Scanner for Automated 3D Modeling of Small Objects

Sam Benjamin, Isaias Velez, Sommer Hilliard and Cary McEwan

Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — With the ubiquity of 3D printers, The Scanner for Automated 3D Modeling of Small Objects aims to bring a quick, inexpensive, and user friendly experience for creating 3D printable object files. It uses a fully automated system that allows the user to simply click and scan to receive the resulting file via email, abstracting away the low-level implementation details of 3D reconstruction. The scanner is lightweight, portable, and powered by a Raspberry Pi to provide full scans within 10 minutes.**

***Index Terms* — 3D Scan, Meshing, Point Clouds, Poisson Reconstruction, Surface Reconstruction**

I. Introduction

Three-dimensional metrology and manufacturing have evolved significantly over the past decade, and the latest technology has adapted it for individual use in the form of 3D printers and scanners. Science labs and professional engineering firms are already using these to simplify production of specialized hardware in experiments and commercial products, and even hobbyists have begun to make use of them in projects of their own. Specialized hardware components that once had to be ordered and shipped over a period of days can now be printed and installed in a matter of hours at little cost or hassle. Creating a custom 3D-printable file of a real-world object, however, does pose a significant issue for some. Design software such as SolidWorks and AutoCAD require specialized knowledge and extensive practice to use effectively, and even with training it can be tricky to duplicate complex geometries in a reasonable amount of time. Our aim is to design a desktop scanner that is affordable, user-friendly and can produce scans of submillimeter quality that are useful in professional STEM-related environments.

There are many ways 3D scanning can be implemented. Ultrasonic emitters, visible-wavelength LEDs, infrared LEDs, laser diodes, grid projections, speckle field projections, and photogrammetry are all viable options. Our implementation uses an active line scan architecture, where a laser diode's beam is passed through diffraction grating to project a vertical line of illumination into a fixed plane. The line of illumination creates a profile of the scanned object's contour, and an image is subsequently captured from an angle at which the contour is pronounced. The image is passed into a series of processing algorithms that filter out extraneous information such as static background and unnecessary color channels. The beam profile's center is determined in terms of pixel array coordinates; once this process has been repeated for the entire vertical profile, the useful real-space coordinates are triangulated from the pixel array data. A point cloud of the coordinates is generated in the form of a text file, which is run through a meshing process and output in the form of a standard 3D-printable file.

The system is designed using low-cost hardware and open-source image processing software, coupled with our custom software suite. The end result is a further minimized price, relieving the user from purchasing expensive third-party software. Similar to a 2D desktop scanner, the final deliverable is a product that can be unboxed, plugged in, and immediately put to use.

II. System Components

A. Microcontroller

Serving as the core of the 3D Scanning System, the microcontroller is one of the most important components in the system. The microcontroller is responsible for controlling the stepper motor, camera and laser. In addition to this, it is also responsible for processing the captured images for the three dimensional point cloud data. With 800 images captured per scan, each featuring a 3920 x 2464 pixel resolution, it is imperative that the microcontroller has sufficient clock speed in order to process this data within the specified scan time, without compromising the sample rate and overall quality of the scan. As such, the Raspberry Pi 3 Model B was selected. Its 1.2 GHz clock rate and 1 GB SDRAM served crucial for processing point cloud data while also remaining within the time constraint. The 40 GPIO pins allow for significant input/output potential needed for controlling the laser and the motor. The onboard HDMI allows for direct connection to the LCD Display. Another one of the primary reasons it was chosen over the other microcontrollers in consideration was its wide adoption and extensive

developer community. The many references and resources available served vital in streamlining the software development process.

III. Electrical Detail

A. Hardware Block Diagram

The hardware block diagram depicts the flow of the power and the respective required voltages for each component. The block diagram can be seen in the figure below. Refer to this figure to reference the material described in the remaining subsections of the electrical design section.

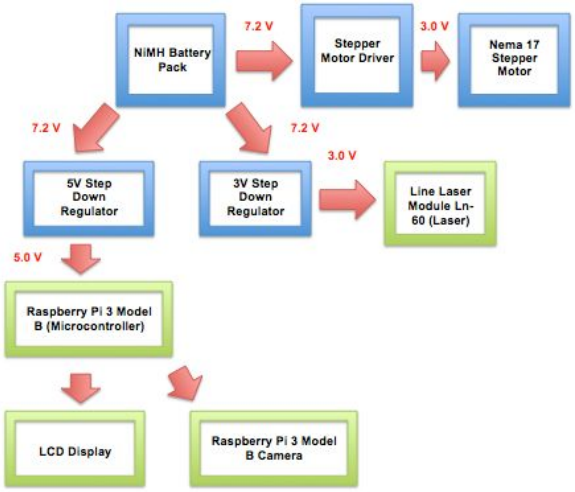

Fig. 1. Hardware Block Diagram

B. Battery Technologies

The most common type of rechargeable battery used in similar projects like this one is the nickel-metal hydride battery. This battery is popular because of its value. It is fairly cheap in relation to its capacity and weight. Furthermore, it hardly has any memory effect, which means that with each charge it will reach its full capacity. Also, this battery has high energy density compared to its counterparts. However, it's self discharge rate is quite unfavorable and it is sensitive to overcharging, but this can be avoided using a smart battery charger. The battery was chosen based on the total power consumption of the project. The major components that comprise this total power draw are the Raspberry Pi 3 Model B, the stepper motor driver, and the line laser. With all peripherals in use on the Raspberry Pi, it can draw up to 2.5A at 5V. The stepper motor can draw up to 2A between 6-30volts. Finally, the line laser will draw a measly 35mA at 3V. The component that requires the highest operating voltage is the stepper motor driver. It requires an input between 6 volts and 30 volts. The battery chosen for this project is the 7.2V 5000mAh NiMH battery pack. It is sufficient for the power management of this project. It has a higher nominal voltage than any component in the design and it can provide 5A of current for an hour. Based on the information provided above, the max current draw will be roughly 4.5A; therefore, the 5A current rating is more than enough. This battery can be fully charged at a rate of 1A or 2A, where it takes approximately 200 minutes to fully charge at a rate of 2A.

C. Stepper Motor & Stepper Motor Driver

The stepper motor driver is built using the A3967 IC chip. This stepper motor driver can drive up to 750mA per phase. The stepper motor used is the NEMA 17 stepper motor. The stepper motor is a bipolar motor meaning the stepper motor driver can drive up to 1.5A. The stepper motor is able to provide 400 steps per revolution, however, the stepper motor driver defaults to 8-step microstepping mode. This means that with this particular stepper motor driver in conjunction with the NEMA 17 stepper motor, it would be possible to get 3200 steps per revolution. The A3967 stepper motor driver has internal circuit protection that includes thermal shutdown with hysteresis, undervoltage lockout, and crossover current protection. The stepper motor driver schematic can be seen in figure 2 below.

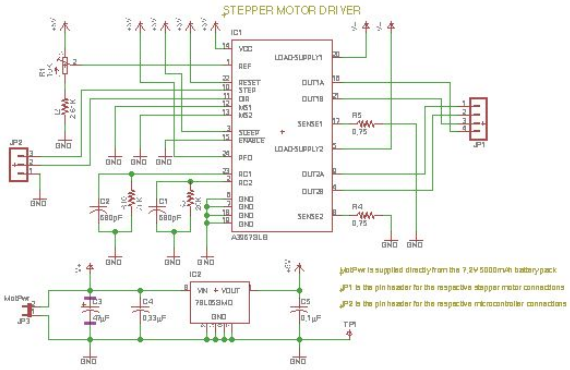

Fig. 2. Stepper Motor Driver Schematic

D. Voltage Regulation

This project consists of two voltage regulators that step down the 7.2V at the input to 3V and 5V respectively. Both of the voltage regulators will utilize the LM2576-ADJ switching regulator. This regulator has adjustable output voltages that range from 1.23V to 40V. Furthermore, the output current

is guaranteed to sufficiently reach 3A. Lastly, the input voltage must be at least 7V. This is the perfect regulator for supplying a regulated voltage to the Raspberry Pi and the line laser. The calculation to design this regulator are seen below.

$$V_{out} = V_{ref}(1 + {}^{R_2}/_{R_1}) \qquad (1)$$

$$R_2 = R_1({}^{V_{out}}/_{V_{ref}} - 1) \qquad (2)$$

Both the 5V switching regulator and the 3V switching regulator were built on a breadboard in the lab before designing the schematic for the printable circuit board. The input voltage was stepped up from 0V to 15V with 500Ω load on the output. The data confirmed that the input voltage needs to be at least 7V for proper regulator functionality. As discussed previously, the battery will provide an input of 7.2V, which is sufficient for proper operation for the switching regulator. Furthermore, the ripple voltages were measured at the output to compare against the datasheet for the LM2576. The maximum ripple voltage in the datasheet reads 50mV. During testing the output voltage ripple was between 20mV and 25mV for each regulator circuit. The PCB design optimized this output further to reach a very clean output with only 10 mV of ripple voltage.

D. Mosfets

The laser needed to be able to switch on and off in conjunction with the rotation of the stepper motor. This was accomplished with mosfets, specifically the IRFZ44N N-Channel Mosfet. This device has very low on-resistance and fast switching speed. The efficiency and reliability of this device is important for the project and the sensitivity of the raspberry pi gpio pins. Two low side n-channel mosfets were used in conjunction to design a switching circuit from the 3V regulator to the line laser. Originally only one mosfet was to be used, but the second mosfet was important to ramp the input gate voltage up for the other mosfet in order for proper switching. The schematic is seen in the figure 3 below.

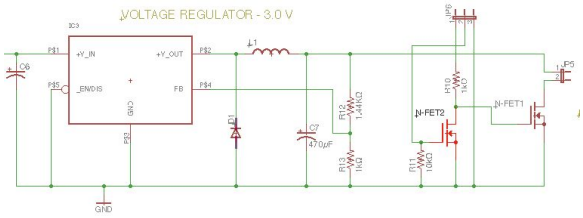

Fig. 3. 3V Switching Regulator and Switch Schematic

E. PCB Design

The schematic and pcb were designed using the PCB design & schematic software called Eagle. All of the subsystems for the overall circuit were tested in the lab before fabrication on the board. This was to ensure that the final board design would function smoothly and only minor troubleshooting would be necessary. The overall schematic design can be seen in the figure below.

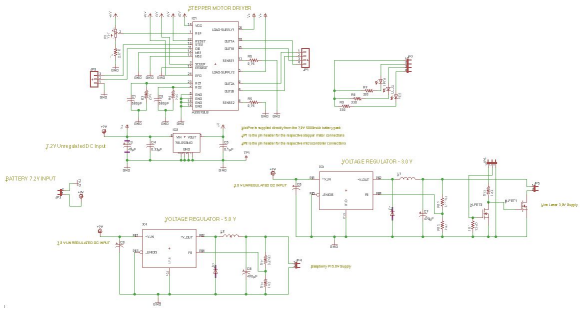

Fig. 4. Overall Design Schematic

The board file was fabricated with OSH Park. It is a two layer board with all components on the top layer and a ground plane on the bottom layer. Many of the parts were in the Eagle libraries, but a handful of the devices had to be designed from scratch using the respective datasheets. The figure below is the revision of the first printable circuit board. The first one had some mistakes with the footprint of several devices and several components needed to be replaced. The final design added the switching circuit for the laser and several LEDs to indicate when the laser is on, when the scan is in process, and when the scan is complete. The pcb design can be seen in figure 5.

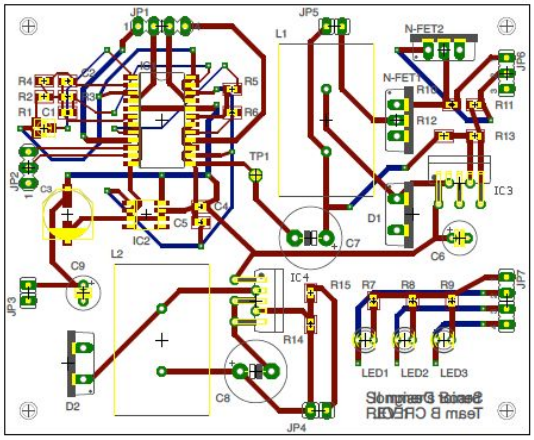

Fig. 5. Final PCB Design

IV. Calibration Procedures

In order to assign real units to a point cloud, it is necessary to define important spatial characteristics of the scanning system using several calibration processes. Three distinct calibration procedures are used for our scanner, each involving a planar chessboard-patterned test object.

A. Intrinsic Calibration

Intrinsic calibration finds a set of characteristics unique to the camera in terms of pixel units. The characteristics useful to us are the camera's center and its focal length. The OpenCV code we use requires an input of at least 12 images of a test chessboard at various incident angles, and it returns the camera center and focus as elements of the camera matrix,

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

where f_x and f_y are the axis-specific foci and c_x and c_y are the camera's optical center. This matrix will later be utilized to calculate the projection vector needed to triangulate each point in terms of real units.

The camera matrix is independent of external conditions such as camera and object positioning, and it only needs to be re-calculated when the camera's focal length is adjusted.

B. Extrinsic Calibration (Laser)

To triangulate the position of any given pixel address obtained through our scans, the laser's plane must also be mathematically defined. This is achieved by performing a scan on a planar object, extracting a normal vector **n** from the resulting point cloud, and then extracting the minimum distance **d** from the camera center to the laser plane.

C. Extrinsic Calibration (Turntable)

Lastly, the chessboard is again placed on the turntable and imaged every 5° of motor rotation for a full 180° of displacement from its starting position. The same OpenCV pattern recognition code used previously for the intrinsic calibration is used here to identify the position of the chessboard's bottom-left corner in each image. The recorded points, when plotted, will form a semicircle about the turntable's center of rotation. A plane can be fitted to this point set in OpenCV using an open-source algorithm based off the least squares method. A translation vector $\mathbf{t}_c$ and rotation matrix $\mathbf{R}_c$ are calculated based on the center of rotation and the fitted plane's normal vector, respectively. These are used later in a crucial coordinate transform.

V. Software Detail

A. Device Communication

One of the primary roles of the Raspberry Pi is communicating with and controlling the other hardware components. As essentially the brain of the system, the Raspberry Pi is responsible for telling the stepper motor when to step, the camera when to capture, and the laser when to light.A. Device Communication

The system begins in the initialization phase, where the camera module is initialized and the GPIO pins for the stepper motor and laser are defined. From here, the system enters the main loop, the scan phase. The platform is rotated by sending a clock signal to the stepper motor since it operates on a rising edge clock. Being that two images are required for the image subtraction, we first send a signal of zero to the laser to turn it off and capture the first image. This is followed by sending a signal of one to the laser and capturing an image with the laser on. This process is repeated 400 times until a full rotation has been completed. The 400th rotation marks the end of the scan phase and the beginning of the processing phase, which begins with clearing the GPIO pins for future use. Outside of this step, the processing phase does not feature any hardware interaction and will be covered in detail in the following sections. The combined phases can be seen in Figure 6.

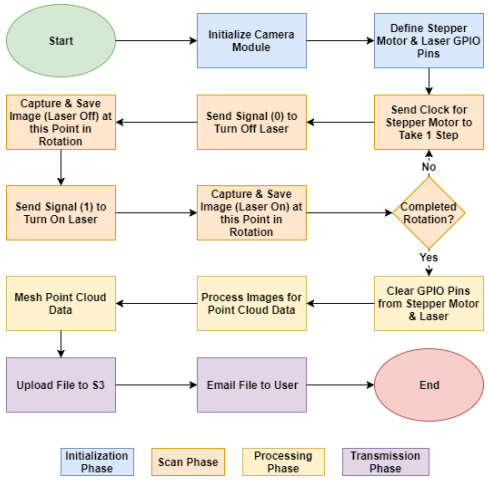

Fig. 6. High level flow chart of integrated software design.

B. Point Cloud Extraction

Extraction of point cloud data from raw images in our system is based around the principle of image subtraction. For each static position of the turntable, two images are captured in quick succession. The first image contains the object with the laser light incident on it, and the second image is of the same scene with no laser light present.

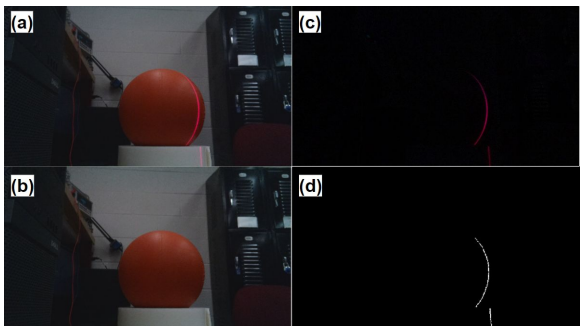

Fig. 7. An illustration of the simple image subtraction technique executed using our OpenCV script. (a) The object is illuminated by the laser. (b) The object is illuminated only by ambient lighting. (c) The image resulting from subtracting the previous two images. (d) The final segmented image after row-by-row peak detection algorithm is executed.

Assuming ambient lighting conditions and the background scene have not significantly changed in the dozen milliseconds separating the two images, all pixel values aside from the laser contour can be brought effectively to zero by subtracting the second image from the first, which are shown as figures 7b and 7a, respectively.. The resulting image is of a completely isolated red gaussian line laser profile, which is shown in figure 7c.

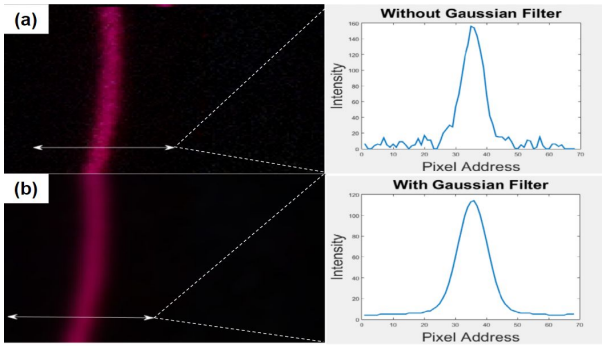

Fig. 8. A close-up of the laser profile (a) before applying a gaussian smoothing filter and (b) after applying the filter.

The next step toward extracting the point cloud data is to smooth the laser profile with a gaussian filter. The purpose of this is to reduce the influence of dark noise inherent to the inexpensive CMOS sensor as well as small inconsistencies in the beam shape as demonstrated in figure 8.

Finally, the peak detection algorithm can be applied to each row of pixels throughout the entire image. An intensity threshold filter is applied to each row's peak so that the dim maxima in rows not containing the laser profile don't make it into the point cloud. Each pixel address (u,v) above the threshold is flagged and stored in a 3xN array. These coordinates are subsequently converted to real units with the help of the calibration data.

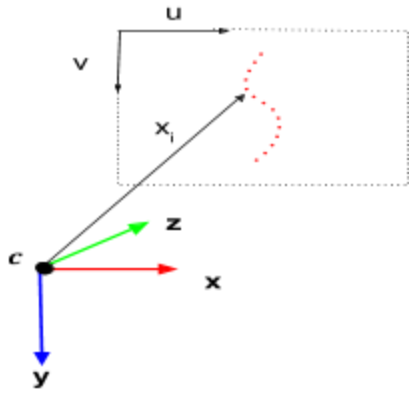

Fig. 9. Illustration of the unitless projection vector from the origin of the camera coordinate system ***c*** to the image plane.

The camera matrix K, previously obtained through intrinsic calibrations, will now be used to find a unitless projection vector **x** which is illustrated in

figure 9. This vector extends from the camera center through the image plane at each pixel address via the following relation,

$$x = K^{-1}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \left(\frac{u-c_x}{f_x}\right) \\ \left(\frac{v-c_y}{f_y}\right) \\ 1 \end{pmatrix} \qquad (3)$$

To arrive at coordinates in units of millimeters in the camera's reference system, the intersection of this projection vector with the laser's plane of incidence is to be calculated. The figures that were obtained during the extrinsic laser calibrations are used to achieve this:

$$X_c = \left(\frac{d}{n^T \cdot x}\right) * x \qquad (4)$$

where X_c is the three-dimensional location corresponding to a pixel address (u,v) in the camera's coordinate system in millimeters.

With the point cloud defined in terms of millimeters, all that remains is to perform the transformation from the camera coordinate system ***c*** to the turntable's world coordinate system w_o, which can be seen in figure 10.

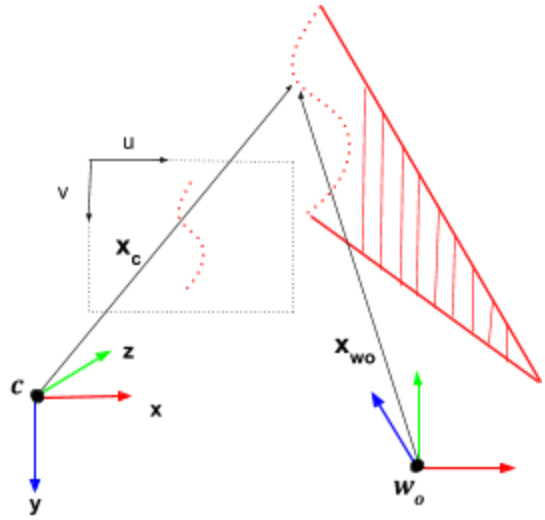

Fig. 10. Line-plane triangulation and the relationship between the camera coordinate system ***c*** and the world coordinate system w_o, which is centered about the turntable's center of rotation.

This homogeneous transformation from camera to world coordinates is performed using $\mathbf{R}_c$ and $\mathbf{t}_c$ obtained in the extrinsic turntable calibrations.

$$X_{wo} = R^T X_c - R^T t_c \qquad (5)$$

X_{wo} is the equivalent of X_c, only it is expressed using the center of the scanner's turntable as a point of origin.

From this point, all that remains is to rotate the point set to match the corresponding angle $\boldsymbol{\theta}$ which describes the turntable's current position with respect to its starting position. This angle is tracked throughout the execution of the software. The rotation is achieved by multiplying X_{wo} by the turntable rotation matrix:

$$R_z = \begin{pmatrix} \cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (6)$$

Finally, we arrive at our 3D point set described in the fixed world coordinate system in units of millimeters,

$$X_w = R_z X_{wo} \qquad (7)$$

which is written to a .ply file and subsequently used to construct a meshed surface.

C. Point Cloud Surface Reconstruction

The final process of the scanning system requires the transformation of raw point cloud data acquired from the scanning process. To restate, a point cloud is a simple collection of 3-D coordinates ascribed to an object. The resulting point cloud from the scan does not give enough information to reconstruct a scan as a solid object, which requires careful use of the several computational algorithms.

As stated, extracting the point cloud from an image provides a collection of discrete points which can be meshed together to form a surface along the point cloud. The point extraction process can be viewed as a continuous or segmented profile taken in slices along the rotational axis of an object.

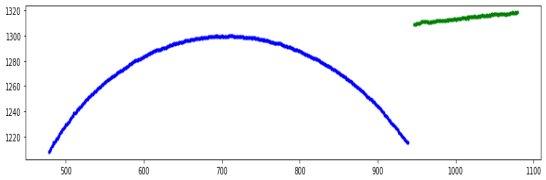

Fig 11. Plot of the disete pixels of a scanned ball at one slice of the scan with each identified segmented.

The meshing process begins with an implementation of search trees to find and store all the nearest neighbor points. This provides a fast and efficient storage data structure to access related points in order to calculate point normals between neighboring pixel points. The normal is calculated by creating pseudo-surface normal from the neighboring points and added to the original point cloud.

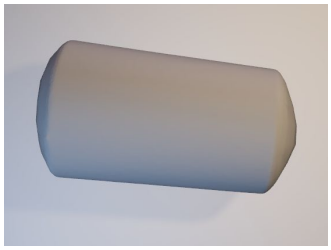

Fig 12. The surface reconstruction of can using a poisson reconstruction through MeshLab.

The next step is to utilize the calculated normals along with the point cloud data to apply a Poisson Surface Reconstruction algorithm. The algorithm uses an octree to store the point cloud where it computes the surfaces between points and applies a solid texture overlay. Once the surface has been reconstructed it can be exported as an STL file to view or 3D print. As can be seen to the image on the left, this is rendering after the point cloud has been processed through a normal calculation and Poisson surface reconstruction. Due to the camera's position the front facing surface is reconstructed as scanned, while the top and bottom cross-sections are not directly seen by the camera and as such must be extrapolated in the surface reconstruction algorithm.

D. Wireless Transmission

A system that requires a direct connection to a desktop or laptop increases the amount of constraint on the overall use of the product. Especially, a 3D Scanning System, that can sometimes have very lengthy durations, which makes it impractical to have the system directly connected to the user's device of choice.

This phase begins with uploading the meshed point cloud (STL) file to a private S3 bucket using our custom S3Uploader client. This client features an uploadFile method which returns a url to the uploaded file. Once this url is obtained, it is ready to be sent to the user. This is accomplished using our custom EmailClient which features a sendEmail function. Using this function, we send an email to the user containing the link to the meshed point cloud file, as well as a personal message from our predefined email address and SMTP configuration.

E. User Interface

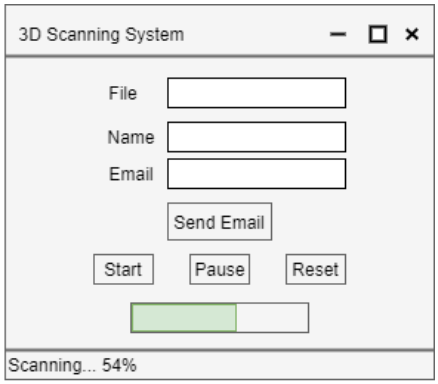

Fig. 13. User Interface during a scan.

As previously mentioned, difficulty of use is one of the major pitfalls of modern day 3D Scanners. As such, we aimed to change this by creating a scanner that does not require a technical background to operate. This was accomplished through our easy to use interface. Users are initially prompted with the option to start their scan. Upon starting a scan, a progress bar appears offering a visual display of the state of your scan. For a more concrete indication of progress, users can refer to the status bar located in the bottom of the screen. While scanning, users are able to pause or reset a scan as they deem fit. This allows users to make adjustments, for instance increasing or decreasing lighting, ultimately resulting in the highest quality scan. After point clouds are constructed and meshed, the file, name and email input fields are enabled, allowing users to transfer their file. By simplifying the interface, similarly to what one would find on a typical printer, the scanner can be unboxed, plugged in and ready for anyone to use.

F. Conclusion

The scope of the project allowed for the exploration of a versatile and low-cost implementation for a 3D scanner. The scanner aimed to be fully automated process for scanning small objects and producing a 3D printable file for personal use. The use of the Raspberry Pi allows for greater versatility throughout the various processes which happen during the scan. It leaves open the possibility of add-ons for changes to the scan such as additional lasers and the use of different surface reconstruction techniques.

Engineers

Cary McEwan is currently a senior student at the University of Central Florida. He plans to graduate and receive a Bachelor's of Science degree in Computer Engineering in December of 2017. Cary will also be graduating with a minor in Business Administration. Outside of school, his interests include basketball, music and sneakers. He interned with Amazon in the Summer of 2017 and will join them in March as a Software Development Engineer.

Samuel Benjamin, a senior at the University of Central Florida, is set to graduate in December 2017 with a Bachelor's of Science in Photonic Science and Engineering. Professionally, his interests center around lasers, imaging, and 3D metrology. Beyond his professional life, Sam has played drum set for over a decade, and more recently he has picked up guitar and bass in his spare time. Currently seeking employment, he is open to working in nearly any of the myriad photonics-related subfields.

Sommer Hilliard is currently a senior at the University of Central Florida. She plans to graduate with her Bachelor's of Science in Electrical Engineering in December of 2017. Outside of school, she enjoys running, playing soccer, and anything outdoors. She has gained valuable experience in her previous internships at Circuitronics Corp, Earthrise Space Foundation, and currently at Orlando Utilities Commission (OUC). She is now seeking full-time employment and is interested in power design.

Isaias Velez is currently a senior at the University of Central Florida. He plans to graduate with a Bachelor's of Science in Electrical Engineering in December of 2017. His professional interests are in software engineering, control and power systems, and optimization. Outside of school he enjoys camping, hiking, and playing the classical guitar. Isaias has gained valuable experience as a former Data Science Intern at Leidos' Energy Division, Undergraduate Research Assistant, and at his current internship at Cole Engineering Services, Inc. where he will transition to a full-time Software Engineer.

References

[1] http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html