Requirements Document: Implement a 3-D Scanner Using Triangulation

Grady Barrett–BSCS & BSCE
Jeffrey Bishop–BSCS
Tyler Gunter–BSCS

Faculty Mentor: Dr. David Wolff

CSCE 499, Fall 2012

Contents

List of Figures

1 Introduction

The three of us have worked together in the past, and coming into the semester, we had briefly corresponded on a project topic. However, after Dr. Wolff presented us with the idea of implementing a 3D scanner, we were hooked.

Our 3D scanner will use mathematical triangulation, calculating intersections between a laser line plane and the camera's "rays". The math behind it uses primarily linear algebra and geometry techniques, such as matrix operations and vector-plane calculations [1]. In order for these calculations to accurately display a 3D scan, we must account for intrinsic and extrinsic parameters of the camera [2, 3]. There are multiple techniques to perform this, including the Matlab toolbox provided by Caltech University [3], as well as OpenCV [2], which is what we will be using.

3D scanners are not an original idea; they have been studied and created by others. However, we would like to have the experience of building one ourselves, learning the math and concepts behind it and hopefully producing good results that display accurate depictions of real-world scanned objects.

Three-dimensional scanners are not only a cool concept but an important one with diverse applications, ranging from the medical industry to architectural preservation [4, 5].

2 Project Description

The challenge we face is how to create a digital 3-D representation of a real-world object. To meet this challenge, we must do the following:

1. Calibrate a webcam to obtain the intrinsic and extrinsic parameters to be used to accurately transform a real-world scene to an image.
2. Build a scanning environment that includes a stage, a mounted webcam, and a laser mounted on platform on a stepper motor.
3. Develop code that will interact with a hardware controller to rotate the stepper motor.
4. Develop an application that will record images, using the webcam, of the laser shone on an object and mathematically determine the coordinate system shift of points along the laser line to develop a point cloud.
5. Create a polygonal mesh representation of the object using the point cloud.
6. If time allows, merge multiple point clouds to obtain a full 360 degree representation of the object.
7. If time allows, perform post-processing techniques on the point cloud/mesh to display a more accurate scan.

8. If time allows, obtain object color information to map colors to each point in the point cloud.

For our project to be considered a success, we should be able to produce a polygonal mesh representation of a real-world object. This includes calibration, scanning, and post-processing of an image to a 3-D rendering.

2.1 Functional Goals

We have multiple goals for what our project will accomplish. These are listed below.

- Construct a scanning environment consisting of a motor-mounted laser, camera, and stage.
- Successfully calibrate a camera to isolate the intrinsic parameters.
- Successfully calibrate a camera to obtian the extrinsic parameters.
- Successfully capture a set images containing a laser line distortion across an object.
- Generate a set of data points using optical triangulation and image processing.
- Transform the set of data points into a polygonal mesh that represents the scanned object which can be manipulated in a 3D modeling application (e.g. Blender, Maya).
- Merge multiple point clouds of the same object from various angles into one point cloud giving a full 360° representation of the object.
- Perform post-processing on polygonal meshes to increase accuracy and appearance.
- Apply a texture map defining the colors found on the scanned object.

2.2 Educational Goals

We also have various goals for what we will achieve on an educational level by implementing this project. These are listed below.

- Achieve successful implementation of a medium-scale software application with a hardware component.
- Further develop our software engineering skills, including coding and working in a team environment.
- Learn more about computer graphics, including associated mathematics.

To be considered successful in our learning objectives, we will not only have a working 3D scanner, but will also be able to explain the functionality and most of the math behind it, both in general and in a presentation setting.

3 Development Resources

For developing our application, we will need the following resources:

- C++ Documentation [6]
  - We will be developing our application in C++ and will reference the documentation as needed.
- OpenCV Training Book [2]
  - We are new to OpenCV. This book provides a helpful reference for various applications for visual-based development. It also describes how to do camera calibration.
- Siggraph 3D Scanner Course [1]
  - This source provides a wealth of information on how to build a 3D scanner. It has examples as well as detailed descriptions of the math involved and information on various implementations and post-processing.
- Brown University course website(s) [7, 8]
  - These websites will be used in conjunction with the Siggraph website as they contain material from the same person. They have additional math info and post-processing software to potentially use as a reference.
- Linear Algebra textbook [9]
  - This textbook provides a base for understanding the mathematical concepts involved in computer graphics and can also serve as a reference for matrix and vector mathematics in general.

4 Requirements

This section describes various aspects of the project that are necessary for our project to be successful, as well as what the user can expect from the application.

4.1 Performance Requirements

Our application will focus more on accuracy than speed. The scanning process will be consistent by regulating the scanning motion and speed using the stepper motor.

Results depend on the user's setup and environment, but assuming these are done correctly, each scan should have the same relative level of precision and should depict a fairly accurate representation of the object.

Hardware should be consistent and reliable in order for the scanning process to provide the expected results.

4.2 Design Constraints

Our project has various constraints, including, but not limited to, the following:

- We must have the project sufficiently implemented by early May 2013.
- We must be able to implement the scanner using no more than $100.
- Objects to be scanned must be able to fit within the confines of our stage.
- The camera is limited to what surfaces it can actually see on the object.

4.3 User Characteristics

Users of our scanner system should have a basic technical understanding to both set up the scanning environment (i.e. camera, laser, stage) as well as calibrate the camera and scan objects using our application. Users will likely be hobbyists or general enthusiasts of computer graphics. They should expect a well-documented yet simple procedure to guide them through the entire calibration and scanning process. With this in mind, we will have executables that the user will run to calibrate the camera using simple GUI's. These will save the intrinsic and extrinsic parameters in files to be loaded in the scanning process. There will also be another executable that the user will run to perform the actual scanning process using another simple GUI. We will provide instructions to take the user through the entire process.

4.4 Assumptions

We assume that users of our scanner have the following:

- A basic technical understanding.
- Sufficient memory to accommodate image capture and triangulation processing of image data simultaneously.
- Hardware items:
    - 4.5 V Laser Line generator (SN C005858) or similar that works with OpenCV
    - Logitech Pro 9000 webcam or similar
    - Tripods
    - Applied Motion Products HT23-396 Stepper motor
    - Dragon 12-+ Development Board
    - Computer with Windows installed
- A dark room to scan in.

4.5 Security

Security is not a primary concern. However, we want to avoid memory overflows and other errors that could be used to compromise a system when run over a network.

4.6 Reliability

Given a successful calibration and environment (e.g. lighting), our application should maintain a high level of reliability and provide successful data to the user. However, due to the type and amount of processing the application will be performing, there is a higher probability of software failure. When this happens, our application should handle it with grace, including a message to the user to restart the scanning process.

Our scan reliability is limited by what the camera can actually see. Therefore, any surface of the object that is obscured by a surface in front of it will not be able to be picked up in the scan. The user should not expect this missing data to be inferred by the application. The data may be included in scans from alternate angles and therefore may be partially included in a merge of point clouds.

4.7 Portability

At this time, we only plan to support the Windows platform. Our application will contain Windows-specific components required to run the software. A future release of the application could extend the software to another platform (e.g. Mac, Unix/Linux).

4.8 Maintainability

Maintaining our software will include updating versions of both C++ and OpenCV. To handle this requirement, we will write our software with high cohesion and low coupling, allowing for smoother changes to code. The user will only be expected to update Windows and camera drivers.

4.9 External Interface

4.9.1 Calibration Interface

Figure 1 in Appendix A shows the user interface for the intrinsic parameter camera calibration process of the application.

Similarly, figure 2 in Appendix A shows the user interface for the extrinsic parameter camera calibration process of the application.

Both will allow the user to interact with the software directly, and the camera will be used in calibration.

4.9.2 Scanning Interface

Figures 3 and 4 in Appendix A show user interfaces for the scanning process of the application. The camera, laser line, and motor will all be used in the scanning process.

4.9.3 Hardware

Figure 5 in Appendix A shows a sketch of the hardware setup we will be using.

The stepper motor will be connected to H-bridges on a Dragon 12 development board which will interface with C code to control the motion of the motor. This motion will drive the laser across the object being scanned. The development board will also contain voltage dividers to provide correct voltage to the stepper motor and laser. The webcam will be connected via USB to the computer running the application.

4.9.4 File Formats

Our project will be written using OpenCV and C++. The code for the development board will be written in C.

Other file formats used in the application include XML files to save and load the intrinsic and extrinsic parameters, as well as VRML files for point clouds.

4.10 Use Cases

We have 2 main uses cases: Calibration and Scanning.

4.10.1 Calibration: Intrinsic Parameters

Description: The user will need to calibrate the camera they will use in order to get the intrinsic parameters required for scanning. Once calibrated, the output .xml files can be loaded for Scanning.

Precondition: Program started with webcam connected.

Normal Postcondition: Calibration successful with intrinsic parameter .xml files output.

Abnormal Postcondition: Calibration unsuccessful and user must exit.

Table 1: Calibration: Intrinsic Parameters

User	Application
1. The user enters the number of horizontal and vertical squares on the checker board and then presses Start to begin the calibration process.	
	2. The application stores the number of horizontal and vertical squares entered by the user minus one as a variable, which defines the number of horizontal and vertical inner corners on the board.
	3. The application turns the camera on and displays a message for the user to position the board for the next picture. A video feed will be displayed in order to make positioning the board easier on the user.
4. The user positions the board for the next calibration picture and presses the Take Picture button.	
	5. The application captures the image.
	6. The application processes the image.
	7a. The image is processed successfully and the application increments the # Images Successfully Processed field.
	7b. The image is not processed successfully and the application displays the error processing image message.
8a. The image is processed successfully so the user repositions the board for the next picture and presses the Take Picture button.	
8b. The image did not process successfully so the user repositions the board close to where it was for the picture that did not process successfully. The user then presses the Take Picture button.	
	9. Steps 5-8 are repeated until the hard-coded specified number of pictures have been successfully processed.

Continued on next page

Table 1 – *Continued from previous page*

User	Application
	10. Once all pictures have been successfully taken and processed, the application performs the necessary remaining calculations, notifies the user that it was successful, asks the user for a file names and stores the intrinsic data in the .xml files specified by the user.
11. The user presses the Exit button.	
	12. The application exits.

4.10.2 Calibration: Extrinsic Parameters

Description: The user will need to determine the extrinsic parameters for the camera in order to transform the scanning environment from the world coordinate system to the camera coordinate system.

Precondition: Camera has been calibrated to determine the intrinsic parameters and the camera and stage are set up in the proper positions.

Normal Postcondition: Extrinsic parameters are successfully determined and saved to an .xml file.

Abnormal Postcondition: Extrinsic parameters were not successfully determined and user must exit.

Table 2: Calibration: Extrinsic Parameters

User	Application
1. The user enters the number of horizontal and vertical squares on the checker board.	
2. User clicks Browse on each of the Load XML and selects a directory to load the respective XML files containing the data obtained from the first calibration step.	
	3. The XML files that are loaded into the application are parsed.
	4a. If the XML is parsed correctly, the application continues.

Continued on next page

Table 2 – *Continued from previous page*

User	Application
	4b. If the XML is not parsed correctly, the application returns back to step 2 and displays an error message asking the user to correct mistakes.
	5. The application stores number of horizontal and vertical squares entered by the user minus one, which defines the number of horizontal and vertical inner corners on the board.
	6. The application turns the camera on and displays video feed and a message for the user to position the board on the ground plane for the next picture.
7. The user positions the board on the ground plane of the stage and presses the Take Picture button.	
	8. The application captures the image.
	9. The application processes the image.
	10a. The image is processed successfully and the application continues for the next image if there is one.
	10b. The image is not processed successfully and the application displays the error processing image message.
11a. The image is processed successfully so the user repositions the board for the next picture and presses the Take Picture button.	
11b. The image did not process successfully so the user repositions the board close to where it was for the picture that did not process successfully. The user then presses the Retake Picture button.	
	12. Steps 8-10 are repeated with the board on the back plane of the stage.

Continued on next page

Table 2 – *Continued from previous page*

User	Application
	13. Once both pictures have been successfully taken and processed, the application performs the necessary remaining calculations, the user determines where the output extrinsic parameter files are to be placed. The application stores the extrinsic parameters in the respectively named .xml files.
14. The user presses the Exit button.	
	15. The application exits.

4.10.3 Scanning

Description: The user will scan an object (of an acceptable size) using the laser and the calibrated camera.

Precondition: Camera has been calibrated and all hardware components are connected properly with the stage in place.

Normal Postcondition: Scanning succeeds and a point cloud representation of the object is shown.

Abnormal Postcondition: Scanning failed and error message is shown to the user.

Table 3: Scanning

User	Application
1. User clicks Browse on each of the Load XML and selects a directory to load the respective XML files containing the data obtained from calibration.	
2. User clicks Start to begin scanning the positioned object.	
	3. The application determines if the XML files are readable and if they exist. If they are, the application will parse the files, loading the correct calibration data into the application.

Continued on next page

Table 3 – *Continued from previous page*

User	Application
	3a. If the application is unable to read the XML data or cannot write to the output directory, the application will display an error message to the user, and allow them to correct and mistakes. 4. After XML is done parsing, the application will bring up another window that will display a picture of the object being scanned where the user will be able to click regions of the image to determine where the object is in the image plane.
5. The user clicks the regions within the displayed image and then presses the "Begin" button.	
	6. Application sends interrupt signal to hardware, which was previously in a "wait" state. 7. Motor position is set to initial position. The laser and camera are turned on, the motor begins to rotate, and the camera captures video. Scanning process begins. A progress bar will appear, showing the progress of the scan. Each frame will be stored via the hard disk drive. 8. All mathematical formulas will then be computed to locate the laser plane and the intersection with the object. This will require reading the frame from the disk, doing the aforementioned calculations and storing the time associated with the points (the frame number) in an Image object in the application. 9. The scanning process completes and displays a results window containing the file output location, the point cloud, and prompts the user for the output directory and the file name. The user also has the ability to exit the program now that the scan is complete.

5 Task Breakdown

1. Set up the scanning environment.
   (a) Assemble the 2 boards to act as our stage. This includes purchasing the materials, cutting the boards to size, coloring them as needed, and connecting them at a right angle.
   (b) Set up the tripods, one with the camera and the other with the stepper motor and laser. We will connect the camera to the computer.
   (c) Prototype the stepper motor development board control logic and hardware connections. We will connect the stepper motor to the computer.
      i. Connect the stepper motor to the Dragon 12 development board. This will include connecting to the H-bridge for control of current flow direction.
      ii. Develop and compile code to interface with the motor.
      iii. Attenuate a voltage source using two voltage dividers to provide power to the laser and the stepper motor.
2. Camera calibration. We will need to determine the parameters of the camera to use for scanning.
   (a) Calibrate our own camera using the setup location we have determined.
      i. Obtain a chessboard and take a series of photos using provided OpenCV code. The series of photos will be analyzed by the code to determine the intrinsic parameters. Then we will determine the extrinsic parameters related to our stage setup. We will use our calibration in order to provide essential information for us to proceed to the main portion of our project: 3D scanning.
   (b) Prototype the GUI and applications for calibration.
   (c) Develop GUI's for the calibration process. This will occur later in the project in order to provide a user-friendly interface for the calibration use case.
3. Scanning an object. This will include traversing the object to be scanned with the laser line and analyzing the data collected by the camera.
   (a) Write a simple program to analyze the laser component and plot the intensity across a row of pixels.
   (b) Prototype the GUI and application for scanning.
   (c) Write an application to collect image data about the laser line traversing the object.
   (d) Extend the application to analyze the collected images.
      i. Determine data points using triangulation.

        ii. Form a point cloud representation of the scanned object.

    (e) Develop a GUI for the scanning process to provide a user-friendly interface. This will occur after we have successfully scanned an object.

4. Post-processing. This will occur after obtaining a point cloud.

    (a) Connect the points using time data to form a polygonal mesh.

    (b) As time allows, merge multiple point clouds to obtain a full 360° representation of the object.

    (c) As time allows, perform post-processing techniques to provide a higher degree of accuracy of the 3D scan.

    (d) As time allows, add color data to the polygonal mesh.

6 Preliminary Timetable

Figure 6 in Appendix A shows a preliminary schedule for our project as a Gantt chart.

7 Budget

Our project will require the following items:

- Laser line generator (Provided by CSCE Department)
- Motor platform (Provided by CSCE Department)
- 2 Tripods ($20: Preferred CSCE Purchase)
- Logitech Pro 9000 Webcam (Already Owned)
- Stepper motor (Provided by CSCE Department)
- Various electronic items (Provided by CSCE Department)
- Dragon 12 Development board (Provided by CSCE Department)
- Fiber Boards ($20: Preferred CSCE Purchase)
- Books, etc. (library, online, purchased by Dr. Wolff)
- Visual Studio 2010 (Student License)
- Code Warrior (Free Special Edition)
- C++ (no cost)
- OpenCV (open source)

- Blender (open source)

8 Development Documentation

The website for our project can be found at `http://www.cs.plu.edu/~scanners`. All associated documents for the project can be found in links on this website

All revisions to documents, code, etc. can be found in our github repository, which can be found at `http://github.com/scanners/3DScanner`.

Each developer of the application has a blog as well. These are linked from our website; the URL for each is also provided below:

- Grady Barrett: `http://barretgm14.wordpress.com/`
- Jeff Bishop: `http://jeff3dscanner.blogspot.com/`
- Tyler Gunter: `http://tylergunter.wordpress.com/`

Glossary

extrinsic parameters The translation and rotation vectors relating the scanning stage's coordinate system ("world coordinates") to the camera's coordinate system ("camera coordinates").

H-bridge A circuit that provides functionality to reverse DC current direction through a load using switches.

intrinsic parameters A model of the camera's geometry and a distortion model of the camera's lens [2, p. 371].

stage Two perpendicular boards in which the object to be scanned is placed.

triangulation A mathematical process of determining an intersection point using geometry.

References

[1] D. Lanman and G. Taubin, "Build your own 3d scanner: Optical triangulation for beginners." http://mesh.brown.edu/byo3d/index.html, 2012.

This website provides a major resource for how to build a 3D scanner. It includes critical mathematics necessary for triangulation as well as helpful information regarding the scanner system itself.

[2] G. Bradski and A. Kaehler, *Learning OpenCV.* Sebastopol, CA: O'Reilly Media, Inc., 2008.

This book provides a learning tool for OpenCV, as well as examples. We are using OpenCV to program our application; the book provides direct assistance for calibrating a camera for use in OpenCV and we will extend the calibration parameters to use in the scanning process.

[3] J.-Y. Bouget, "Camera calibration toolbox for matlab." http://www.vision.caltech.edu/bouguetj/calib_doc/index.html, 2010.

This website contains the toolbox for camera calibration in Matlab. It also contains information and examples for using it.

[4] T. Tong, "Medical applications in 3d scanning." http://blog.3d3solutions.com/bid/78455/Medical-Applications-in-3D-Scanning, 2011.

This blog provides information from the 3D3 Solutions company, which makes professional-grade 3D scanning products. The blog provides examples of how 3D scanning is used in the medical industry.

[5] "3d digital corp applications: Architecture." http://www.3ddigitalcorp.com/applications/architecture, 2012.

This webpage provides architectural applications of the scanners from 3D Digital Corporation. Their website also lists other applications, including manufacturing and automotive.

[6] "C++ documentation." http://www.cplusplus.com, 2012.

This provides a helpful reference for C++ functionality.

[7] G. Taubin, "3d photography and image processing." http://mesh.brown.edu/3DPGP-2009/index.html, 2009.

This website provides additional 3D information and mathematics from Brown University. It also gives a resource for post-processing information.

[8] G. Taubin, "ENGN 2502 3d photography." http://mesh.brown.edu/3DP/index.html, 2012.

This website provides additional information about 3D processing and photography from

Brown University. It includes some of the professor's lecture slides, which can serve as a resource.

[9] T. Shifrin and M. R. Adams, *Linear Algebra: A Geometric Approach.* New York, NY: W.H. Freeman and Co., 2 ed., 2011.

This book serves as a math resource for learning about some of the mathematics needed for matrix calculations and vector.

A Appendix A

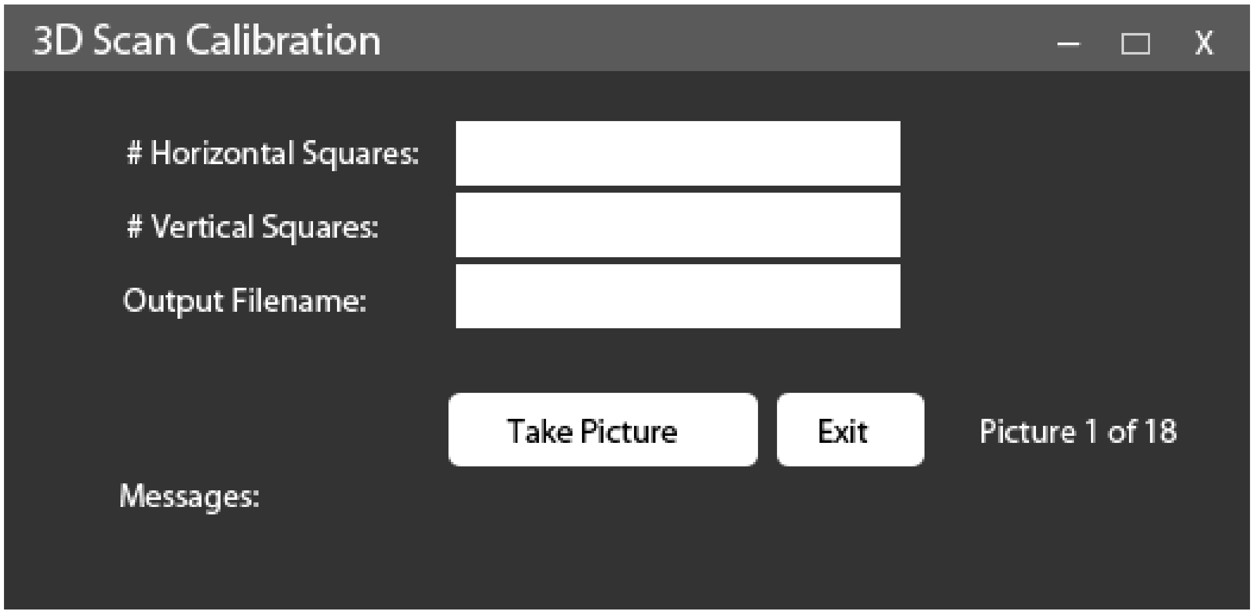

Figure 1: Main calibration screen for intrinsic parameters.

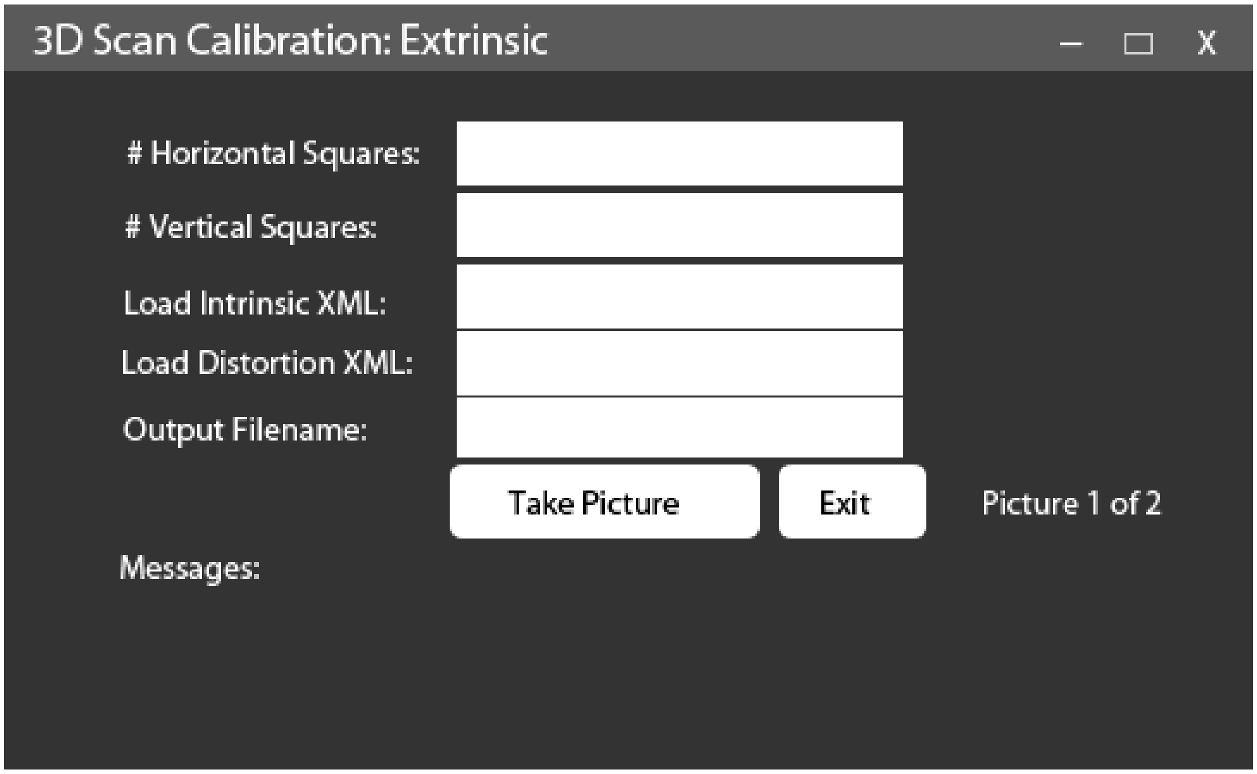

Figure 2: Main calibration screen for extrinsic parameters.

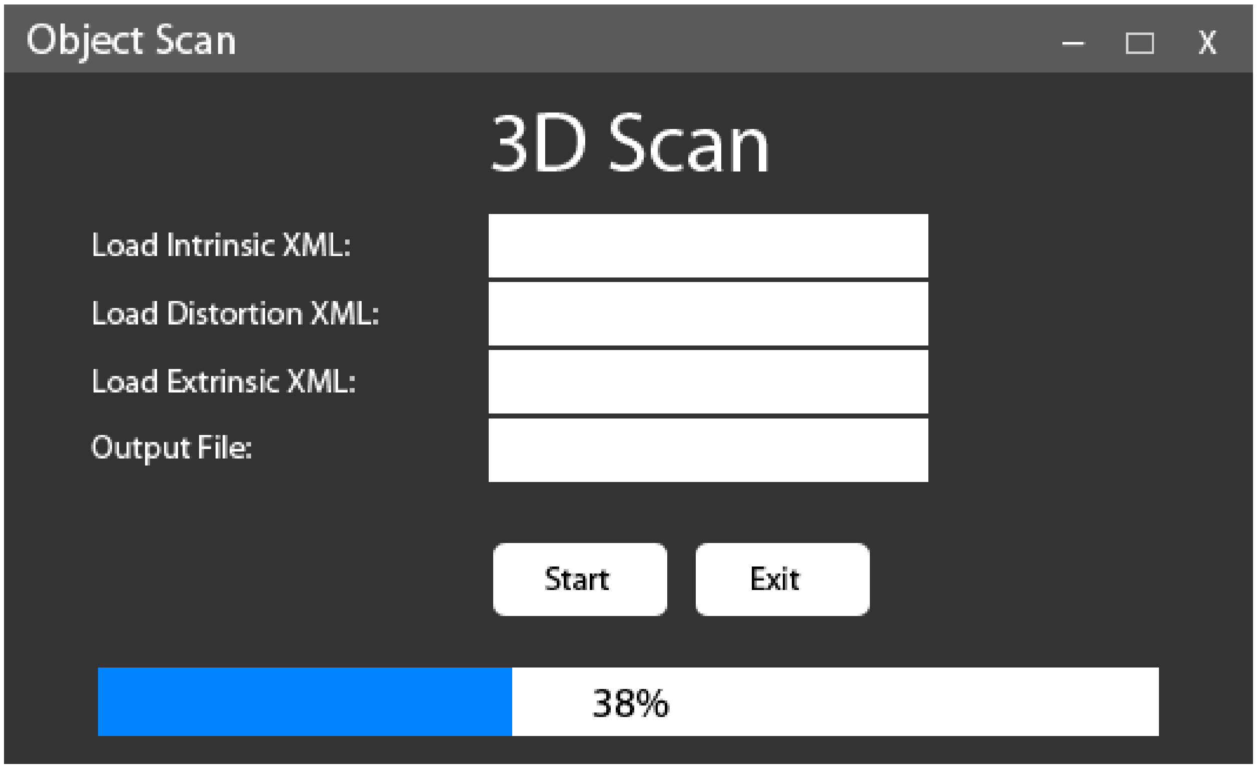

Figure 3: Main scanning screen

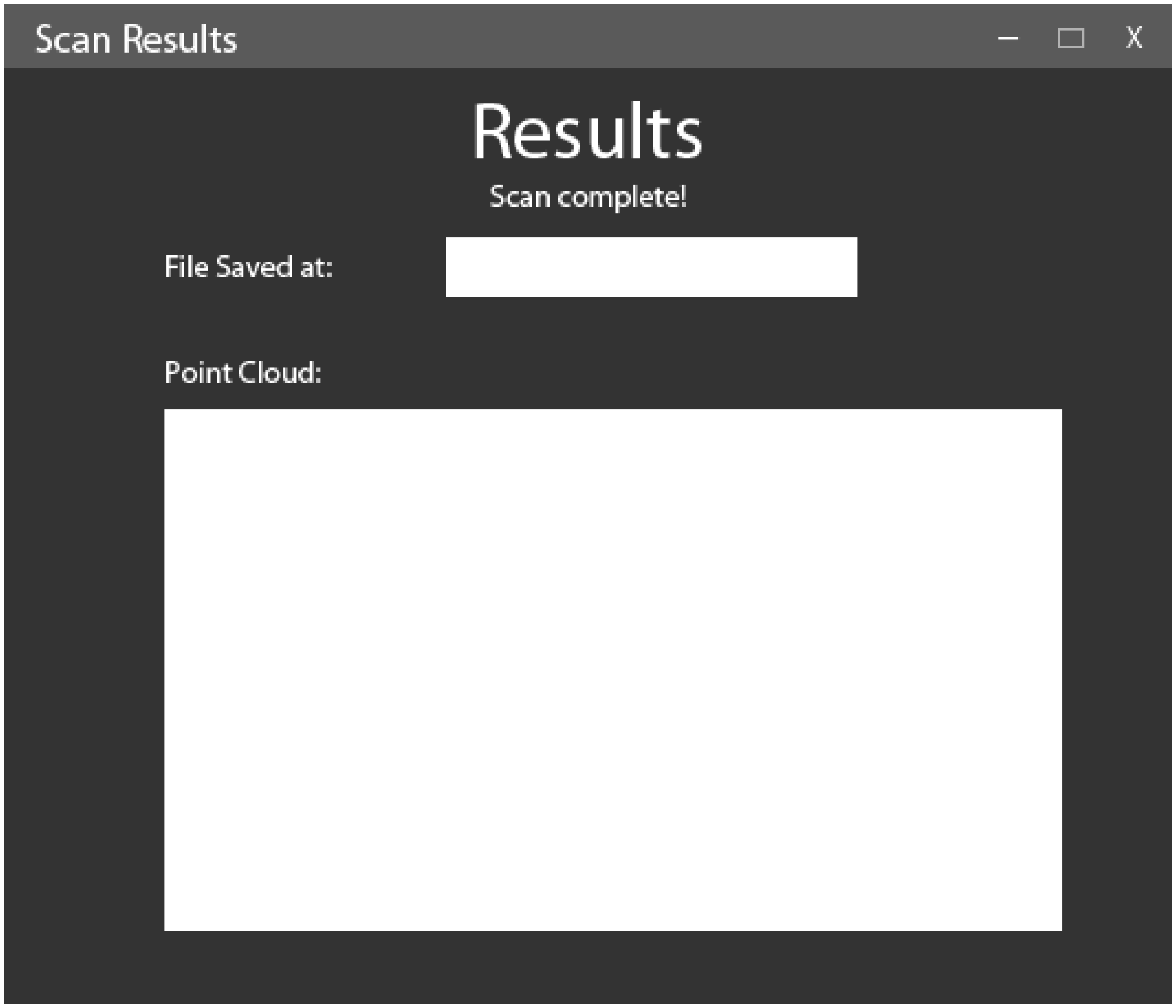

Figure 4: Screen showing the resulting point cloud.

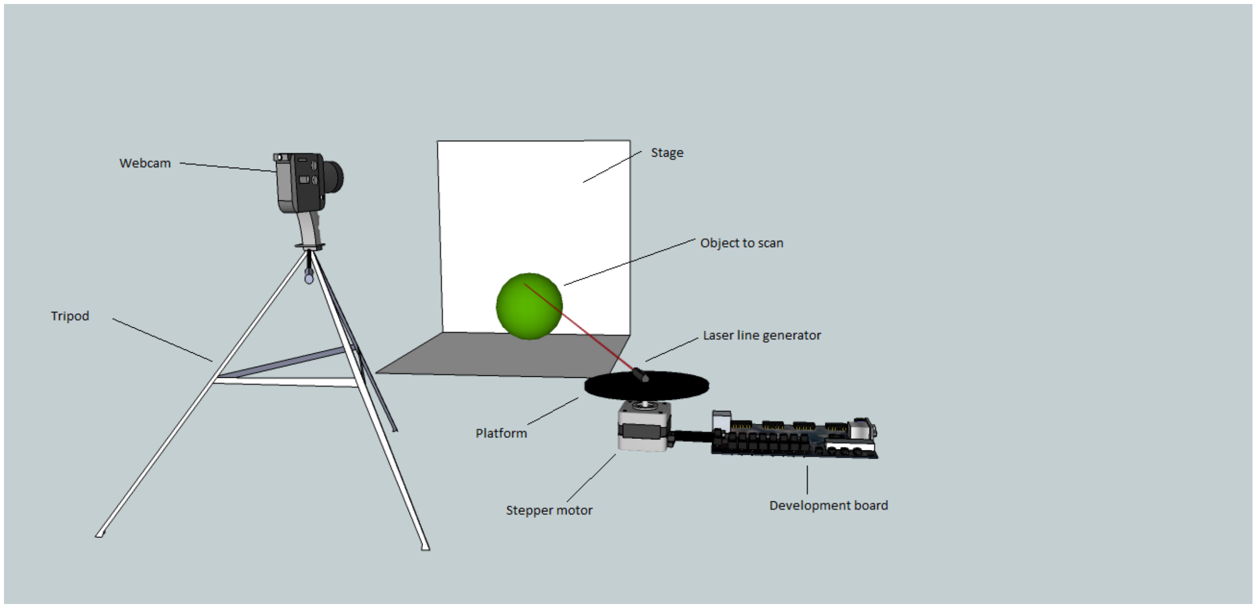

Figure 5: Sketch of what our setup will be.

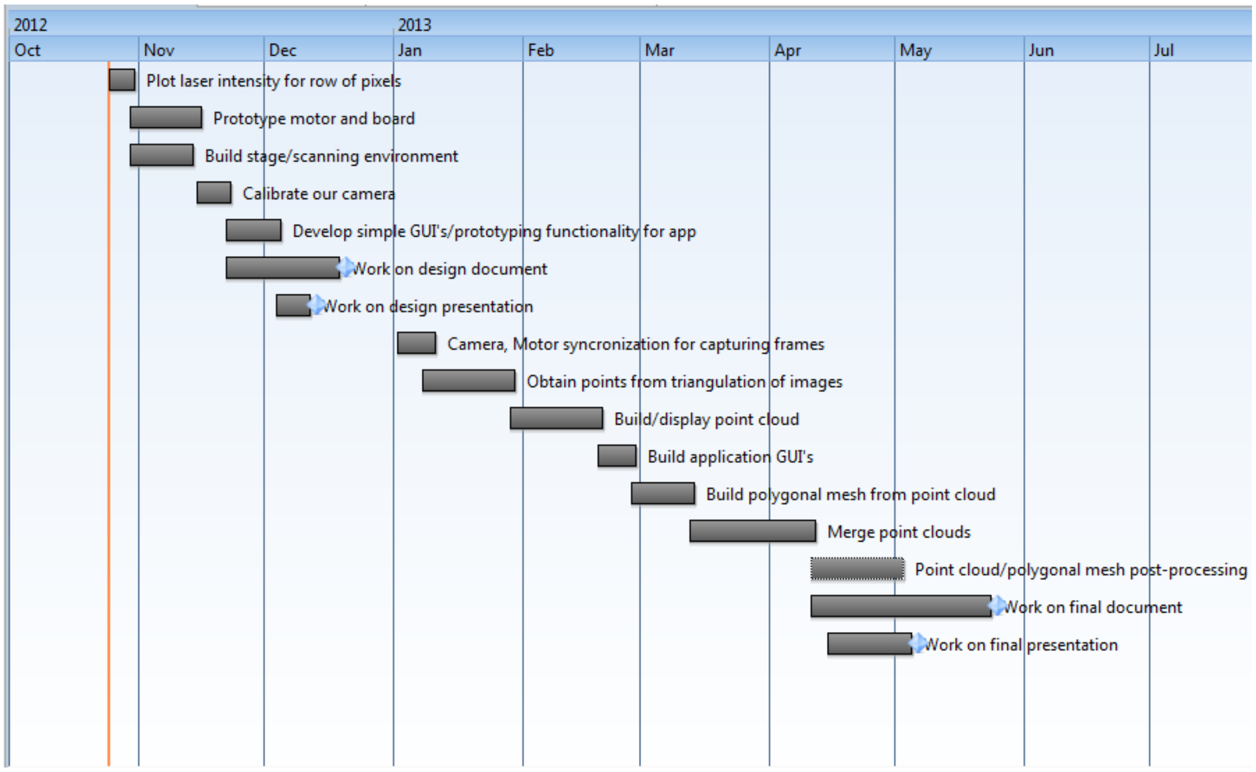

Figure 6: Gantt Chart for our project schedule.