Special Project Final Report

# *Single-Camera 3D Microscope Scanner*

Made by **Alejandro Vázquez Marino**
and directed by **Boyang Wang** and **Jafar Saniie**

**Illinois Institute of Technology**
*2017-2018*

ECASP

# ABSTRACT

In the last decade, computer vision and image processing technologies have been a central point for researchers due to their applications in multiple fields. In particular, microscope image processing is extremely relevant in fields such as medicine, biological research and metallurgy. This document describes in detail the series of methods and algorithms we have used for building a single-camera 3D microscope scanner system, which generates 2D and 3D composite very large resolution images of the scanned objects.

# TABLE OF CONTENTS

# 1. INTRODUCTION

The use of digital image processing techniques on images obtained from microscopes dates back a half century, when some of the techniques first developed for television were explored to process and analyze microscope images. Today, modern computing platforms and their increased processing power, speed and memory allow for a wider and more complex range of applications in fields such as medicine, biology or metallurgy.

This project applies microscope image processing to build a single-camera 3D microscope scanner. The system will scan the surface of an object by sliding a digital microscope camera over a given area at quantized steps, taking a picture of the surface at each step. Then, a software will process these pictures to combine them into a single very large resolution image that features the whole scanned area by applying image stitching techniques. Finally, the software will apply 3D reconstruction techniques to generate a depth map and a point cloud of the scanned surface, extracting depth information from the overlapping areas between images taken by the sliding camera.

The purpose of these composite very large resolution images is to provide the user a single picture with all the information of the scanned surface, so that the global context can be easily perceived, and the user can jump between local areas without having to deal with multiple images. Both the final stitched image and the depth map information can be input to posterior image processing algorithms to perform operations such as classifying, counting or measuring.

In this chapter, we will introduce some of the fundamentals on which the project is sustained. These include some concepts related to image acquisition such as microscopes and charge-coupled devices (CCDs), previous research on image stitching applied to microscope images, and epipolar geometry for 3D reconstruction.

## 1.1. Microscope Camera and CCDs

Images are captured using a very simple digital microscope camera. This camera is composed by a small magnifying lens that directs the light towards a charge-coupled device (CCD). These devices feature a grid of cells that get charged by incoming photons during a charging phase. Then the stored value gets shifted vertically one by one towards a horizontal register, and then horizontally towards the output, generating an output signal that represents how intense the light was during the charging phase for each cell. That is, the output signal holds the intensity value for each pixel in an image.

The distance from the microscope lens to the object is adjustable, allowing for different zooms. However, the camera we used in this project does not include adjustable focusing, so it only produces focused images when the object is at two certain distances from the lens, which in effect only allows for 20X and 800X zoom.

## 1.2. Previous Research

This project takes from some of the ideas presented in [1]. In that paper, the authors describe a method for image stitching based on mosaic tiles. Starting from the center, images are stitched together by sliding new images over previously stitched images until a best match is found. The best match is determined by computing the normalized cross correlation coefficient, defined in equation (1):

$$cross - correlation = \frac{\sum_{x=0}^{L-1}\sum_{y=0}^{K-1}(w(x,y) - \bar{w})\left(f(x+i,y+j) - \bar{f}(i,j)\right)}{\sqrt{\sum_{x=0}^{L-1}\sum_{y=0}^{K-1}(w(x,y) - \bar{w})^2}\sqrt{\sum_{x=0}^{L-1}\sum_{y=0}^{K-1}\left(f(x+i,y+j) - \bar{f}(i,j)\right)^2}} \quad (1)$$

The stitching order goes from the center to the edges in a spiral pattern, assuring that the information from the center of the scanned area gets stitched first. This order was decided because of the nature of the samples that were being scanned, which generally only occupied the center of the scanned area. Border images represented featureless, flat areas that are harder to stitch because of the lack of references. The maximum cross-correlation coefficient for these kinds of images will be lower than for images with multiple and/or strong features. The authors assume that any coefficient value below 0.7 represents an incorrect match, and thus leave a 'hole' in the composite image when a value above 0.7 cannot be found. Later, they perform a second pass where they try to fill these holes by sliding the missing images again when more images have been stitched.

The blending method that is used in our project is also inspired in the one used in [1]. There, the authors use a gradient blending method that computes the intensity values of the overlapping areas of the new composite image as a weighted sum of the pixels from the previous composite image and the pixels from the new image that is being stitched. The weighting (α) for the new image is calculated as a distance from the image edge, going from 0 to 1 as the distance increases. The weighting for the previous composite image will be the opposite of that of the new image (i.e. α-1).

It is also worth mentioning that during the first attempts to assemble an image stitching software OpenCV's stitcher class [2] was tried. This class implements a complex pipeline that is based on feature finding and matching. The problem with this approach is that while it is very well suited for images taken under different conditions (e.g. panoramic images taken manually with a regular camera or a smartphone with different camera poses), such a level of complexity is not required in our system, where images can be assumed to suffer from no rotation effects. This class was both slower and less accurate than the cross-correlation approach, as it will later be shown in this report.

## 1.3. Epipolar Geometry

Epipolar geometry describes the geometric relations between the 3D points of a scene and their projections onto the 2D images taken by a pair of cameras observing that scene. These relations allow us to extract depth information from a pair of stereo images, just like the human brain does with the information captured by the eyes.

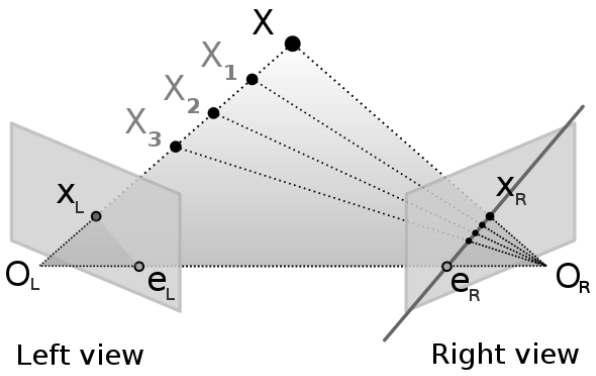


***Figure 1.1.*** *Epipolar geometry*

The fundamental relations in epipolar geometry are illustrated in the figure above. A 2D point in the left view, $X_L$, can be at any distance from the image plane, but we know it will be contained in the direction of the line that goes from the left camera optical center, $O_L$, to the 2D point $X_L$. If we project that line on to the right view that will give us a 2D line in the right image where the point $X_L$ from the left view must be contained. This is named an epipolar line, and it gives us all the possible locations of $X_L$ in the right view.

We can notice that this epipolar line and the point $X_L$ form a plane that contains both cameras' optical centers. This is called an epipolar plane, and it can be easily deduced that for any point in the left view its corresponding epipolar plane will contain both optical centers as well. That is, epipolar planes spin with the baseline (the line $O_L$-$O_R$ that unites both optical centers) as its axis. This means that all epipolar lines, which must be contained within an epipolar plane, will pass through the common point that all epipolar planes share with the right view: the point where the baseline crosses the right image plane, $e_R$.

Therefore, all epipolar lines in the right view converge in $e_R$. As such, if we want to find a point from the left view in the right view to determine the depth of the point, all we have to do is find the epipolar line in the right view corresponding to that point, knowing that the line converges in $e_R$, and then perform a search along that line to find a best match with the left point.

This process will be simpler in the case both the left view and the right view are contained within the same plane and the baseline is parallel to that plane (i.e., the focal distances for both cameras are the same). In this case, the baseline never crosses the image planes, and thus epipolar planes will converge in the infinite. As such, epipolar lines will be parallel, and the search can be performed in horizontal rows.

Generally, stereo vision systems will be rectified so that this condition is satisfied in order to simplify the search. In our case, the view planes can be assumed to be perfectly contained in the same plane as we do not use a pair of cameras but a single camera that is slided along a plane, so there is no need to rectify the images.

# 2. PROJECT DESCRIPTION

In this chapter we will describe in detail the different procedures and algorithms used in the project. First, we will introduce an overall description of the system, the different pieces of hardware that compose it and their interconnections, and how the software controls these. Then, we will explain the image acquisition procedure. We will continue detailing the image stitching algorithm, including the blending method that we used and some commentary on the observations that we made testing the algorithm on different kinds of surfaces. Finally, we will discuss the 3D reconstruction algorithm, and how it uses the information acquired during the stitching process.

## 2.1. System Description

An illustration of the different components of the system can be found in the figure below. The central component is the computing platform, which runs the software that will send orders to the other components and will process the captured images. The software has been coded in Python and uses some functions from the OpenCV library. The computing platform is directly connected to the digital microscope camera and sends orders to the CNC machine that moves the camera through an Arduino board running GRBL, a controlling software for machine motion. The connection between the computing platform and the Arduino is serial, and messages are sent in G-Code, a language for numeric control commonly used in automation.

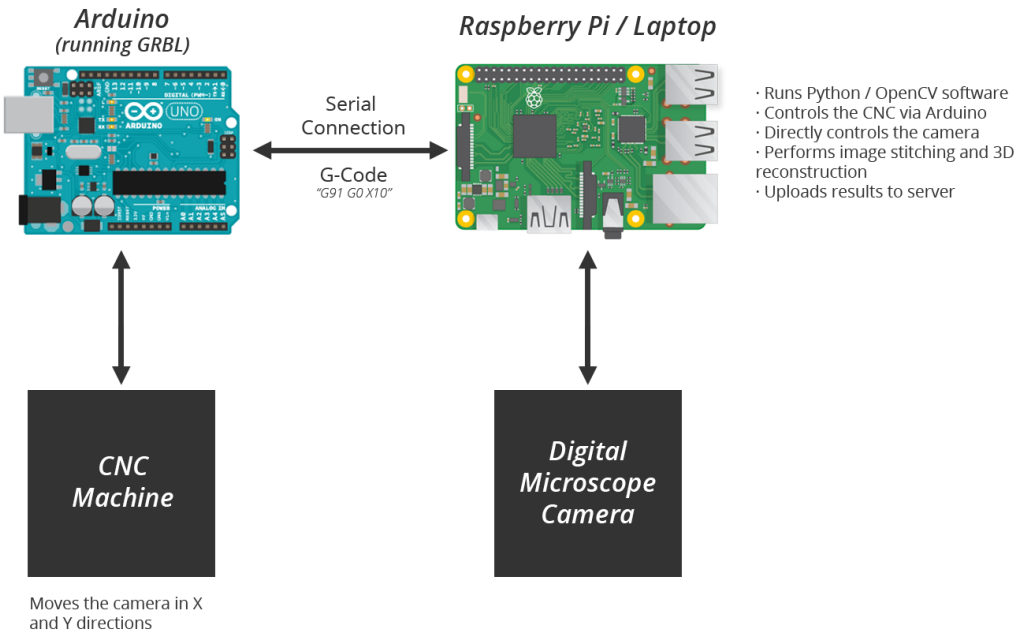


***Figure 2.1.*** *System schematic*

## 2.2. Image Acquisition

The image acquisition procedure is quite simple. The computing platform will send a command to the CNC machine through the Arduino controller to move in a certain direction for a given step size, and then it will capture and store an image from the microscope camera. The relative coordinates of the image of the captured image are saved as the filename of the image, which has the format *"x-y-.png"*. This process is repeated following a zig-zag pattern until the whole area, which size is input by the user, has been scanned. All images are saved in a folder named after the date and time when the scan was initiated.

The value of the step size is extremely relevant for the correct performance of the later image stitching and 3D reconstruction steps. The step size will determine how big is the overlap between adjacent images, and a sufficient overlap is required for both of these steps. The 3D reconstruction step is the most critical, as we can only extract depth information from the overlapping areas between images. Then, if we want to combine or stitch that depth information together, a common overlap between the overlapping areas is also needed. A step size value that guarantees an overlap of about 2/3rds of the image's width has been found to be enough for the processing algorithms. A smaller step-size will result in increased scanning and computation time, and will reduce the baseline length for 3D reconstruction, which is not desired.

Because of the limitations of the microscope camera regarding the zoom and focus we previously commented, it made no sense to implement an automated selection of the step size. However, for a microscope camera with multiple available zooms it would be optimal to automate the step size selection based on either known information about the zoom or after a template matching search to determine the current overlap between images.

As a pre-processing step before following with image stitching, we will check the histograms of the acquired images to try to correct any image that is under or over exposed.

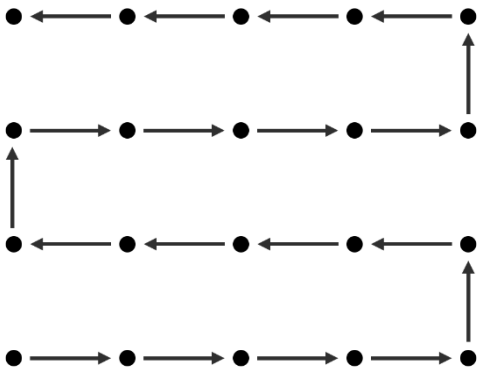

***Figure 2.2.*** *Zig-zag scanning pattern*

## 2.3. Image Stitching

The image stitching algorithm can be divided in two major steps: finding the new image position within the previous result and blending the new image with the previous result.

We use a template matching approach to find the new image position that is very similar to the one used in [1]. The new image is slided over the previous image until a best match is found, where the best match is determined by the normalized cross-correlation coefficient. In our case, we use OpenCV's template matching function [3], which implements equation (2):

$$R(x,y) = \frac{\sum_{x',y'}\left(T'(x',y') \cdot I'(x+x',y+y')\right)}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \qquad (2)$$

*where:*

$$T'(x',y') = T(x',y') - \frac{1}{w \cdot h} \cdot \sum_{x'',y''} T(x'',y'')$$

$$I'(x+x',y+y') = I(x+x',y+y') - \frac{1}{w \cdot h} \cdot \sum_{x'',y''} I(x+x'',y+y'')$$

This coefficient has a value that goes from 0 to 1, where 1 denotes a perfect match. The function returns a matrix with the coefficient values for each possible position of the template (i.e. the new image) over the previous image. The position of the new image will be determined from the location of the maximum value within said matrix. A colormap of a sample matrix returned by this function can be found in the figure below.

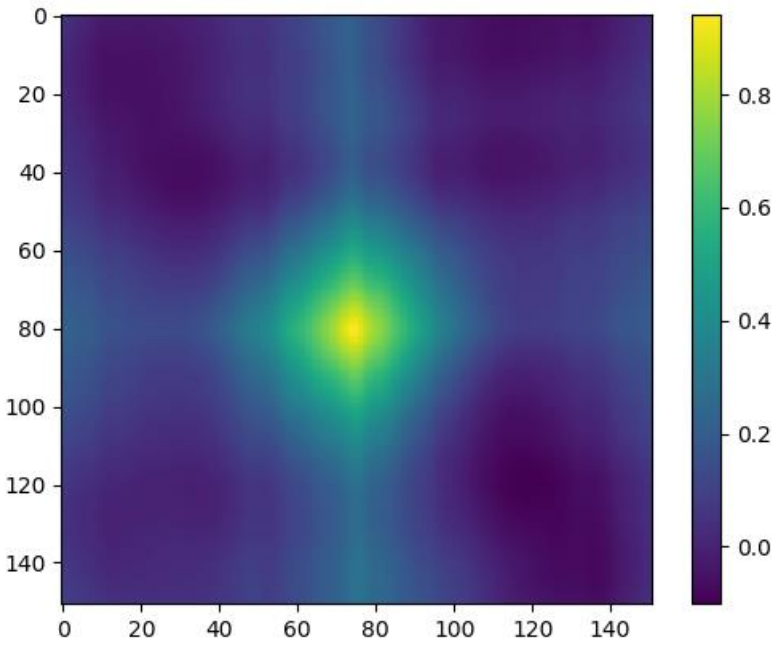


***Figure 2.3.*** *Colormap of a matrix holding correlation coefficient values after template searching*

This colormap shows a very distinct point where the coefficient has its maximum value. The colormap also shows that the image has some strong vertical and horizontal components, indicated by how the coefficient value decreases at a lower rate from the maximum to the edges in the direction of the vertical and horizontal axes.

Images with weaker features, those of flat surfaces with less contrast, will result in a colormap with a maximum that is not as sharp, and with a value that is not as close to 1. If the image is extremely flat the template matching might provide an inaccurate match. This is generally not a relevant problem, since we rarely want to scan flat surfaces with no relevant information.

A more relevant issue is associated with images with periodic patterns. These images will result in multiple peaks that are close in value to the maximum at the actual matching position, and in some cases they may even surpass that value, causing a matching error.

In order to decrease computation time and increase the probability of an accurate match, a reduced search method has been developed. The idea is that instead of sliding a template block from the new image over all the previous image, we can take advantage of the fact that the step size is constant and the overlapping areas between images should be similar. That is, if we have an approximate for what the overlapping area should be, we can simply search around that area instead of performing a full search.

The first step is to estimate this approximate value for the overlapping area between two adjacent images by performing a full search on any pair of images. The Y offset or position of the image will give us the width of the non-overlapping area for that pair. Now, for any new search we can reduce the search area to one that is centered at that position plus some margin or search range to account for background depth changes or camera motion imprecision.

The way the template and target blocks are defined is shown in the figure below. The target block from the left or previous image has a size equal to the estimated overlapping area plus a margin on the left side. The template block from the right or new image has a size equal to the overlapping area minus a margin on all its sides. The reason for applying the margin on all sides corresponds to the need to leave some room for sliding, and because of the lighting effects caused by the camera LEDs on the image borders.

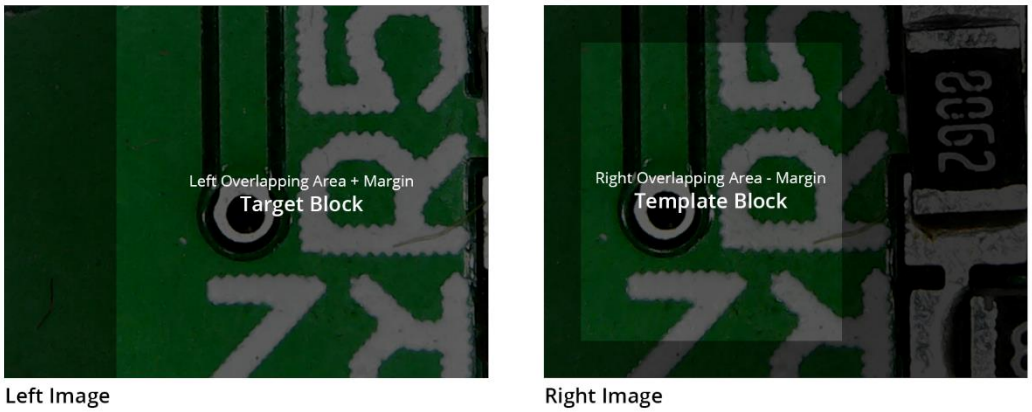


***Figure 2.4.*** *Reduced search based on estimated overlap*

This approach contrasts with the one used in [1], where the authors used only the pixels closer to the edges of the new image as their template block.

Once we have found the position of the new image we have to combine the information from both images into a single image. In order to eliminate any visible edges between the stitched images we will apply a gradient blending algorithm that combines the overlapping areas by computing the weighted sum of the left and right images. The algorithm does so in a way that each image offers a bigger contribution coming from the pixels that are closer to its center.

We have employed a method for blending that allows for very fast computation times while offering high quality results. The overlapping areas of each image are divided into *N* vertical slots or strips, assigning to each slot in the left image a weight *w* equal to *(N-n)/N*, and a weight equal to *1-w = n/N* for each slot in the right image. An illustration of how these weights are assigned can be found in Figure 2.5. Finally, the stitched overlapping area is formed as the weighted sums of corresponding slots from the left and right images. The non-overlapping areas will simply take their value from the original images.

Once the images have been stitched, any vertical offset in the new image position will cause the resulting image to be non-rectangular, with blank areas at the top and bottom edges. We crop the resulting image reducing its height to form a perfectly rectangular result, for visual ease. In practice, we actually crop these areas from the original images before blending for simplicity, and we only blend the areas that we know will not need to be cropped.

The images in the following page show the difference between blended and unblended results, as well as some cropped results. These images date from the early iterations of the blending algorithm, so there are some visible errors that are no longer present in the current version of the algorithm, as it will be shown later. The reason for choosing this images is that we had no examples of unblended results for the current version of the algorithm, and we wanted to show those for contrast.

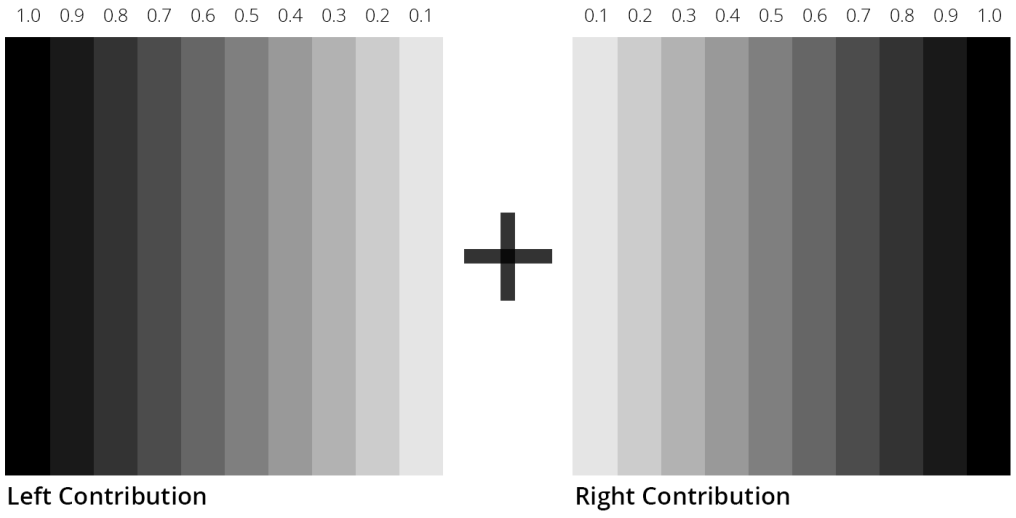


***Figure 2.5.*** *Gradient blending algorithm*

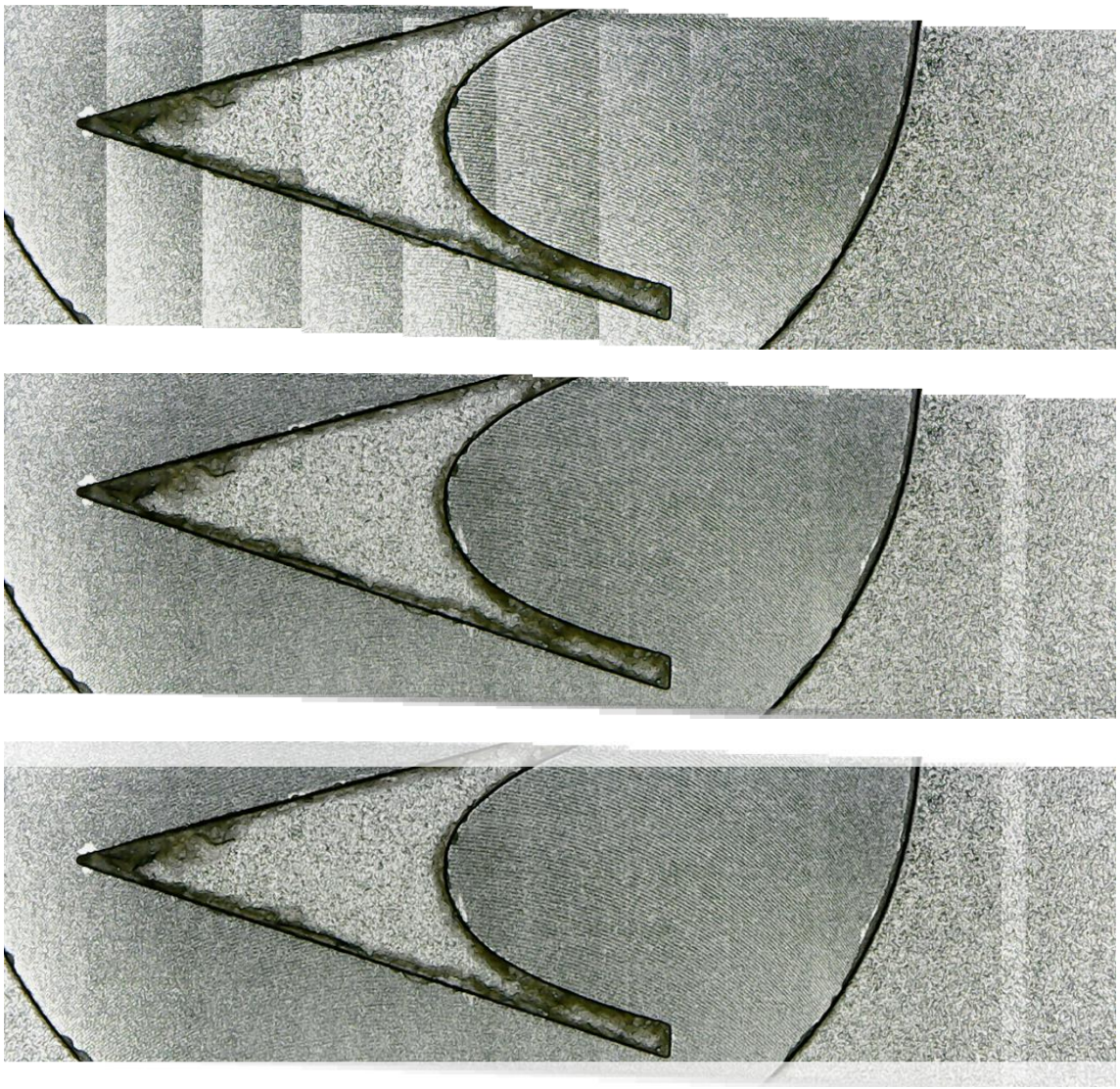

***Figure 2.6.*** *Blending and cropping results for a strip of 9 images of the backcase of a Motorola smartphone. From top to bottom: Stitched images without blending, stitched images after blending and resulting area after cropping.*

These results show how the blending algorithm effectively eliminates any visible edges, and it helps reducing the undesirable lighting effects introduced by the camera LEDs. They also help illustrate the way we have implemented image stitching for multiple images. New images are stitched one by one to the previous result horizontally, generating these horizontal strips. Then, the horizontal strips are stitched vertically, generating the global composite image.

## 2.4. 3D Reconstruction

We extract the depth information from the acquired images based on the epipolar geometry we discussed in the introduction of this report. As we noted there, we do not need to rectify our images for stereo matching as they have been taken with a single camera slided over a singular plane. That means we can assume the view planes for all of our images are contained within the same plane, and they all have been acquired by the same camera with the same focal length. Therefore, epipolar lines converge in the infinite and we only have to search along horizontal rows to find the disparity values.

OpenCV's stereo matcher class has been used to compute the depth maps for each pair of images. We use the vertical offset information obtained during the stitching process to align the image pairs that will be fed to the stereo matcher. The process the stereo matcher follows to compute the disparity values has been illustrated in Figure 2.7.

For each pixel in the left image, a block of a given *block size* is built around that pixel. Then, in the right image, the block from the left image is slided along a horizontal row until a best match is found. A search range is defined by the *number of disparities* parameter, where the search area will go from the position the block was in the left image to the left. The search only happens to the left of that position because it is virtually impossible that the actual point is to the right of that position, as the camera is more to the right and is assumed to have the same angle.

In order to reduce the search range, which reduces computation time and the probability of inaccurate disparity values, we feed the stereo matcher only with the overlapping areas of both images. This causes a problem, as the overlapping area is determined by the best match during the stitching process, which means it is based on the average background depth. Points above or below that average depth will not match in both images (which is the basis of 3D reconstruction), where points above the average background will move to the left on the right image, but points below the average background depth will move to the right on the right image. That is, points below the average background will be outside of the stereo matcher's search range, and the disparity values for those points will either be erroneous or not found.

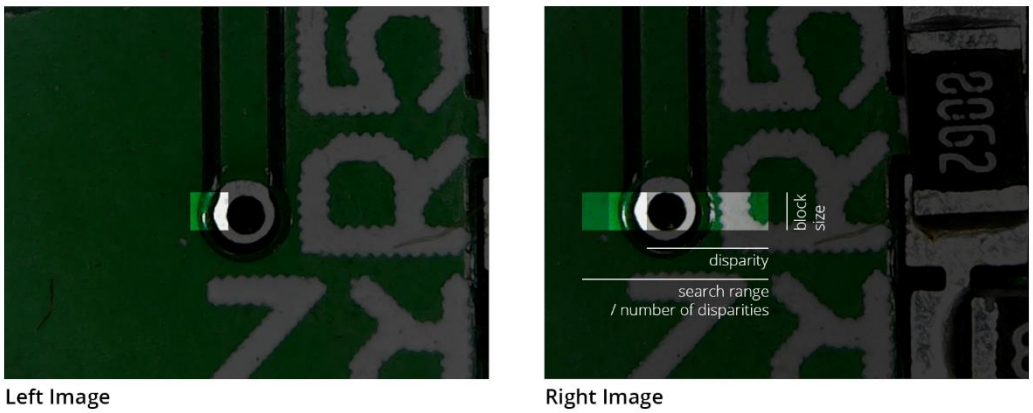


***Figure 2.7.*** *Stereo matching algorithm*

The solution to this problem is to add a margin to the cropped images we feed to the stereo matcher. This margin will have to be sufficient so that the points below the average background still appear within the stereo matcher's search range.

It is worth mentioning that we opted for a high baseline because that makes errors in the disparity values relatively smaller. Another approach we tried was a smaller baseline that increased the overlapping areas, but the performance for the high baseline was much better.

After we obtain the depth maps for every image pair, we combine them into a single depth map following a similar process to the one we used during the stitching phase. Again, we use the information from the stitching phase to determine the positions of the depth maps, and then we blend them together using a weighted sum.

Because we fed the stereo matcher with only the overlapping areas of the images, the resulting depth maps have a varying offset that can we compensate by comparing the mean values in the overlapping areas between depth maps.

Once we have the composite depth map, we can generate a point cloud of the scanned surface by adding the color information from the composite stitched image to the depth information of the depth map. We generate a *.ply* file where every line is a point in the point cloud, containing the X, Y, and Z coordinates as well as the R, G, and B color channels. These files can be visualized using any software that is compatible with point clouds, such as MeshLab.

The most accurate way to generate these point clouds involves reprojecting the 2D points to their actual X and Y coordinates, which can be computed knowing the depth or Z coordinate, the focal length, and the distance of the point to the optical center as shown in equations (4) and (5). The disparity values can be easily converted into actual metric units knowing the baseline length and the focal length as shown in equation (3).

$$disparity = \frac{B * f}{z} \quad (3)$$

$$x' = (x - x_0) * \frac{z}{f} \quad (4)$$

$$y' = (y - y_0) * \frac{z}{f} \quad (5)$$

*where:*

$B = \ baseline$

$f = \ focal\ length$

$x_0\ and\ y_0 = \ coordinates\ of\ the\ optical\ center$

# 3. **RESULTS**

The following pages show results obtained for different scanned surfaces and area sizes. A table with the computation times for two samples after different stages has been included below. The computing platform on which these results have been obtained is equipped with an Intel Core i7-7700HQ processor and an NVIDIA GeForce GTX 1050 graphic card.

| Number of Images | Stitched Image | + Depth Map | + Pointcloud | + Histogram Fixing |
|---|---|---|---|---|
| *20x20* | 15.78s | 52.20s | 59.57s | 62.87s |
| *9x9* | 4.10s | 11.34s | 13.88s | 17.95s |

***Table 2.1.*** *Computation times for 20x20 and 9x9 input images*

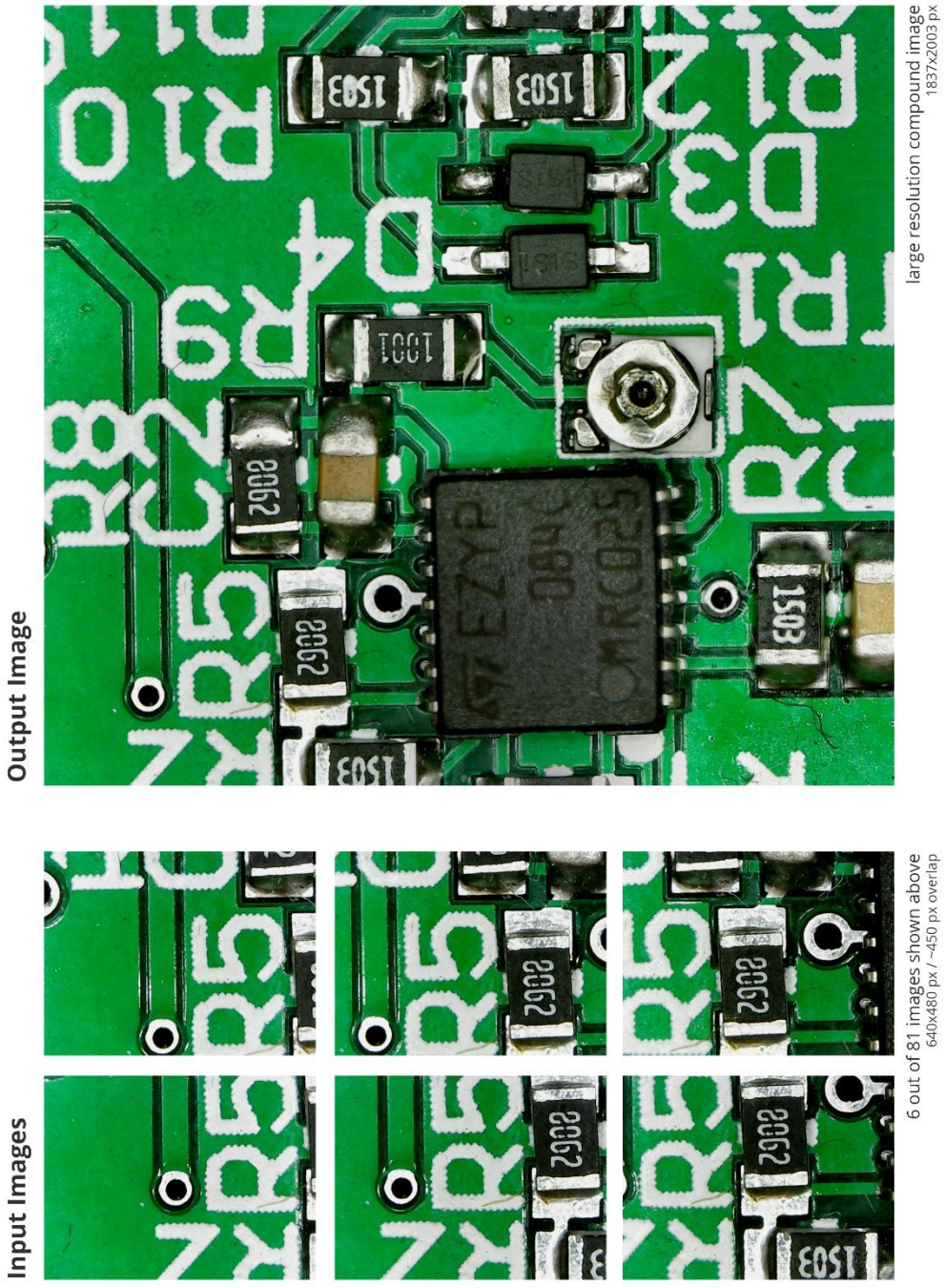


***Figure 3.1.*** *Image stitching results for a 9x9 images scan of an electronic board*

The image above shows the output composite image after feeding the stitching algorithm with a 9x9 grid of captured images, where 6 of the 81 images that were input to the software are shown in the bottom of the figure. Edges are unnoticeable thanks to the blending algorithm. The depth map and the point cloud in the next page show that the 3D reconstruction algorithm has been able to properly identify the height of the different electronic components.

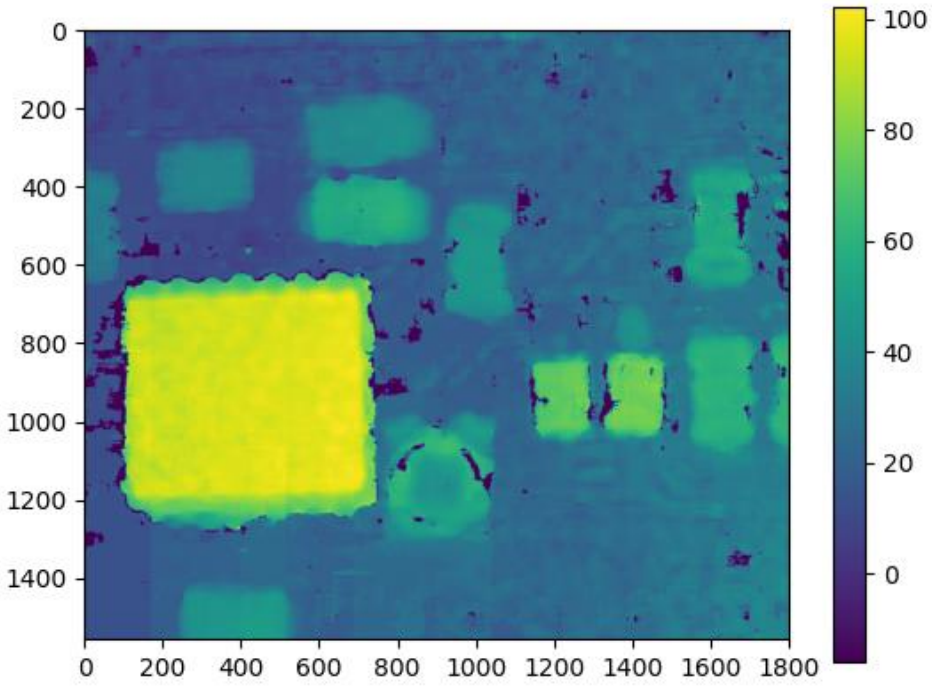


***Figure 3.2.*** *Depth map for a 9x9 images scan of an electronic board*

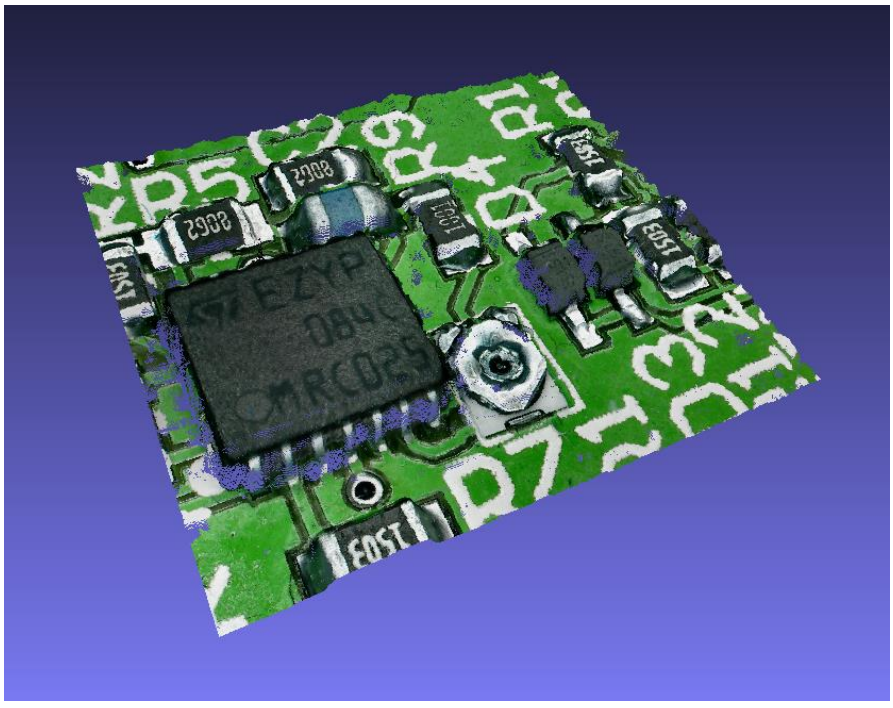

***Figure 3.3.*** *Generated point cloud for a 9x9 images scan of an electronic board*

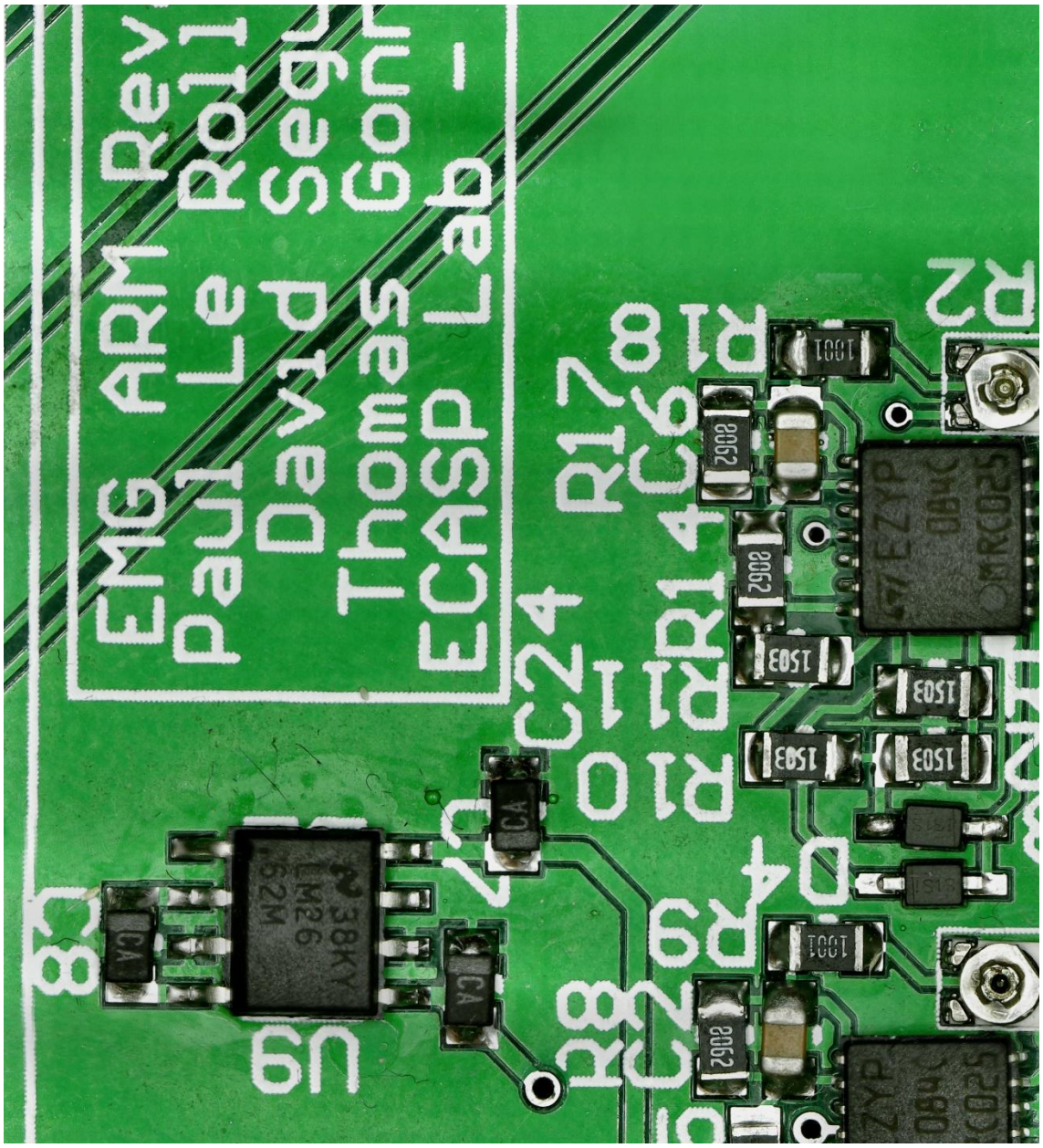


***Figure 3.4.*** *Image stitching results for a 20x20 images scan of an electronic board*

The image above shows the image stitching results for the same board, but this time for a 20x20 image grid given as an input. We can see in the top right area the problems the algorithm encounters when facing flat surfaces with little information. This problem was made worse by the fact that the input images for that area show different lighting effects depending on the illumination angle, providing false references. In the depth map and point cloud results shown in next page we can see how the stereo matcher does not return a value for this area, as the correlation coefficient for the best match is not above the confidence threshold.

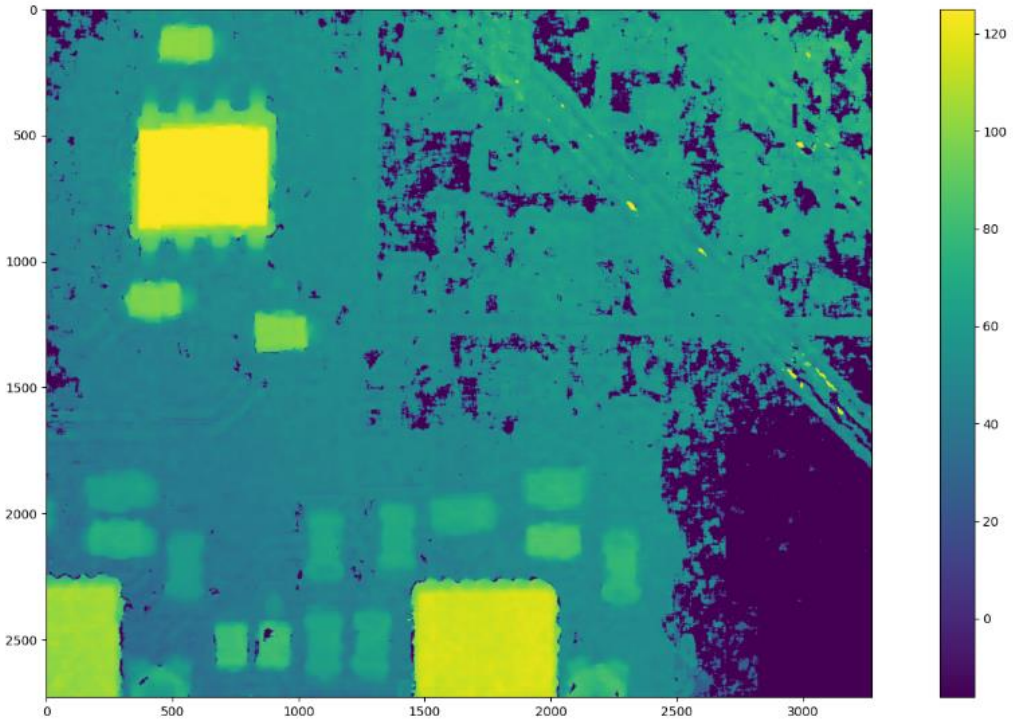


***Figure 3.5.*** *Depth map for a 20x20 images scan of an electronic board*

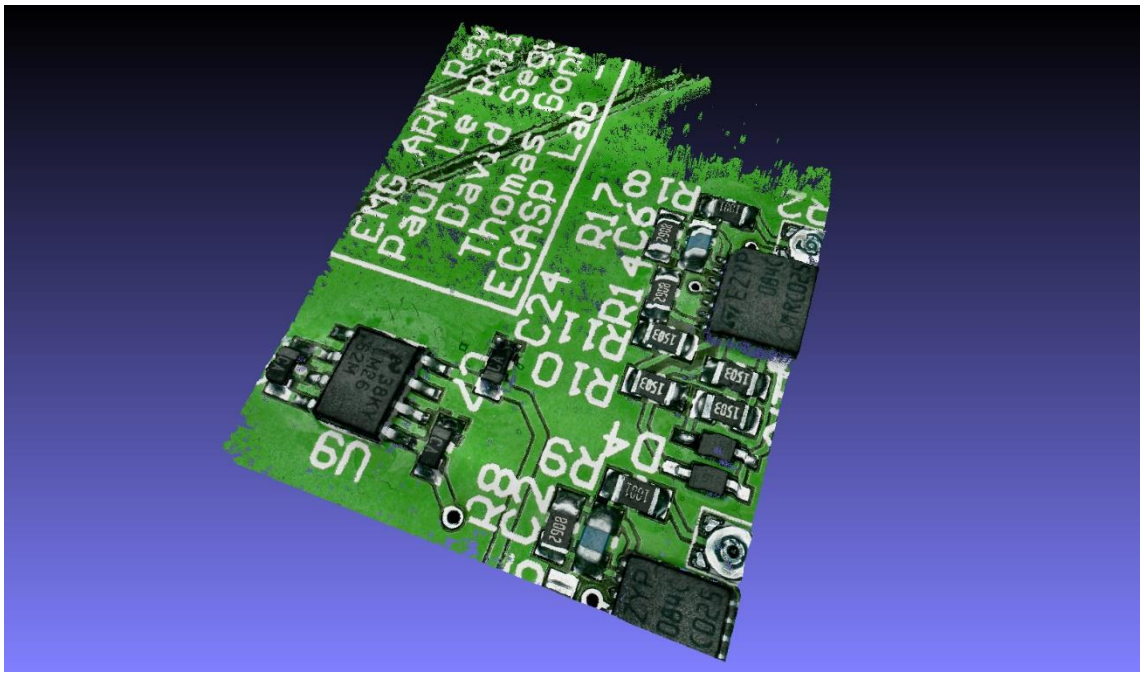


***Figure 3.6.*** *Point cloud for a 20x20 images scan of an electronic board*


***Figure 3.7.*** *Image stitching results for a 16x18 images scan of a coin*

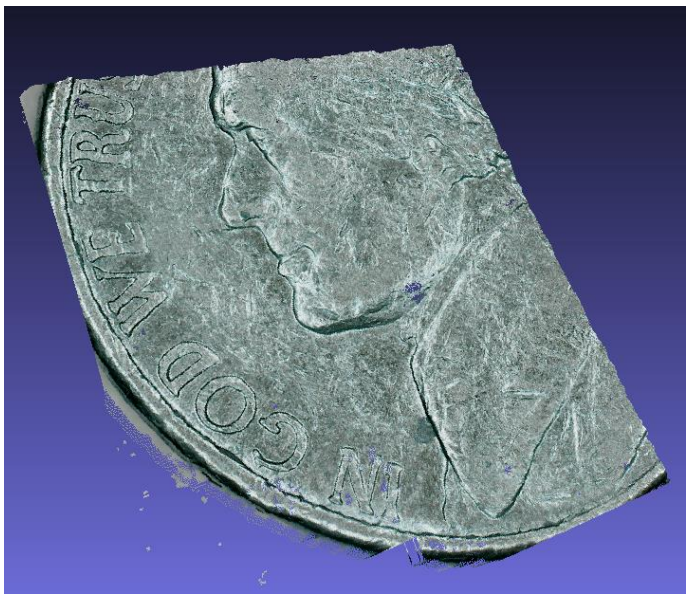

***Figure 3.8.*** *Point cloud for a 16x18 images scan of a coin*

# 4. **CONCLUSION**

The system shows promising results both in terms of quality and computation speed. The stitching algorithm produces high quality composite images with unnoticeable edges between the stitched images, although it has room for improvement when it comes to dealing with flat, featureless surfaces. Computation speed can also be improved by developing a smarter search method that follows gradients to find the highest cross-correlation coefficient, instead of computing the coefficient value for each position within the search range.

As for 3D reconstruction, results can be improved by applying post-processing techniques such as smoothing or hole-filling. Improvements can be achieved as well by generating more depth maps based on different image pairs other than those directly adjacent.

# REFERENCES

[1] Vladan Rankov, Rosalind J. Locke, Richard J. Edens, Paul R. Barber and Borivoj Vojnovic, *"An algorithm for image stitching and blending".*
March 2005
[2] OpenCV's Stitcher Class:
https://docs.opencv.org/trunk/d8/d19/tutorial_stitcher.html
[3] OpenCV's Template Matching:
https://docs.opencv.org/3.3.0/d4/dc6/tutorial_py_template_matching.html
[4] S. K. Chow, H. Hakozaki, D. L. Price, N. A. B. Maclean, and others, *"Automated microscopy system for mosaic acquisition and processing".*
2006
[5] F.B. Legesse, O. Chernavskaia, S. Heuke, T. Bocklitz, T. Meyer, J. Popp and R. Heintzmann, *"Seamless stitching of tile scan microscope images".*
2015
[6] Qiang Wu, Fatima A. Merchant and Kenneth R. Castleman, *"Microscope Image Processing".*
2008