collected on the participants was the total time spent on the study. Using these data, 2 participants were excluded as they completed the study too quickly, suggesting they did not read the tutorial materials. There were more non-experts (7) than experts (4) in this study. The remaining participants were then separated into groups of experts and non-experts. This determination was made based on the captured demographics information. Results of the remaining content validity questionnaire (See Appendix A) were also recorded. The statistical analysis of results of each section are presented below. Finally the result of each quiz in the milestones of the tutorial was also available.

3.2 Time Spent on Tutorial

The overall time spent was calculated as a sum of the time spent on each milestone in the tutorial. The time spent data indicates the presence of outliers as the mean was 51.4 minutes with a standard deviation of 58 minutes. Reviewing the results was helpful in eliminating 2 participants that had simply skimmed the study as indicated by their extremely low (compared to the mean) time spent (2 and 4 minutes) on the tutorial. The excluded participants also selected the same answer for each multiple choice question and had nonsensical answers for written answers.

3.3 Content Validity

The content validity results were based on Section A of the questionnaire. The experts and non-experts (Figure 3-1) both validated that the content of SimCAM was appropriate and valid for

[2] http://simcam.imaging.robarts.ca

teaching camera calibration. Experts had an average of 74.6% (17.5% STD) agreement with Section A questions that the content was valid. Non-experts had an average of 85.7% (9% STD) agreement that the content was valid.

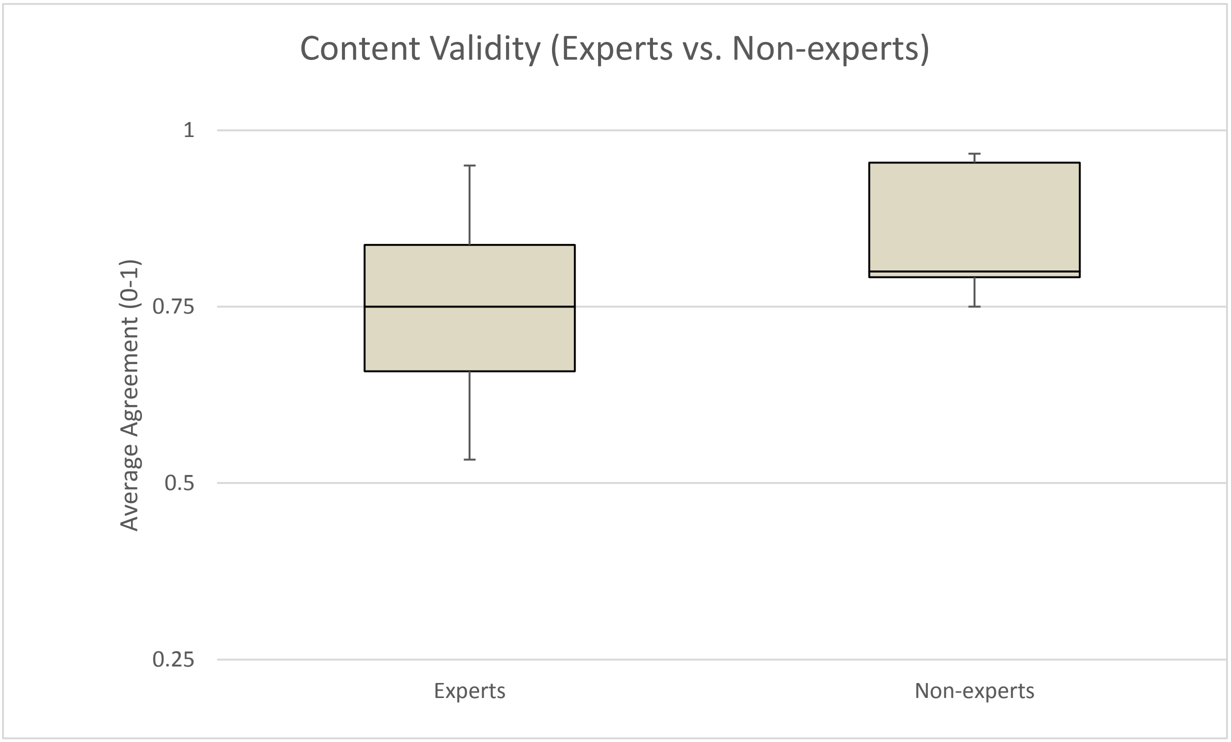


Figure 3-1: Overall content validity score between Experts vs. Non-experts. The results indicate that SimCAM has valid content to be a training environment for beginners

3.3.1 Section A: Specific content validity questions

Due to the small number of participants, the possibility of a skewed dataset is high. The outliers in the dataset may result in a misleading standard deviation and standard error of the mean. In a similar study the median and mean were compared to determine an absence or presence of outliers [48].

QUESTION	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**[3]	**11**	**12**
MEAN	1.73	1.64	2.09	2.00	1.82	1.82	2.00	2.45	1.73	1.6	2.00	1.91
MEDIAN	2	2	2	2	2	2	2	3	2	2	2	2

Table 3-1: Mean vs. Median of each question in Section A

The mean and median for the all participants' answers are similar, suggesting an absence of outliers. However, the non-experts were more positive than the experts (Figure 3-1 and Figure 3-2).

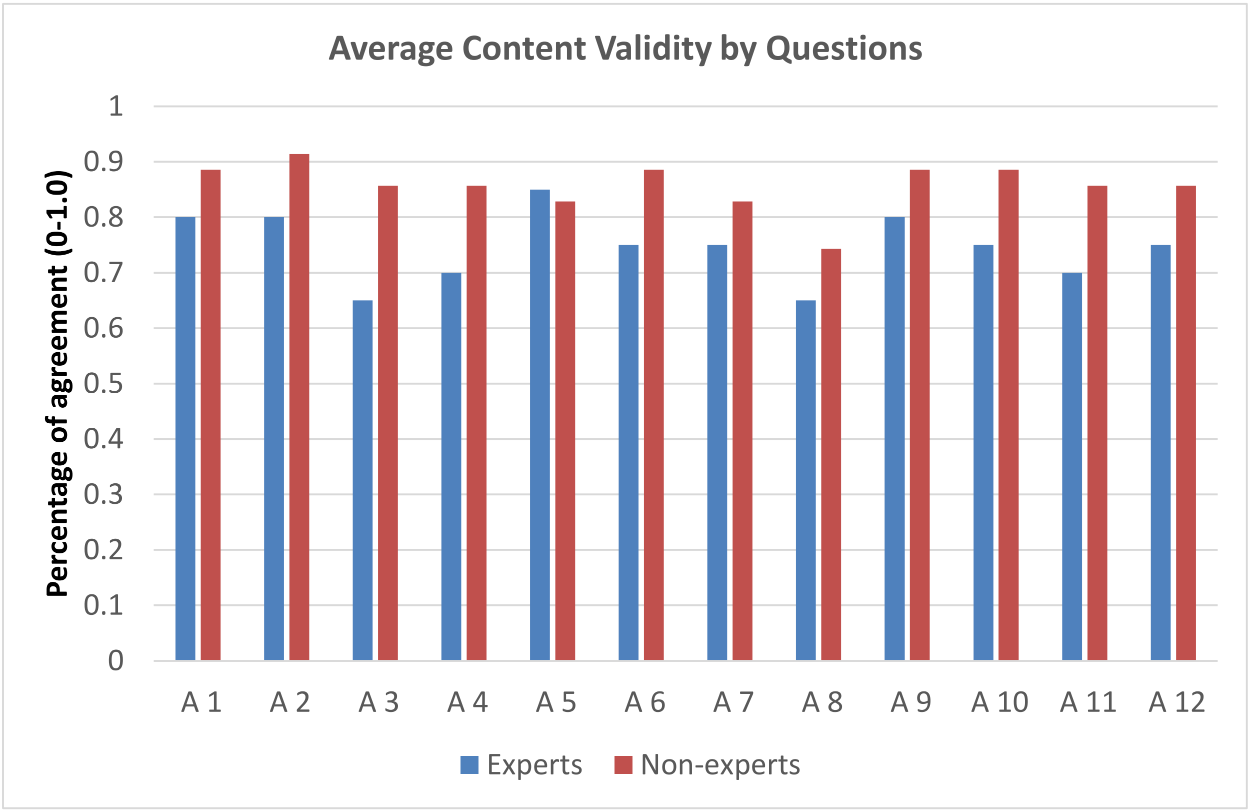


Figure 3-2: Mean Section A (specific content validity questions) answers compared between experts and non-experts

[3] Ignoring 2 results as the webcam didn't work for the participants. This question asks about the webcam.

Results for questions 1 and 2 showed that both experts and non-experts agreed that these training tools are valuable to have.

From the results for question 3 (the system is a good introduction to camera calibration), it appears that non-experts agreed that SimCAM was a good introduction for camera calibration. However, experts agreed less and reviewing their comments indicated that they were focused on the presentation of the tutorial content, rather than the actual simulation components for this question. Some experts mentioned several additional materials that should be included with SimCAM. In future work, these comments will be used to improve SimCAM's coverage. Results for question 4 indicated that non-experts and experts agreed that SimCAM did improve understanding of camera calibration. Recall that question 5 to question 10 asked participants if the various implemented components listed below were valuable for learning camera calibration.

5. Simulation of pin-hole camera model, camera matrix and pose matrix (Figure 2-13, Figure 2-16, and Figure 2-17)

6. Simulation of the radial and tangential distortions (Figure 2-14)

7. Simulation of the calibration task (Figure 2-15)

8. Feedback provided with the line graphs (Figure 2-21)

9. Feedback provided with difference images (Figure 2-20)

10. Webcam calibration component (Figure 2-18)

Except for the line graph feedback mechanism, both non-experts and experts strongly agree that the components were valuable for camera calibration training. The experts especially agreed that the camera, camera matrix and pose matrix simulations were valuable for camera calibration training. It is unclear why participants thought the line graphs feedback of camera calibration parameters (Figure 2-21) were less valuable for training. Perhaps a follow-up with participants would reveal the reason, as there were no comments about the line graphs. The line graphs were perhaps too abstract, which would require the user to spend more time to understand results (decreasing the feedback value). A potential solution to this would be to normalize the line graphs, to highlight changes in the values rather than the actual values themselves. Another option would be to move the line graphs directly next to the 3D interactive component (Figure 2-10), rather than hidden away in dialog box. By moving the location of the line graphs, perhaps users can pick up on more hints and understand the intent of the line graph (convergence of values indicating that more calibration attempts are unnecessary or may even decrease accuracy).

Finally, the last two questions asked if participants would continue to use SimCAM and if they would recommend it to beginners. The results suggested that the non-experts would recommend it and continue to use SimCAM. The experts' results suggested that they were less inclined than non-experts to recommend it to beginners and to use SimCAM. Rewording the tutorial content, with a quorum (agreement on content) between current experts, may change these results.

3.3.2 Section B: Determining attitudes on design decisions

In Section B, users were asked 4 (Yes/No) questions to determine attitudes on SimCAM design decisions that are meant to be used to reveal further insight. Results are summarized in Table 3-2. All users agreed that the individual simulation components were valuable for training. Except one expert, all users believed that the overall calibration task was an ideal environment, important for camera calibration training. The one exception mentioned in additional comments that:

"The interface might be useful for a beginner who does not have background in camera geometry. It would be better if the uncertainty in parameter estimation can be plotted."

Indicating that the expert believed that the overall simulation component may be useful for a beginner (which was the intent of the question). Therefore, the users believed that design decisions for the presentation of camera calibration components, were appropriate.

The results for question 3 and 4 were split and inconclusive for the overall user population. Question 3 asked users if they believed a tutorial module for OpenCV programming was required. Question 4 asked users if the simulation provide was enough as an introduction to camera calibration. A second look reveals that experts (3 yes) believed a tutorial module for OpenCV software programming is necessary for SimCAM.

On the other hand 3 out of 4 experts recorded "No" for question 4. The question 4 results, taking into consideration the Section A results, indicate that although the SimCAM pilot is a good start, more simulation components (such as uncertainty propagation and OpenCV programming) are required. These results can be used to capture further specifications to improve SimCAM. The captured specifications from the content validity will be used for the next iteration of SimCAM development.

Question	Yes	No
1	11	0
2	10	1
3	6	5
4	5	6

Table 3-2: Results for Section B (questions regarding attitudes on design decisions)

3.3.3 Section C: Measuring difficulty of simulator

Section C asked the user to compare the difficulty of the simulator and a real camera calibration procedure. All experts agreed that the difficulty of the simulator was much less than the actual camera calibration procedure. This may indicate that the simulator is too easy and that additional implementation (OpenCV programming module, uncertainty propagation simulation) will be needed to challenge students. However, the purpose of SimCAM was to provide an introduction to camera calibration and as such, the difficulty of the material may be appropriate. This can be further investigated by conducting a study that gauges participants' knowledge transfer from the simulator and training materials to the actual camera calibration task.

3.3.4 Section D: Ensuring all simulation components worked

Section D was used to qualify certain questions in Section A, based on whether or not the simulator and webcam technology worked as expected. Out of the 11 participants, the simulator worked as expected for all of them. The webcam technology worked for only 9 participants. The participants for which the webcam did not work had their answers for Section A, Question 10 ignored in the results above.

3.3.5 Additional Comments

In the final section of the questionnaire (Appendix A), participants were allowed to add additional comments. Participants noted that SimCAM was a good first step (“It is a good first step”, and “Very good tutorial”).

Finally, participants also saw an opportunity to add more simulation components, especially displaying how calibration affects the uncertainty (uncertainty propagation) of simulated camera systems and applications. These comments will provide a launch board into future work to improve SimCAM.

3.4 Expert Determination

Expert users were selected based on their experience with performing camera calibration, field of study, background training and quiz results. Users were determined to be experts if they had completed more than 5 camera calibrations, while the non-experts had no experience performing camera calibration. The experts had a mean of 36.3 calibration attempts with a standard deviation of 43 calibration.

Additionally, the training background of the users was used to determine if the experts and non-experts were classified correctly. The experts' response of their training backgrounds were in Computer Vision and Biomedical Engineering (Medical Imaging) equally and was as expected. The non-experts responded that they were training in a variety of backgrounds: 5 from Computer Science, 2 from Biomedical Engineering, and 1 from Medicine.

These results indicate that the experts were trained in the fields that would require solid knowledge of camera calibration, as expected. Additionally the variety in training backgrounds of non-experts indicates a good representative sample of expected end-users.

Additional results of reported computer vision and linear algebra training shows a clear distinction between the experts and non-experts.

Training in Computer Vision & Linear Algebra

Reviewing specifically the years of training in computer vision and linear algebra of all the users help to confirm the expert selections. The experts had on average 16 years (standard deviation 11.4 years) of training in computer vision, while non-experts had an average of 4.5 years (standard deviation of 5.2 years). The experts had on average 7.2 years (standard deviation 11.1 years) of training in linear algebra, while non-experts had an average of 2.9 years (standard deviation of 3.8 years). Comparing the experts and non-experts (where experts have at least 5 calibration attempts), shows that experts had significantly more years of training in the reported computer vision and linear algebra years in training results strengthening the selection of experts from the user pool.

3.4.1 Quiz Results

Throughout the tutorial, the participants answered simple multiple choice questions that quizzed them on the material just presented. Overall, 11 questions were asked in 3 tutorial milestones. Comparing the quiz with the time spent on the study seemed to show a slight correlation (Figure 3-3, Pearson's correlation p=0.73), indicating that higher scores correlated with longer time:

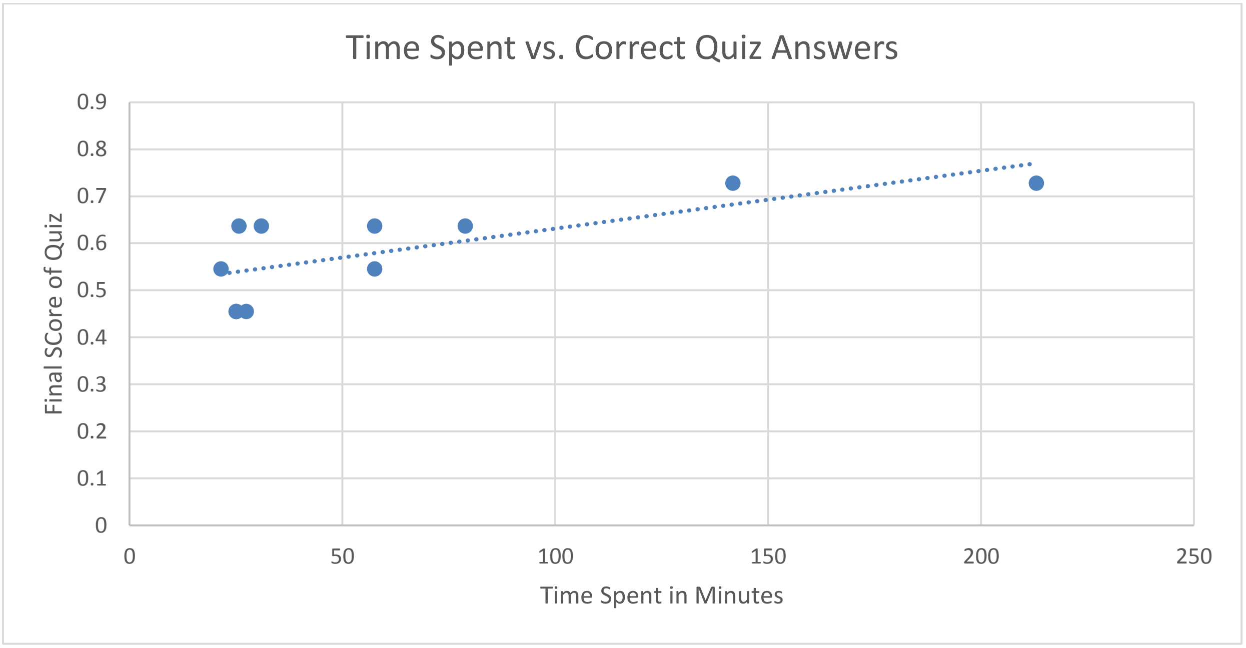


Figure 3-3: Time spent on study vs. correct answers in the quiz, p=0.73.

Additionally the participants' quiz did not seem affected by how long they had been awake (Figure 3-4, Pearson's Correlation p=0.36). It was strange that none of the participants scored 100% on the quiz and even more unexpected was that the experts' average score (56.8%) was lower than the non-experts' average (59.7%). The number of waking hours did not correlate well with the score (Figure 3-4, p=0.36), and could not explain the low score results. After reviewing the quiz results and following up with experts' additional comments, several ambiguities were noted in the tutorial material that would need to be resolved.

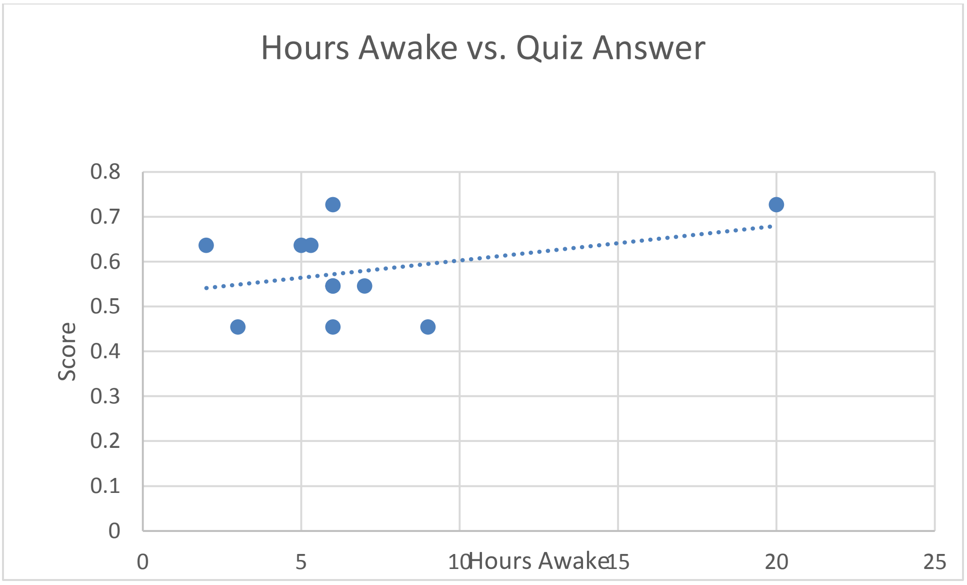


Figure 3-4: Hours participant was awake vs. quiz score, p=0.36

4 Conclusion & Future Work

A pilot web application (SimCAM) and an application service was implemented for training camera calibration. User stories from end users were gathered and distilled into specifications, and functional specifications were created that describe the specific features of the system. Non-functional specifications were generated to define the expected behaviour of the system. Using these specifications, a client and server architecture was selected for SimCAM.

SimCAM integrates simulated and web cameras with camera calibration algorithms (currently only OpenCV), while providing teaching materials in a tutorial format. WebGL, Three.JS and Backbone.JS libraries were used to create 3D interactive components on users' web browsers. Using the REST and WebSocket protocols, image data were shared with a custom service. The service was implemented using Mojolicious, OpenCV and DBIx::Class ORM.

The Mojolicious framework was used to describe and drive the SimCAMService. SimCAMService was implemented and made available as a URL, providing session, image and computer vision resources, were accessible to via SimCAM::Model package to the rest of the client application. The client application code had several views responsible for interactive components.

Once completed, SimCAM provided users with interactive components for experimenting with a perspective camera's camera matrix, distortions, pose and calibration. SimCAM used tutorials and supplementary quizzes to guide users through the background material and interactive components.

SimCAM was evaluated through software testing of each unit, entire modules, as well as user acceptance testing. A content validity questionnaire was used to capture users' opinion on the simulator, tutorial and content. The results demonstrated that the need and value of such a simulator for camera calibration, with both experts and non-experts being polled with the questionnaire. Experts and non-experts were targeted for the content validity questionnaire.

Overall the aim of the content validity study was to determine the efficacy of SimCAM as a training environment for camera calibration. Participants were split into groups, experts and non-experts, based on their demographics. Experts were selected from the population if they had completed camera calibration more than 5 times. Experts were then verified by comparing, to non-experts their training in linear algebra and computer vision/graphics. Experts were also expected to have a higher score for the milestone quizzes. Unexpectedly, neither the non-experts nor the experts performed well (Experts 56.8% and Non-Experts 59.7%) in the milestone studies. The reason for low scores remain unexplained by either how long they spent on the tutorial or how many hours they were awake. Further feedback will be needed to clarify the reason for these results, however we are limited in our ability to gather more data due to time constraints.

Specific content validity results from both the experts (75%) and non-experts (80%) showed the content was valid and appropriate for camera calibration training. Design decision consideration questions revealed that SimCAM's individual components were valuable for training and appropriate for beginners. Experts also believed that an OpenCV software programming module is required in SimCAM, and that more simulation components are required.

Additional comments also indicated that although SimCAM was a good start, there were some more simulation components that the users' expected to see. Additional inquiries would help to clarify what additional simulation components are required. Overall users believed that the content was valid and appropriate for a pilot study.

Based on the feedback captured from the content validity, additional improvements can be made to the current implementation of SimCAM. Following up on the participants' comments may help to launch another project that will make SimCAM more useable, relevant and pertinent for students of camera calibration. Besides the content validity study, a knowledge transfer study (KTS) was considered and would be valuable to perform. The KTS would help to determine how well students perform calibration on real live cameras after being trained with SimCAM. A preliminary implementation has already been implemented with the existing code base of SimCAM, which just needs to be utilized. Finally SimCAM would benefit from additional implementation of modules such as uncertainty propagation, OpenCV programming, additional camera types, etc.

The author's contributions to this thesis are:

1. Creation a simulation framework for simulating ideal, non-ideal cameras and camera calibration
2. Creation of interactive widgets for the purpose of exploring various camera parameters
3. Writing a custom software platform for making OpenCV features compatible with WebGL 3D components
4. Conducting a content validity study to validate the created system

Bibliography

[1] M. Esteghamatian, S. E. Pautler, C. A. McKenzie and T. M. Peters, "A 2D to 3D ultrasound image registration algorithm for robotically assisted laparoscopic radical prostatectomy," *SPIE Proceddings,* vol. 7962, pp. 79621Z-79621Z-8, 2011.

[2] R. S. J. Estépar, C.-F. Westin and K. Vosburgh, "Towards real time 2D to 3D registration for ultrasound-guided endoscopic and laparoscopic procedures," *International Journal of Computer Assisted Radiology and Surgery ,* vol. 4, no. 6, pp. 549-560, 2009.

[3] D. Stoyanov, "A practical approach towards accurate dense 3D depth recovery for robotic laparoscopic surgery," *Computer Aided Surgery,* vol. 10, no. 4, pp. 199-208, 2005.

[4] P. Bao, J. R. Warmath, B. Poulose, J. Robert L. Galloway and A. J. Herline, "Tracked ultrasound for laparoscopic surgery," *SPIE Proceedings: Medical Imaging,* vol. 5367, pp. 237-246, 2004.

[5] M. Gschwandtner, M. Liedlgruber, A. Uhl and A. V´ecsei, "Experimental study on the impact of endoscope distortion correction on computer-assisted celiac disease diagnosis," in *10th IEEE International Conference on Information Technology and Applications in Biomedicine*, Corfu, Greece, pp. 1-6, 2010.

[6] S. Yamaguchia, A. Nishikawa, J. Shimada, K. Itoh and F. Miyazaki, "Real-time image overlay system for endoscopic surgery using direct calibration of endoscopic camera," *CARS 2005: Computer Assisted Radiology and Surgery,* vol. 1281, no. 19, pp. 756-761, 2005.

[7] S. P. Xiaoli Zhang, "Application of visual tracking for robot-assisted laparoscopic surgery," *Journal of Field Robotics,* vol. 19, no. 7, pp. 315-328, 2002.

[8] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," *Proceedings of the 7th International Conference on Computer Vision,* pp. 666-673, 1999.

[9] D. Samper, J. Santolaria, J. Pastor and J. Aguilar, "Teaching Camera Calibration by a Constructivist Methodology," *Education, IEEE Transactions,* vol. 53, no. 4, pp. 646,652, 2010.

[10] A. D. a. C. L. A. Heidi S. Chumley-Jones, "Web-based Learning: Sound Education Method or Hype? A Review of the Evaluation Literature," *Academic Medicine,* vol. 77, no. 10, pp. S86-S93, 2002.

[11] J. S. a. H. M. G. Wu, "Imaging Modalities," in *Bone Tumours*, New York, Springer New York, 2012, pp. 51-86.

[12] K. H. Wong, T. Peters and K. Cleary, "Imaging Modalities," in *Image-Guided Interventions*, Springer US, 2008, pp. 241 - 273.

[13] D. Merrill, "The Next-Generation Digital Camera," *Optics and Photonics News,* vol. 14, no. 1, pp. 26-33, 2003.

[14] C. L. Cheung, Fusion of Stereoscopic Video and Ultrasound for Laparoscopic Partial, London, Ontario, Canada: Published masters' dissertation, University of Western Ontario, 2010.

[15] Y. Morvan, Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video, Eindhoven: Published doctoral dissertation Eindhoven University of Technology, The Netherlands, 2009.

[16] S. J. Simon J. D. Prince, "The pinhole camera," in *Computer Vision: Models, Learning, and Inference*, Cambridge, Cambridge University Press, 2012, pp. 297-320.

[17] C. Wöhler, 3D Computer Vision: Efficient Methods and Applications, London: Springer, 2013.

[18] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE: Journal of robotics and automation,* Vols. RA-3, no. 4, pp. 323-344, 1987.

[19] "Open Source Computer Vision," 2013. [Online]. Available: http://opencv.org/. [Accessed 12 January 2013].

[20] W. T. J. Eugene M. Strand, "Prototyping and small scale software projects," *Proceedings of the workshop on Rapid prototyping,* vol. 7, no. 5, pp. 169-170 , 1982 .

[21] A. B. Zachary Dwight, "Laboratory Driven, Lean-to-Adaptive Prototyping in Parallel for Web Software Project Identification and Application Development in Health Science Research," *Journal of software engineering and applications,* vol. 5, no. 2, pp. 62 -68, 2012.

[22] M. O. Grant Allen, The definitive guide to SQLite, New York: Apress, 2010.

[23] E. F. Codd, "A relational model of data for large shared data banks," *Communications of the ACM,* vol. 26, no. 1, pp. 64-69, 1983 .

[24] B. D. F. L. W. Tom Christiansen, Programming Perl, Sebastopol, CA: O'Reilly, 2012.

[25] B. W. Kernighan and D. M. Ritchie, C Programming Language, Prentice-Hall software series, 1988.

[26] G. Bradski, Learning OpenCV: Computer Vision with the OpenCV Library, Cambridge: O'Reilly Media, 2008.

[27] Z. Nicholas, Maintainable Javascript, Sebastopol, CA: O'Reilly, 2012.

[28] B. Danchilla, Beginning WebGL for HTML5, Berkeley, CA: Apress: Imprint, 2012.

[29] "WebSockets," Mozilla Developer Network, [Online]. Available: https://developer.mozilla.org/en/docs/WebSockets. [Accessed 01 June 2013].

[30] "SQLite," [Online]. Available: https://www.sqlite.org/. [Accessed August 18 2013].

[31] "Perl's Database Interface," The Perl Foundation, [Online]. Available: http://dbi.perl.org/. [Accessed 20 October 2012].

[32] P. Rabbitson, The Perl Foundation, [Online]. Available: http://search.cpan.org/~ribasushi/DBIx-Class-0.08250/. [Accessed 20 November 2012].

[33] "Comprehensive Perl Archive Network," The Perl Foundation, [Online]. Available: http://cpan.org/. [Accessed 20 September 2012].

[34] S. Riedel, "Mojolicious: Modern Perl Framework," [Online]. Available: http://mojolicio.us/. [Accessed 23 Janurary 2013].

[35] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," *Dissertation completed for Doctor of Philosophy,* p. chapter 5, 2000.

[36] "Introducing JSON," ECMA International, [Online]. Available: http://json.org/. [Accessed 10 October 2012].

[37] E. International, "ECMA-262: ECMAScript Language Specifcation," Ecma International, Geneva, 2011.

[38] A. S. Catherine Leung, "Enabling WebGL," *Proceedings of the 19th international conference on World wide web,* no. 19, pp. 1369-1370, 2010.

[39] D. Brian, "Three.js Framework," in *Beginning WebGL for HTML5*, Berkeley, CA, Apress : Imprint, 2012, pp. 173 - 203.

[40] mrdoob, "Three.js," [Online]. Available: http://threejs.org/. [Accessed 3 October 2012].

[41] W. W. W. Consortium, "HTML5 : A vocabulary and associated APIs for HTML and XHTML," World Wide Web Consortium, [Online]. Available: http://www.w3.org/TR/html5/. [Accessed 20 Feburary 2013].

[42] "Backbone.js," Documentcloud, [Online]. Available: backbonejs.org/. [Accessed 10 Feburary 2013].

[43] J.-P. Boodhoo, "Model View Presenter," MDSN Magazine, [Online]. Available: http://msdn.microsoft.com/en-us/magazine/cc188690.aspx. [Accessed 10 August 2013].

[44] "Bootstrap," Twitter, [Online]. Available: http://getbootstrap.com/2.3.2/. [Accessed 13 September 2012].

[45] C. Snyder, Paper prototyping : the fast and easy way to design and refine user interfaces, San Francisco, California: Morgan Kaufmann, Elsevier Science, 2003.

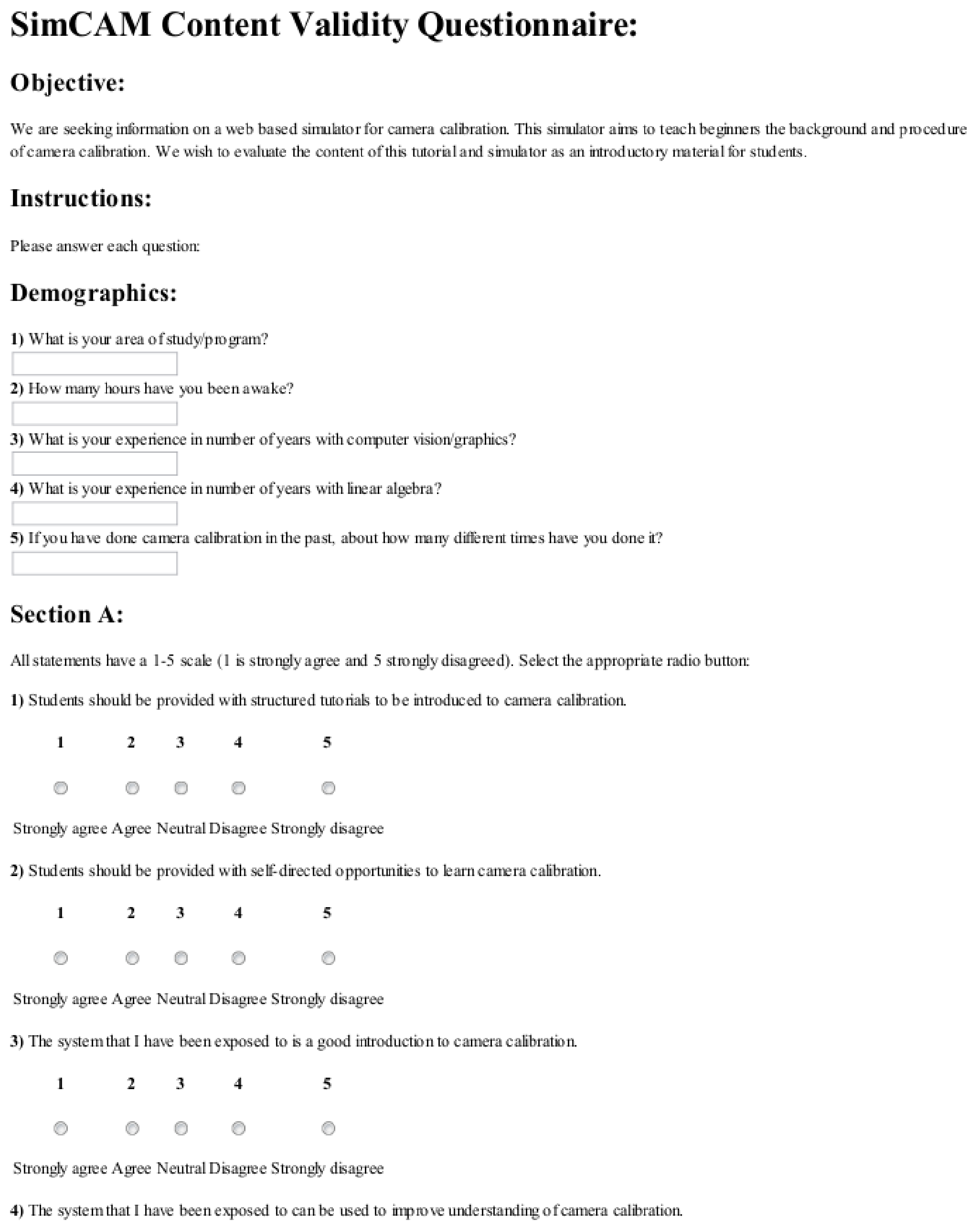
[46] A. Anastasi and S. Urbina, Psychological Testing, Prentice Hall, 1997.

[47] R. Likert, "A technique for the measurements of attitudes," in *Archives of Psychology*, 1932, pp. 1-55.

[48] A. K. Ho, "INTERACTIVE COMPUTER MODEL OF THE EARDRUM FOR MYRINGOTOMY SIMULATION," *Published masters' dissertation,* pp. 75-78, 2010.

Appendices

Appendix A: Content Validity Questionnaire



SimCAM Content Validity Questionnaire:

Objective:

We are seeking information on a web based simulator for camera calibration. This simulator aims to teach beginners the background and procedure of camera calibration. We wish to evaluate the content of this tutorial and simulator as an introductory material for students.

Instructions:

Please answer each question:

Demographics:

1) What is your area of study/program?

2) How many hours have you been awake?

3) What is your experience in number of years with computer vision/graphics?

4) What is your experience in number of years with linear algebra?

5) If you have done camera calibration in the past, about how many different times have you done it?

Section A:

All statements have a 1-5 scale (1 is strongly agree and 5 strongly disagreed). Select the appropriate radio button:

1) Students should be provided with structured tutorials to be introduced to camera calibration.

1	**2**	**3**	**4**	**5**
○	○	○	○	○
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

2) Students should be provided with self-directed opportunities to learn camera calibration.

1	**2**	**3**	**4**	**5**
○	○	○	○	○
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

3) The system that I have been exposed to is a good introduction to camera calibration.

1	**2**	**3**	**4**	**5**
○	○	○	○	○
Strongly agree	Agree	Neutral	Disagree	Strongly disagree

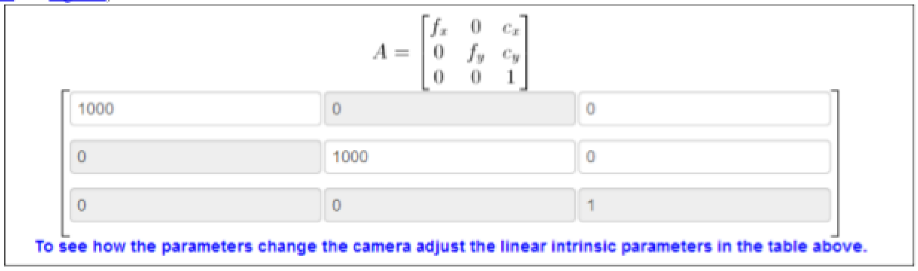
4) The system that I have been exposed to can be used to improve understanding of camera calibration.

1 **2** **3** **4** **5**

○ ○ ○ ○ ○

Strongly agree Agree Neutral Disagree Strongly disagree

5) The interactive simulation of the pin-hole camera model, camera matrix and pose matrix helps to improve understanding of camera calibration (Figure 1 and Figure 2).

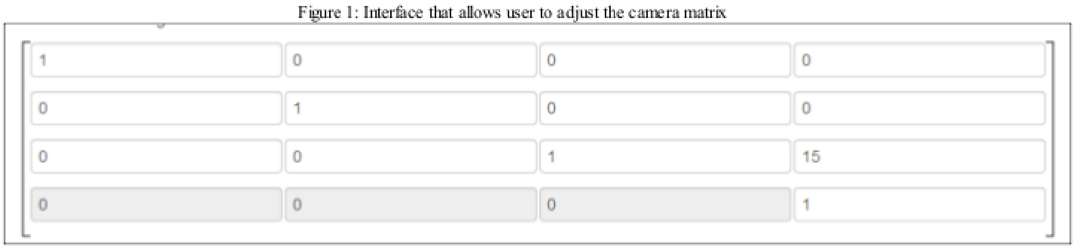


$$A = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

1000	0	0
0	1000	0
0	0	1

To see how the parameters change the camera adjust the linear intrinsic parameters in the table above.

Figure 1: Interface that allows user to adjust the camera matrix



1	0	0	0
0	1	0	0
0	0	1	15
0	0	0	1

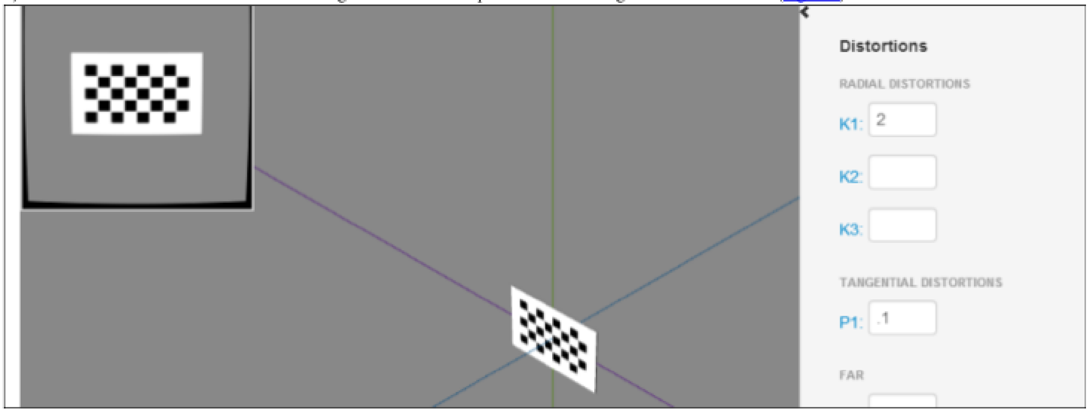
Figure 2:Interface that allows the user to adjust the pose matrix

1 **2** **3** **4** **5**

○ ○ ○ ○ ○

Strongly agree Agree Neutral Disagree Strongly disagree

6) The interactive simulation of the radial and tangential distortions improves understanding of camera distortions (Figure 3).



Distortions

RADIAL DISTORTIONS

K1: 2

K2:

K3:

TANGENTIAL DISTORTIONS

P1: .1

FAR

Figure 3: Interface that allows user to adjust distortion values

1 **2** **3** **4** **5**

○ ○ ○ ○ ○

Strongly agree Agree Neutral Disagree Strongly disagree

7) The interactive simulation of the calibration task improves understanding of camera calibration (Figure 4).

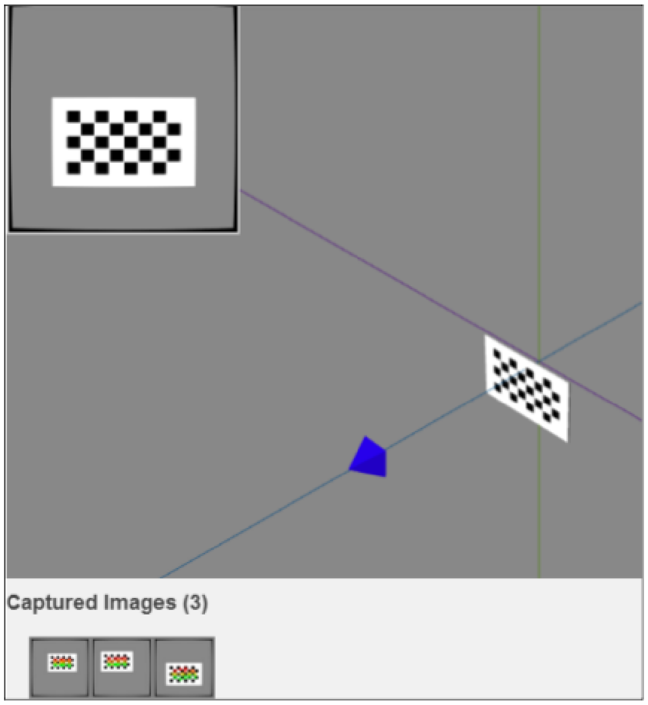


Figure 4: Interface that allows user to calibration distorted cameras

1	2	3	4	5
○	○	○	○	○

Strongly agree Agree Neutral Disagree Strongly disagree

8) The line graphs of the camera calibration parameters v.s calibration attempts are valuable to understand the effect of calibration gird placement on the estimated parameters (and how the eventually converge) (Figure 5).

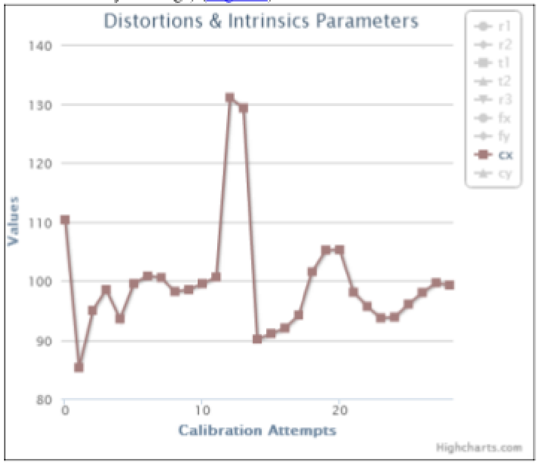


Figure 5: Series of graphs that plot values of parameters v.s calibration attempts

1	2	3	4	5
○	○	○	○	○

Strongly agree Agree Neutral Disagree Strongly disagree

9) The real-time feedback provided by showing the difference image which is computed by subtracting the corrected from the undistorted image, improves understanding of camera calibration (Figure 6).

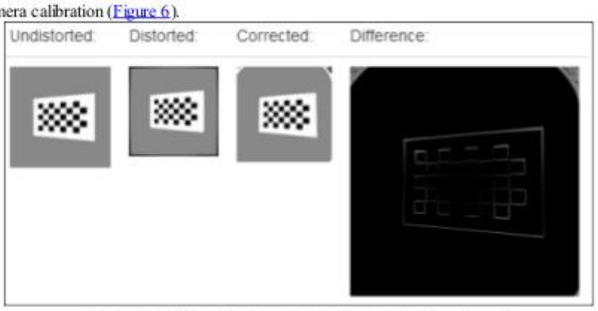


Figure 6: Difference image between undistorted and corrected

1	**2**	**3**	**4**	**5**
○	○	○	○	○

Strongly agree Agree Neutral Disagree Strongly disagree

10) Allowing calibration of the user's webcam is valuable as a practical exercise of camera calibration and improves understanding (Figure 7).

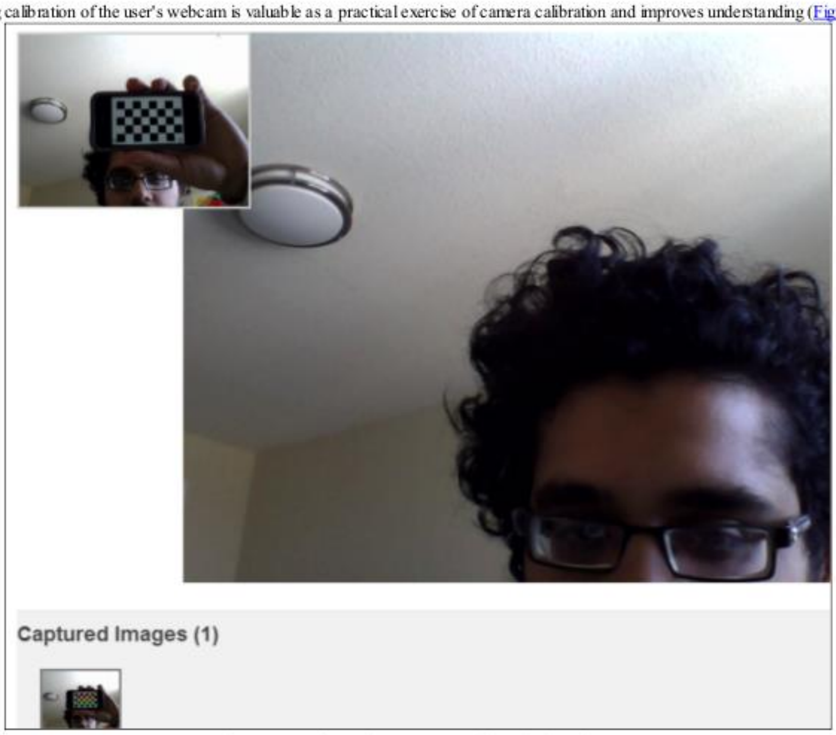


Figure 7: Interface where user can calibrate their webcam

1	**2**	**3**	**4**	**5**
○	○	○	○	○

Strongly agree Agree Neutral Disagree Strongly disagree

11) I would recommend use of the system for training camera calibration to beginners.

1	2	3	4	5
○	○	○	○	○

Strongly agree Agree Neutral Disagree Strongly disagree

12) I would continue use of the system for training if it were readily available.

1	2	3	4	5
○	○	○	○	○

Strongly agree Agree Neutral Disagree Strongly disagree

Section B:

Please select your response as yes or no:

1) Is simulating individual aspects (e.g. rotation matrix, distortion parameters, etc) of the camera calibration task valuable for training?

○ ○

Yes No

2) Is simulation of the overall calibration task in an ideal environment important for camera calibration training?

○ ○

Yes No

3) Should software programming using the OpenCV library (used for calibrating images) be included in the tutorial for camera calibration training?

○ ○

Yes No

4) Is the amount of simulation you were exposed to in the environment sufficient for an introduction to camera calibration?

○ ○

Yes No

Section C:

Please select your response.

Which of the following statement best describes the simulator's level of difficulty in comparison to a real camera calibration procedure (if you have performed it in the past, if not leave this question unanswered)?

Much less	Somewhat less	Equal	More	Much More
○	○	○	○	○

Section D:

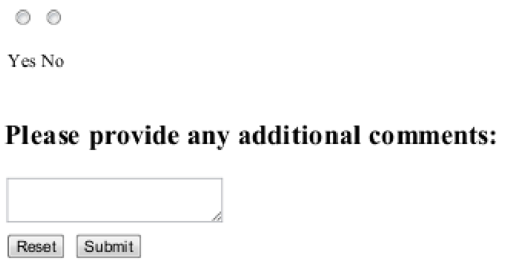
This section just determines if there were no technical difficulties during the study, please select yes or no:

1) Did the 3D interactive components load and were usable?

○ ○

Yes No

2) Did the webcam interactive component load and was usable?


○ ○

Yes No

Please provide any additional comments:

Appendix B: List of used OpenCV functions

cvFindChessboardCorners – Used to find the checker board board corners

cvFindCornerSubPix – Detects corners with sub pixel accuracy

cvDrawChessboardCorners – Draws circles and lines around detected corners

cvCalibrateCamera2 – Calibrates images from a camera using intrinsic parameters

cvInitUndistortMap – Applies the correction to undistort an image

Curriculum Vitae

Name:	Kartik Thakore
Post-secondary Education and Degrees:	Western University London, Ontario, Canada 2005-2011 BESc with professional internship (Software Engineering)
Related Work Experience	Graduate Research Assistant Western University, London, Ontario 2012 – 2013 Quality Assurance Test Analyst Canadian Imperial Bank of Canada, Toronto, Ontario 2009-2010

Presentations:

Poster

1. **Kartik Thakore**, Hanif Ladak and Terry Peters, Development and application of a web-based simulation and training environment for endoscopy camera calibration, London Health Research Day, March 14th, 2013

Conference Talk Presentations

1. **Kartik Thakore,** Perl, Medical Research and Maple Syrups, Yet Another Perl Conference North America, University of Wisconsin, Madison, June 3rd, 2012
2. **Kartik Thakore,** Game Development in Perl, Yet Another Perl Conference North America, Asheville, North Carolina, June 4th, 2011