3D Scanning with a Calibrated Stereo Camera Array

Simón Lorenzo
Stanford University
Department of Electrical Engineering
slorenzo@stanford.edu

Kevin Johnson
Stanford University
Department of Electrical Engineering
Kmjohnso@stanford.edu

Abstract

We design and test an automated low-cost scanner for producing three-dimensional models of sub-meter scale objects with millimeter-scale resolution. We mount stereo camera pairs on an aluminum scanning arc controlled by a stepper motor. These camera pairs acquire image pairs of a centered object as the motor rotates the aluminum arc, performing a scan. By first calibrating these pairs of low-cost internet-protocol cameras, we can convert the pixel disparity between image pairs to depth. We process these calibrated depth maps to reduce noise and remove holes. We then generate aligned point clouds from each of these calibrated depth maps and merge them. By further processing the merged point cloud, we generate a color-correct 3D object model with millimeter-scale resolution.

1. Introduction

1.1. Motivation

Our primary motivation is to develop a low-cost and fully automated three-dimensional (3D) scanner for producing 3D models of sub-meter scale objects with millimeter resolution. Current 3D scanning technologies are high-cost and not entirely automated [1,2]. Our final 3D scan could then be converted to the appropriate file format for existing 3D printer technologies. Ultimately, our device would allow rapid and accurate scanning and printing of any appropriately size object.

1.2. Related Software

In this project, we will use existing software to interface with, collect, and process image data from commercially available hardware. We will be using the MATLAB Computer Vision Toolbox for calibrating images and generating point clouds from stereo pairs of ELP Megapixel Mini™ IP cameras [3-7]. We will collect data from these cameras using the open source ONVIF device manager and iSpy surveillance software [8,9]. The scanning arc will be controlled using the TIC stepped motor controller and software [10].

1.3. Related Depth Mapping Systems

The depth mapping capabilities of our system are comparable to those of the Intel® RealSense™ D435 depth camera seen in Figure 1 [11]. This commercially available camera uses active infrared stereo vision. It generates infrared laser light, projects it onto the scene, and uses the infrared-sensitive stereo cameras to determine depth. It also records visible light data which can be overlaid onto the depth map. This device outputs a 1280 x 720 pixel resolution depth map in real time [11].

In contrast, our proposed design uses only the visible light data recorded from pairs of stereo cameras. By thoroughly calibrating these camera pairs, we can convert the pixel disparity between left and right images to depth. This reduces cost by eliminating the need for infrared laser sources and detectors. Our static depth maps are 1280 x 720 pixels.

Figure 1: Intel® RealSense™ Depth Camera.

1.4. Related 3D Scanning Systems

Our envisioned final product is similar to stereo scanning technology already on the market. The David Visions HP 3D-scanner in Stanford's Product Realization Laboratory (PRL) is one such example (Fig. 2) [1,2]. This scanner projects an array of known light patterns onto an object surface. The 3D structure of the surface is then determined using the distortion in the patterns. This technology can achieve sub-millimeter resolution, but costs upwards of four thousand US dollars [2]. This resolution is below our target millimeter-scale resolution, but we can reduce costs by using only IP camera pairs for data collection. This 3D scanner is also not entirely automated, as the object must be rotated manually to yield a full scan. Our scanner autonomously moves about the object to reduce the work load on the user.

Figure 2: David Vision HP 3D Scanner.

2. Methods

We combine existing hardware and software techniques in novel configurations to create our functioning 3D scanner. We begin by describing the hardware setup of the constructed scanner. Afterwards, we outline the high-level software process used to both calibrate the cameras and generate depth maps from stereo image pairs. Finally, we describe the image processing used to remove noise and fill holes in both the generated two-dimensional (2D) and 3D data sets.

2.1. Hardware

The high-level hardware system is described in the block diagram of Figure 3. A central computer with graphical user interface (GUI) interfaces with the stepper motor and the power over ethernet (POE) switch. The POE switch provides power to each of the IP camera pairs and allows data acquisition by the computer through a local network. Each camera was assigned an IP address through ONVIF Device manager [192.168.1.100 – 192.168.1.107] [8]. An external LED strip was added to help illuminate the scanning bed. A power supply located in the scanner base provides the necessary power for all elements.

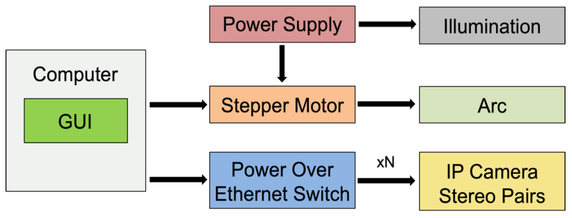

Figure 3: High-Level hardware system design. A single central computer with graphical user interface (GUI) controls the IP camera pairs, scanner arc motion, and scene illumination.

The scanner consists of several hardware components: the IP cameras, aluminum scanning arc, aluminum stage, POE switch, motor control driver, motor with gearbox, 3D printed motor mount, 3D printed aluminum to motor shaft interfaces, and a computer (Fig. 4). The IP stereo camera pairs are first mounted to an aluminum arc (Fig. 5). The aluminum arc was custom made in the PRL by putting it through a metal rod bender. The arc is free to rotate via D-shaft rods which are attached to the scanner's frame through pillow bearings. One of these D-shafts is coupled and up-shafted to and rotated by the stepper motor during the scan (Fig. 6). The computer controls the stepper motor using a Tic stepper motor controller chip [10]. The rotation occurs about the object of interest which is placed on the center of the scan bed (Fig. 4). The central computer acquires image data from each of the IP cameras via ethernet cables through the POE switch enclosed in the server rack mount hardware. A Cyberpower surge protector with an industrial grade metal housing was used to power the system through a wall outlet 120V / 20A.

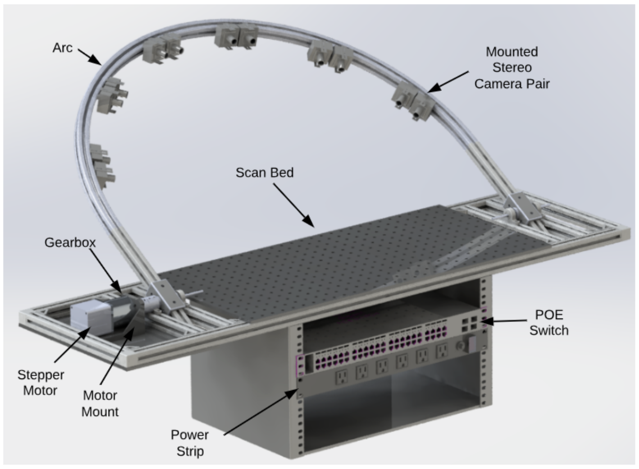

Figure 4: Rendered CAD Model of 3D Scanner.

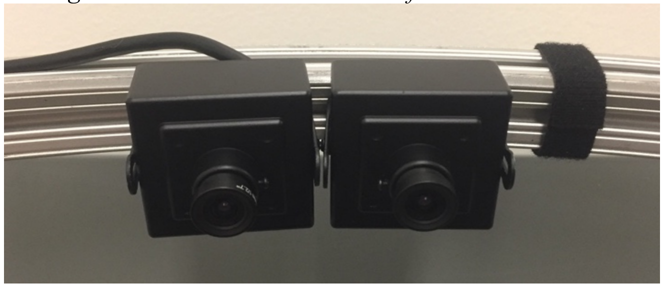
Figure 5: Mounted Stereo Camera Pair.

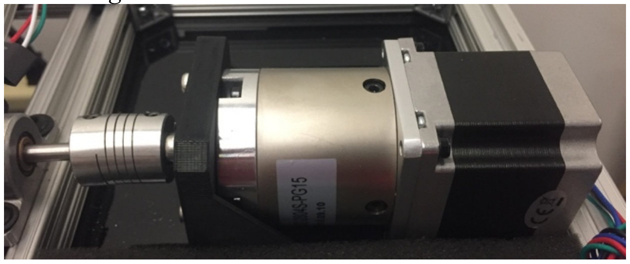
Figure 6: Stepper Motor.

2.2. Software

The software used for the scanner falls into two main categories: software for interfacing with the hardware interfacing, and software for processing the images.

Several software packages are used to interface with the scanner hardware. The Tic stepper motor controller chip comes with GUI software to control the motor rotation range and speed [10]. The ONFIV device manager is used by the computer to detect and assign IP addresses to the IP cameras [8]. ISpy home security software is used to view the real-time camera data and trigger imaging throughout the scan [9]. These images are saved on the computer for later processing.

The image processing was performed in MATLAB using both the Computer Vision toolbox [3] and additional disparity map processing functions [12]. A high-level overview of the 3D scanning process along with descriptions of the denoising and hole filling algorithms are provided below.

2.2.1 High-Level Software Overview

The camera calibration and three-dimensional (3D) point cloud generation are outlined in the high-level software block diagram of Figure 7.

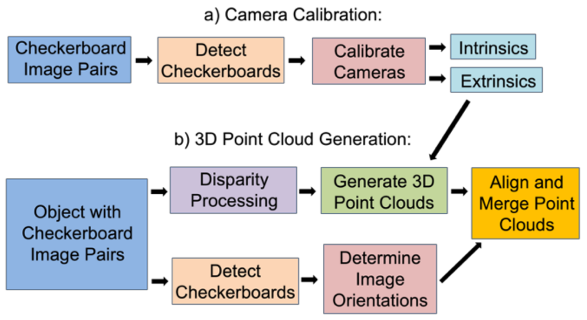

Figure 7: High-level software design: a) We calibrate pairs of stereo cameras to provide the camera intrinsic and extrinsic data necessary to create depth maps. b) We capture stereo pair images of both an object and a small checkerboard. We then compute and process the disparity map for each of the image pairs and generate a 3D point cloud using the calibration data. These point clouds are then aligned using the small checkerboard and merged.

The pairs of stereo cameras mounted onto the aluminum arc (Fig. 5) are calibrated using a reference checkerboard pattern of known square size (Fig. 7a). This determines the camera intrinsic and extrinsic parameters. Extrinsic camera data describes the 3D transform from the world coordinate system to the camera coordinate system. Intrinsic camera data describes the transform from the 3D camera coordinate system to the 2D image plane. Intrinsic data accounts for camera focal length, distortion, and image skew due to viewing angle.

To generate a merged 3D point cloud of an object of interest, we begin by taking stereo image pairs of the object during the scan (Fig. 7b). We compute the pixel disparity between these images by using a block search algorithm. This algorithm searches for similar regions between left and right images to determine their lateral pixel shift, or disparity. Disparity is dependent on the distance from the stereo camera pair. Closer objects have a larger disparity than farther objects. This disparity map is processed to reduce noise and fill holes using the methods detailed in the following section [12].

We then use the calibration data from Figure 7a) to convert this pixel disparity to a distance in mm. This strategy yields a partial point cloud for each disparity map. We combine the point clouds from different viewing angles to fully reconstruct the object. To determine the extrinsic transforms necessary to align these point clouds, we use the detect checkerboard function on a fixed alignment checkerboard next to the object of interest. Finally, the result of the autonomous scan is a merged 3D point cloud.

2.2.2 Disparity Map Processing

In order to improve the quality of the merged 3D point cloud, the disparity maps used to generate each individual point cloud went through two processing steps (Fig 8). First, the rectified stereo image pairs were put through a gaussian filter with a sigma of 3. This value was found through multiple trial and error runs with visual inspection. We further enhanced the results by pre-processing the disparity maps with a gaussian filter [12]. To remove unwanted holes, each disparity map was fed into a hole-filling function. This process is detailed in Figure 8.

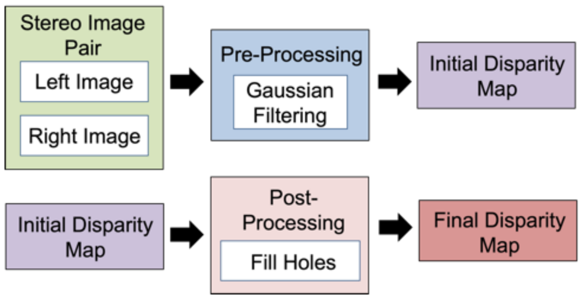

Figure 8: Software system diagram denoting pre and post processing steps used to remove holes and smooth out disparity maps.

2.2.3 Point Cloud Spatial Noise Removal

To further improve the quality of the final merged point cloud, we reduced unwanted spatial noise using the process described in Figure 9. We implemented a voxel density noise removal algorithm to visualize, mark and remove unwanted spatial noise. To help visualize the algorithm in real time, we color valid and invalid regions green and red respectively. The pseudo-code for the algorithm is detailed in Algorithm 1.

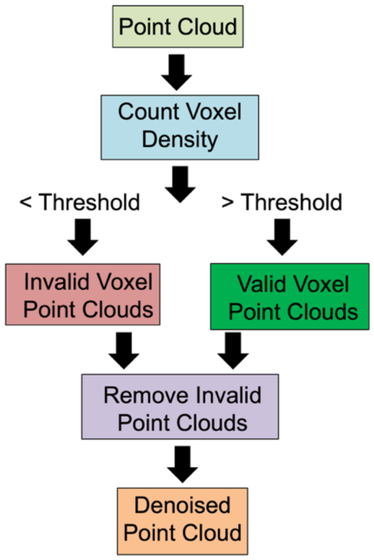

Figure 9: Software flowchart of point cloud spatial noise removal algorithm.

We first assume that lower density regions of the point clouds are noise. This is typically valid for any scanned solid object. We then establish the voxel size, dividing up the 3D point cloud into cubes. We create an empty point cloud to hold new point cloud data. Next, we count the number of points within each voxel region of interest (ROI) in the original point cloud. A user-defined count threshold determines validity. If the count is greater than the threshold, the points included within that voxel are merged into the new point cloud. If the count is less than the threshold, the points are discarded. This algorithm can be implemented on either the partial point clouds or the fully merged point cloud. However, it is computationally more efficient to run the algorithm on the final merged cloud.

Point Cloud Voxel Noise Removal Algorithm

```
1: Define voxel size
2: Define voxel threshold size
3: Define Voxel Step in X, Y and Z
4. Create empty point cloud to hold new data
5: For 0 : ZVoxelStep : Point Cloud X limits
       For 0 : YVoxelStep : Point Cloud Y limits
           For 0 : XVoxelStep : Point Cloud Z limits
6:               Define roi in terms of X, Y and Z limits
7:               Find all points within voxel
8:               Count data points within voxel
9:               if point cloud count < threshold
10:                  Mark as invalid
11:                  Color all points within voxel red
                 else
12:                  Mark as valid
13:                  Color all points within voxel green
14:                  Merge into new point cloud
           End
       End
   End
```

Algorithm 1: Pseudo code for the voxel noise removal algorithm.

3. Results

Below, we provide the graphical results of the hardware and software scanning processes described above. We begin with the calibration process, present the initial and processed 2D data, and conclude with the processed 3D data from the partial and merged point clouds.

3.1. Calibration

We used a 10 x 9 asymmetric black and white checkerboard with 18 mm square size to calibrate the camera pairs in the spatial region of interest (Fig. 8). Due to the sharp pixel intensity gradient, the corners of this pattern are easily localized by a detect-checkerboard function. The detected corners can be visualized by the red circles in Figure 10.

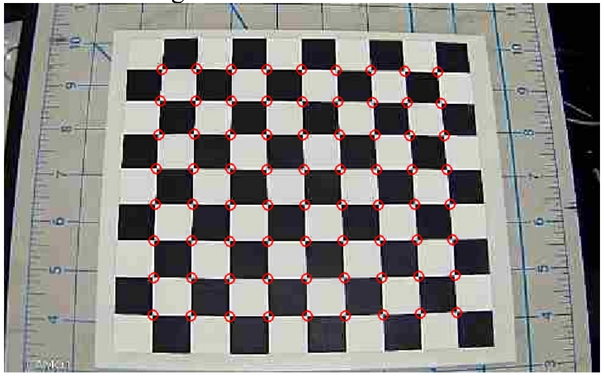

Figure 10: Calibration checkerboard with detected corners.

A thorough calibration over the spatial region of interest is necessary to reliably convert pixel disparity to distance. We image the checkerboard over multiple scan angles and object heights. A visualization of the detected checkerboards for a single angular sweep is shown in Figure 11. The extrinsic visualization from a full calibration sweep requires additional board heights (Fig. 12).

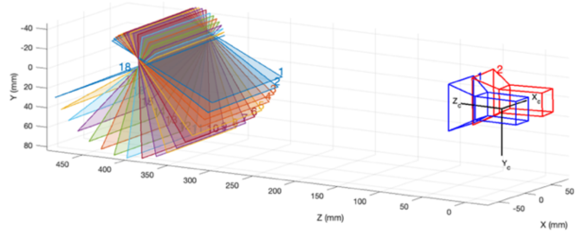

Figure 11: Visualization of the calibration checkerboard and camera positions during a scan.

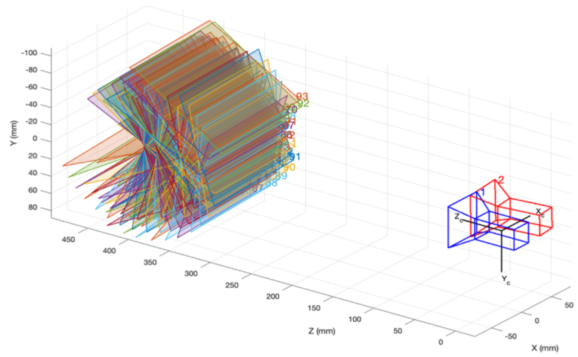

Figure 12: Visualization of the calibration checkerboards and cameras during a full calibration. Note that overlaid angular scan data for 4 board height positions is shown.

3.2. 2D Scan Data

We perform a scan of an object by acquiring stereo pair images from different viewing angles during a single mechanical arc sweep. Sample object stereo pair images acquired at two sweep angles are shown in Figure 13. A detected alignment checkboard is also shown in each image. The sample objects scanned and shown in this report are a roll of duct tape, a coffee mug, and a pair of laser safety goggles. These are difficult objects to image due to their smooth untextured surfaces.

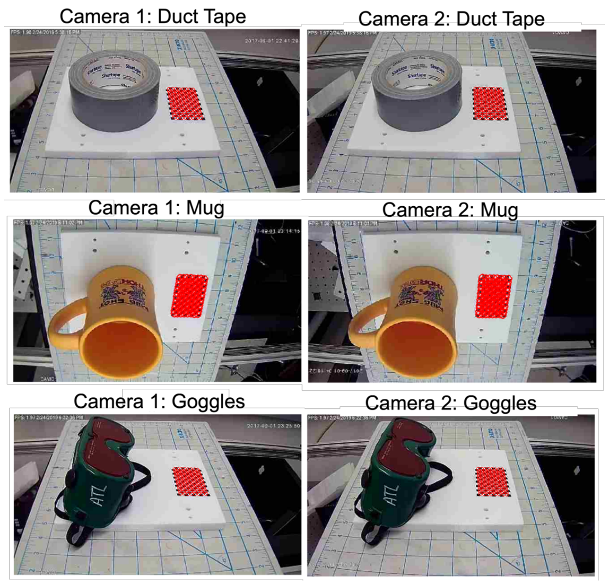

Figure 13: Sample scanner stereo image pairs of the duct tape roll, the coffee mug, and the laser goggles. The detected checkerboard is used to determine the orientation of the selected images.

The disparity algorithm provided in the MATLAB Computer Vision toolbox uses a block-search method for locating objects in each image pair. The disparity maps of Figure 14 are visualizations of the pixel disparity between stereo images. We depict two viewing angles for each object. This disparity estimation algorithm is simple and yields patchy but still recognizable objects. As expected, close object surfaces show a larger pixel disparity (Fig. 14).

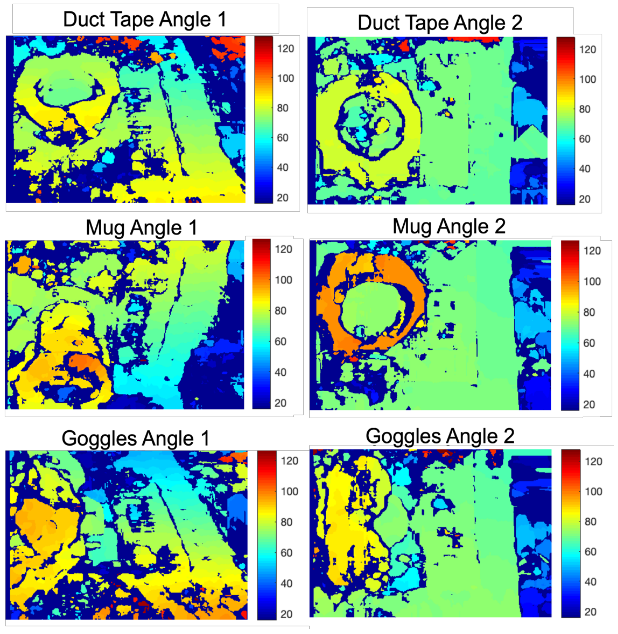

Figure 14: Samples of disparity maps for two different angles during scans of the duct tape roll, the coffee mug, and the laser goggles.

These disparity maps can be improved significantly using the hole-filling algorithm described in section 2.2.2. Figure 15 shows a sample implementation of the algorithm on a disparity map of each object. Most holes and gaps in the map are entirely filled.

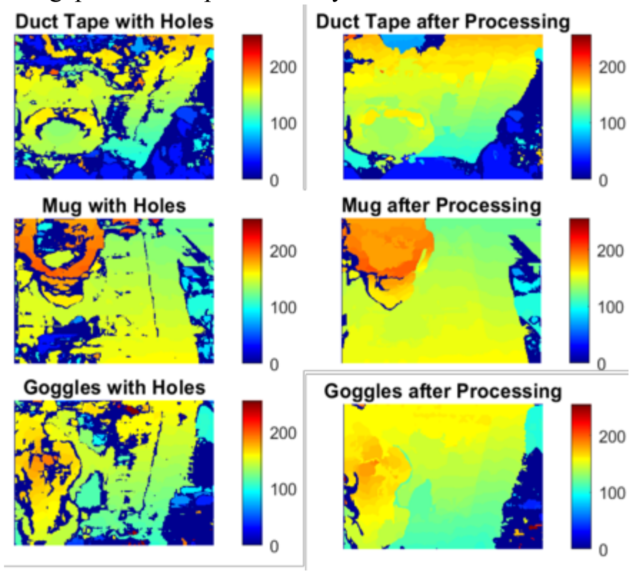

Figure 15: Input disparity maps (left) and output final disparity maps after processing (right)

3.3 3D Scan Data

Using the extrinsic and intrinsic camera data from the calibration run, we convert the processed disparity maps into point clouds. Sample point clouds from two viewing angles for each object are shown in Figure 16. The clouds on the left are from side-on images of the objects. The clouds on the right are from top-down images of the objects. This is reflected in the surface data on each object.

These clouds are aligned and transformed using the small checkerboard depicted in each image (Fig. 13). Here, we are only using the extrinsic data, not performing a new calibration run. Though the point clouds were initially rotated assuming a static camera, we can easily undo the spatial transform to reflect a static object located at the grid origin (Fig. 13).

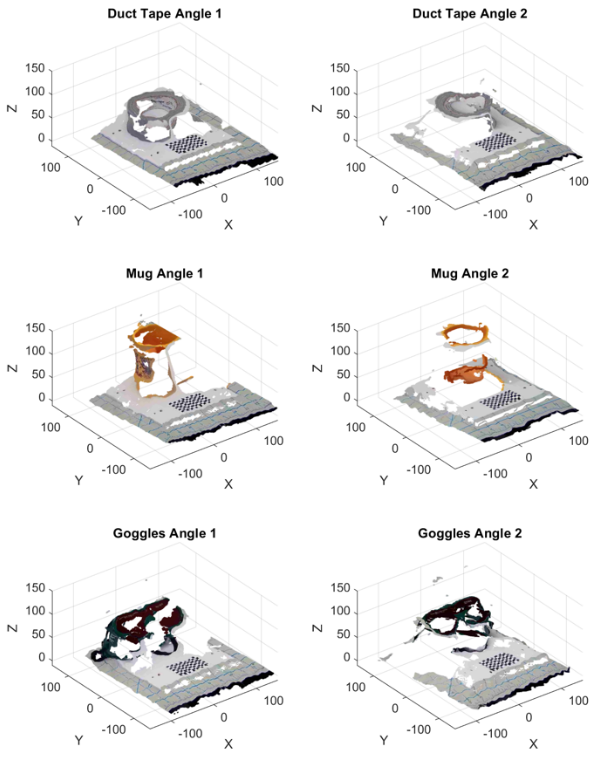

Figure 16: Samples of point clouds for two different angles during scans of the duct tape roll, the coffee mug, and the laser goggles. All are aligned with each other using the small neighboring checkerboard. Point clouds on the left are from side-on images, and point clouds on the right are from top-down images.

Due to the partial nature of each point cloud, we combine them all to generate a full 3D map of each object. We show the merged result of combining several point clouds in Figure 17. Note that many of the holes are filled with data from different viewing angles.

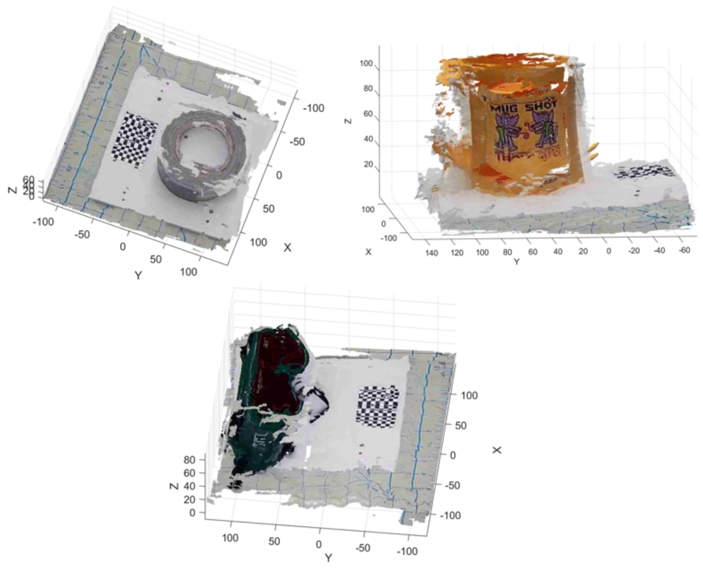

Figure 17: Merged point clouds of the duct tape roll, the coffee mug, and the laser safety goggles. Combining the partial point clouds of Figure 16, filled in the holes and completed the object.

3.4 Denoised 3D Scan Data

To remove any additional spatial noise in the 3D point clouds of Figure 17, we implement the voxel denoising algorithm from section 2.2.3. This is also useful for removing any valid scan data outside the region of interest. An example of the voxel denoising algorithm at work is shown in Figure 19.

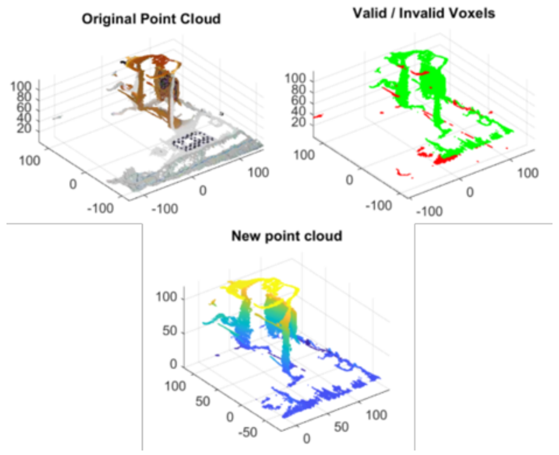

Figure 19: Single Point Cloud (Top Left), valid (green) and invalid (red) single point cloud (Top right) and a reconstructed single point cloud with noise removed (Bottom)

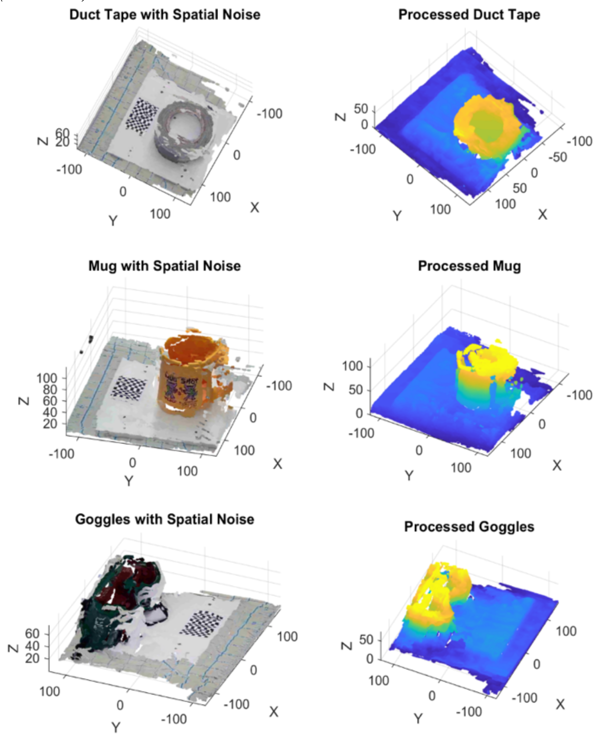

Figure 20: Merged point clouds before and after the voxel denoising algorithm.

Figure 20 shows the results from using the voxel denoising algorithm on the fully merged point clouds. The new reconstructed point cloud shows no sign of the small spatial noise clusters.

4. Analysis and Discussion

The effectiveness of this scanner can be judged through analysis of the calibration and alignment method, the disparity map processing, the resulting 3D scan dimensions, and the scan color.

4.1. Pixel Disparity

Both the camera calibration and point cloud alignment methods rely on the recognition of checkerboards in the image. The reprojection errors represent the accuracy of the detect checkerboard function implemented for each stereo pair image. These errors describe how well the calculated transformed checkerboard aligns with the actual corners detected in the images. Values under 1 pixel represent effectively errorless images. All reprojection errors for the calibration and alignment images in this paper were sub-pixel. This is therefore, not a dominant source of error in the scan.

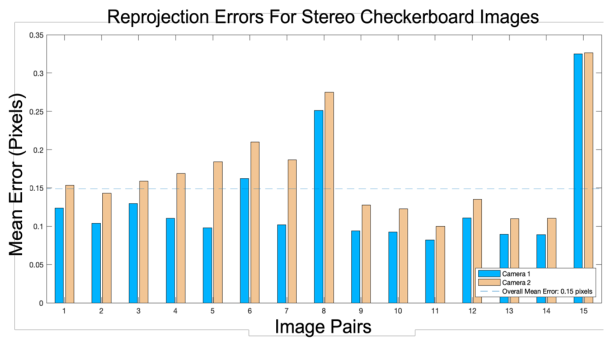

Figure 21: Reprojection errors for sample calibration data. All errors are less than one pixel.

4.2. Noise and Hole Processing

The original disparity maps suffer from spatial noise and holes (Fig. 14). For relatively smooth surfaces the simple disparity algorithm has difficulty in tracking objects between images. This explains the holes in Figure 14. However, the hole-filling algorithm successfully addresses this issue (Fig. 15).

The voxel noise removal algorithm effectively discards both spatial noise data and points outside the region of interest (Fig. 20). For generating a 3D print file, this method could be used to subtract the stage and platform.

4.3. Scan Dimensions

The resulting dimensions of each 3D scan are also an effective evaluation method for performance. Measured key features for each object are compared with the same features in the resulting scans of Figure 17 (Table 1). Beyond preserving the qualitative object shapes, the differences for all features of all objects are sub-centimeter. This achieves the project goal of imaging sub-meter scale objects with millimeter resolution.

Object	Key Feature	Measured (cm)	Scanned (cm)
Duct Tape	Width	11.4	11.9
	Height	4.9	4.4
	Thickness	1.9	1.9
Mug	Width	9.3	8.75
	Height	11.0	11.3
	Thickness	0.7	0.5
Goggles	Width	7.0	6.6
	Height	6.3	5.9
	Length	15.2	15.7

Table 1: Dimensions of key object features: measured versus scanned

4.4. Scan Color

The color of each object is preserved well spatially. Larger colored areas of the duct tape, coffee mug, laser goggles, and background are in place. Smaller colored feature patterns including text on the coffee mug and silver writing on the goggles are also still legible.

4.5. Comparison to Previous Work

Our 3D scanner achieves reliable millimeter resolution over an entire merged 3D point cloud. This resolution is comparable to that of the Intel® RealSense™ Depth Camera (Fig. 1) over the distance of interest. However, we successfully automate the scanning and merge the point clouds into a full 3D model. The RealSense™ only provides depth data from a single viewing angle.

Our scanner shows improvement over the HP David Visions scanner through improved automation and reduced cost but suffers from reduced resolution. We do not achieve the same sub-mm resolution. However, the overall cost of our prototype is below $600 US dollars. The HP David visions scanner costs above $4k, so we succeed in providing a more economical alternative.

5. Future Work

Future implementation of this work may focus on further automation, GUI integration, and improved disparity map processing. For sufficiently denoised 3D scans, we may also advance to 3D printing the results.

While the scans themselves are automated, the data processing is not automatically triggered upon scan completion. We have to access a local version of MATLAB for data processing. We are also currently using two separate GUIs for stepper motor control and imaging. It should also be possible to combine these two separate processes. Ultimately, the entire process will be fully autonomous, controlled through a single button click, and integrated into one GUI. This will produce a high-resolution STL file for rapid 3D printed prototyping.

References

[1] Stanford University. (2018). 3D Scanning. *Stanford Product Realization Lab.* Retrieved from https://productrealization.stanford.edu/resources/processes/3d-scanning.

[2] David Group. (2019). 3D Scanner Introduction. *David and HP Company.* Retrieved from https://www.david-3d.com/en/support/david4/introduction.

[3] The MathWorks Inc. (2019). Computer Vision Systems Toolbox. *MathWorks.* Retrieved from https://www.mathworks.com/products/computer-vision.html.

[4] Zhang, Z. "A Flexible New Technique for Camera Calibration". *IEEE Transactions on Pattern Analysis and Machine Intelligence.*Vol. 22, No. 11, 2000, pp. 1330–1334.

[5] Heikkila, J, and O. Silven. "A Four-step Camera Calibration Procedure with Implicit Image Correction", *IEEE International Conference on Computer Vision and Pattern Recognition*, 1997.

[6] G. Bradski and A. Kaehler, "Learning OpenCV : Computer Vision with the OpenCV Library," O'Reilly, Sebastopol, CA, 2008.

[7] Amazon. (2019). ELP 1280720p 1.0 Megapixel Mini IP Camera, Mini Hidden Network Camera. *Amazon.* Retrieved from https://www.amazon.com/ELP-1280720p-Megapixel-Camera-Network/dp/B00KA4M4WS.

[8] Slashdot Media. (2019). ONVIF Device Manager. *SOURCEFORGE.* Retrieved from https://sourceforge.net/projects/onvifdm/.

[9] DeveloperInABox. (2019). Open Source Video Surveillance Software. *ISPYCONNECT.* Retrieved from https://www.ispyconnect.com/.

[10] Pololu Corporation, (2019). Tic Stepper Motor Controllers. *Pololu.* Retrieved from https://www.pololu.com/category/212/tic-stepper-motom-controllers.

[11] Intel. (2019). Intel RealSense Technology. *Intel Software Developer Zone.* Retrieved from https://software.intel.com/en-us/realsense.

[12] Aditya, K.p., V. Reddy, H.Ramasangu., "Enhancement Technique for Improving the Reliability of Disparity Map under Low Light Condition." *Procedia Technology*, vol 14, 2014