2017

AN ALGORITHM FOR RECONSTRUCTING THREE-DIMENSIONAL IMAGES FROM OVERLAPPING TWO-DIMENSIONAL INTENSITY MEASUREMENTS WITH RELAXED CAMERA POSITIONING REQUIREMENTS, WITH APPLICATION TO ADDITIVE MANUFACTURING

Siranee Nuchitprasitchai
snuchitp@mtu.edu

AN ALGORITHM FOR RECONSTRUCTING THREE-DIMENSIONAL IMAGES FROM OVERLAPPING TWO-DIMENSIONAL INTENSITY MEASUREMENTS WITH RELAXED CAMERA POSITIONING REQUIREMENTS, WITH APPLICATION TO ADDITIVE MANUFACTURING

By

Siranee Nuchitprasitchai

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Computer Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2017

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Computer Engineering.

Department of Electrical and Computer Engineering

Dissertation Advisor:	*Dr. Michael C. Roggemann*
Committee Member:	*Dr. Timothy C. Havens*
Committee Member:	*Dr. Jeremy P. Bos*
Committee Member:	*Dr. Joshua M. Pearce*
Department Chair:	*Dr. Daniel R. Fuhrmann*

Contents

List of Figures

List of Tables

Preface

This dissertation contains published, submitted, or to be submitted journal articles by the author of this dissertation. The overarching goal of this work was to develop, test, and experimentally evaluate measurement and processing techniques for reconstructing three-dimensional object estimates from sets of two dimensional images using inexpensive cameras which could not be located in the geometry used in conventional stereo image reconstruction. The original interest was to apply this work in surveillance activities, but as the project advanced an opportunity to apply this work to monitoring the evolution of object fabrication inside three dimensional printers arose. Three-dimensional printing is an appropriate application for this work due to the fact that many sources of error exist in a three-dimensional printer including errors in locating the print head and the dimensional instability of the materials used. The physical arrangement of a three-dimensional printer prevents conventional stereo imaging camera placement, and hence the algorithms developed here provide an appropriate solution to this problem. Four journal articles were developed from this work. The first one is in print, and the remaining three are submitted and in review at the time of this writing. They are listed below:

Chapter 2: Siranee Nuchitprasitchai, Michael C. Roggemann, and Timothy C. Havens. An Algorithm for Reconstructing Three Dimensional Images from Overlapping Two-Dimensional Intensity Measurements with Relaxed Camera Positioning Requirements. IJMER, 6(9):69–81. Available online September 2016. S.N. wrote the algorithm, performed all experiments and analyzed the results. M.R. and T.H. formulated the project and assisted on the analysis. All authors co-wrote and edited the manuscript.

Chapter 3: Siranee Nuchitprasitchai, Michael C. Roggemann, and Joshua M. Pearce. Factors Effecting Real Time Optical Monitoring of Fused Filament 3-D Printing (to be published). S.N. wrote the algorithm, performed all experiments and analyzed the results. M.R. and J.P. formulated the project and assisted on the analysis. All authors co-wrote and edited the manuscript.

Chapter 4: Siranee Nuchitprasitchai, Michael C. Roggemann, and Joshua M. Pearce. An Open Source Algorithm for Reconstructing 2-D Images of 3-D Objects Being Fabricated for Low-cost, Reliable Real-Time Monitoring of FFF-based 3-D Printing (to be published). S.N. wrote the algorithm, performed all experiments and analyzed the results. M.R. and J.P. formulated the project and assisted on the analysis. All authors co-wrote and edited the manuscript.

Chapter 5: Siranee Nuchitprasitchai, Michael C. Roggemann, and Joshua M. Pearce. 360 Degree Real-time Monitoring of 3-D Printing Using Computer Vision Analysis of Two Camera Views (to be published). S.N. wrote the algorithm, performed all experiments and analyzed the results. M.R. and J.P. formulated the project and assisted on the analysis. All authors co-wrote and edited the manuscript.

Each of these papers is presented in a chapter in the body of this dissertation.

Acknowledgement

First of all, I would like to express my special thanks to my advisor, Dr. Michael C. Roggemann to give me the opportunity to be his advisee. My Ph.D. would not have been possible without my advisor. I really appreciate all his inspiration, support, patience, and guidance throughout the research. I would like to thank Dr. Joshua M. Pearce for his inspiration, support, and guidance throughout the research, and for serving as my committee member. I would like to thank Dr. Timothy C. Havens and Dr. Jeremy P. Bos for serving as my committee members to provide me the helpful suggestion in improving my research. I am grateful for my family and my husband for their support and love. He always be beside me through the storm or the sunny days together. I would like to thank all of my friends whom I met at Michigan Technological University for their love, friendship and all supports. They made my life more colorful while I studied here. I would also like to express my gratitude and appreciation to all the members of the Michigan Tech's Open Sustainability Technology Lab for their friendship and supports, Adam Pringle, John Laureto, Shan Zhong, Handy Chandra, and Shane Oberloier. I am also thankful for the financial support from Royal Thai Scholarship, EERC department at Michigan Technological University, and the John Wesley James Jones Memorial Scholarship. I am also grateful for Sripaipan family who has helped and supported me since I arrived in Houghton. I would not be here to peruse my Ph.D., if my dear sister, Sopit KhanKhangn, did not believe in me that God would help me to accomplish God's will for me in 2004. Finally, I would like to thank God, and my brothers and sisters in Jesus Christ here in USA, in Thailand, and around the world for their prayers and supports.

Abstract

Cameras are everywhere for security purposes and there are often many cameras installed close to each other to cover areas of interest, such as airport passenger terminals. These systems are often designed to have overlapping fields of view to provide different aspects of the scene to review when, for example, law enforcement issues arise. However, these cameras are rarely, if ever positioned in a way that would be conducive to conventional stereo image processing. To address this, issue an algorithm was developed to rectify images measured under such conditions, and then perform stereo image reconstruction. The initial experiments described here were set up using two scientific cameras to capture overlapping images in various cameras positons. The results showed that the algorithm was accurately reconstructing the three-dimensional (3-D) surface locations of the input objects.

During the research an opportunity arose to further develop and test the algorithms for the problem of monitoring the fabrication process inside a 3-D printer. The geometry of 3-D printers prevents the location of cameras in the conventional stereo imaging geometry, making the algorithms described above seem like an attractive solution to this problem. The emphasis in 3-D printing on using extremely low cost components and open source software, and the need to develop the means of comparing observed progress in the fabrication process to a model of the device being fabricated posed additional development challenges. Inside the 3-D printer the algorithm was applied using two scientific cameras to detect the errors during the printing of the low-cost open-source RepRap style 3-D printer developed by the Michigan Tech's Open Sustainability Technology Lab. An

algorithm to detect errors in the shape of a device being fabricated using only one camera was also developed. The results show that a 3-D reconstruction algorithm can be used to accurately detect the 3-D printing errors.

The initial development of the algorithm was in MATLAB. The cost of the MATLAB software might prevent it from being used by open-source communities. Thus, the algorithm was ported to Python and made open-source for everyone to use and customize. To reduce the cost, the commonly used and widely available inexpensive webcams were also used instead of the expensive scientific cameras. In order to detect errors around the printed part, six webcams were used, so there were 3 pairs of webcams and each pair were 120 degrees apart. The results indicated that the algorithms are precisely detect the 3-D printing errors around the printed part in shape and size aspects. With this low-cost and open-source approach, the algorithms are ready for wide range of use and applications.

Chapter 1: Introduction

1.1 Motivation

Three-dimensional (3-D) image reconstruction from sets of two-dimensional (2-D) images using the stereovision technique has been an area of active research for many decades, and has been applied in many fields, such as medical imaging [1-2], robot navigation [3-4], image analysis [5-6], machine vision [7], and architecture [8-9]. In most cases, the geometries of the stereo cameras and the scene are carefully controlled to make the processing straightforward. In the most common configuration, the two cameras and the target are arranged in a simplified epipolar geometry. In this case the camera positions are arranged so that horizontal lines in the two camera images result from the same points in the scene viewed from different perspectives [10-16]. When this is the case, the disparities needed to compute a 3-D image can be obtained from block matching applied in a horizontal line search manner. In the simplified epipolar geometry the spatial scale of the two images is guaranteed to be the same, and the stereo reconstruction problem can be reduced to finding the disparities between corresponding points in the two images. The corresponding points can be found with a manual, human in the loop approach [17-18], automated block-matching algorithms [19-21], gradient-based optimization [22-23], feature matching [24-27], dynamic programming [28-31], graph cuts [32-35], or belief propagation [36]. These techniques have been successfully demonstrated, and are commercially available products [37-38].

The work presented here addresses the more general problem of stereo reconstruction when the camera-target geometry cannot be completely controlled. In this case the cameras are

not arranged in the conventional, simplified epipolar stereo imaging geometry. An example of this situation is crowd surveillance in an airport, where the cameras would have overlapping fields of view, but might not have matching physical parameters. As a result, the image scales may be different, and the area where the images overlap may be uncontrolled and irregular. The geometry is illustrated in Figure 1.1. In this case the two cameras have x and z-displacements from the simplified epipolar geometry, and the same y-displacement; therefore, the images measured from identical cameras would be shifted and scaled differently, violating the condition of the simplified epipolar geometry, since for parallel cameras in the simplified epipolar geometry the image of any point must lie on the same horizontal line in each image. When the cameras are not in the simplified epipolar geometry the images need to be rectified. Rectifying the images in stereovision is the step of transforming the measured images to lie in a common plane with a common spatial sampling. If two overlapping images of the same scene from a pair of cameras are aligned correctly, a conventional stereovision algorithm can be used to reconstruct the desired 3-D map of the surfaces in the scene. We propose a new approach for stereovision in this situation. Our approach uses the Scale Invariant Feature Transform (SIFT) [39-40] to find matching points between a pair of images, and using the RANdom SAmple Consensus (RANSAC) [41] to eliminate the wrong matching points. This information then is used to rescale and rectify the images. Next, a block-matching algorithm [42] is used to find the corresponding points in the left and in the right images of a stereo pair. Finally, the 3-D surface location is found by using a set of equations generalized for the general epipolar geometry for the corresponding points.

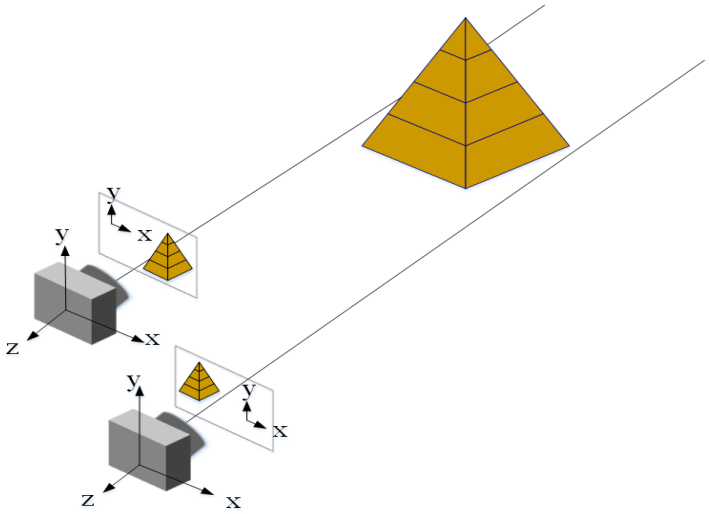


Figure 1.1 Set up of two cameras for stereovision used in this study

The main purpose of the rest of this chapter is to establish background information which is used throughout this dissertation. The remainder of this chapter is organized as follows: theory regarding the camera model, stereo reconstruction in the simplified epipolar geometry, Scale Invariant Feature Transform (SIFT), RANdom SAmple Consensus, approach, summary of key results, and organization.

1.2 Camera Model

We now review the basic camera model for stereo reconstruction. To reconstruct the 3-D images from sets of 2-D images knowledge of the camera parameters is required. The single thin lens camera is the simplest camera model that describes the mathematical relationship between the 3-D object points and the image points. In the thin lens model, the rays of light emitted from a point travel along paths through the lens, converging at a point behind the lens. In geometric optics, a ray passing through the center of a thin lens is called the chief ray, which is not deflected by the lens. The image is inverted in the image plane.

Figure 1.2 shows a chief ray in the thin lens camera model, except that the image plane is moved to the front of the lens instead of behind it, and in this case the image is not inverted. The perspective model explains the projection of an object point at location P to the point P′ where it is imaged as defined by a chief ray traced from P to P′ through the center of the lens.

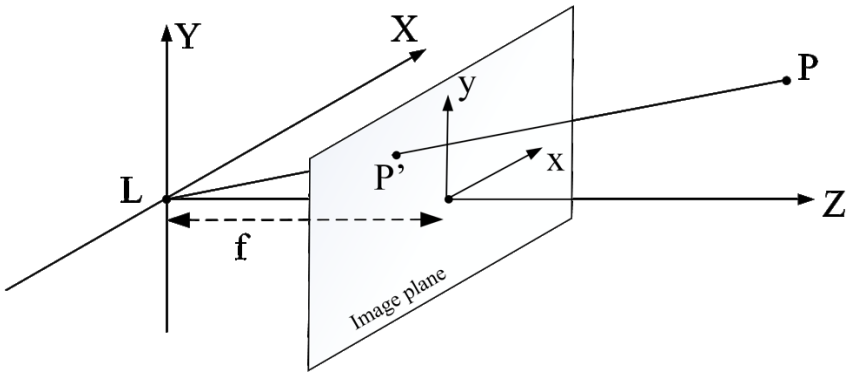


Figure 1.2 The equivalent of single thin lens camera geometry

Inspection of Figure 1.2 shows that when a ray passes through the center of the aperture located at $(0,0,0)$ and the notional image plane is located a distance f in front of the lens, the projection of the object point P at (X, Y, Z) in an object space onto the image plane point P' located at (x, y) in the image plane. The image location (x, y) is related to the object location (X, Y, Z) by

$$x = fX/Z \tag{1-1}$$

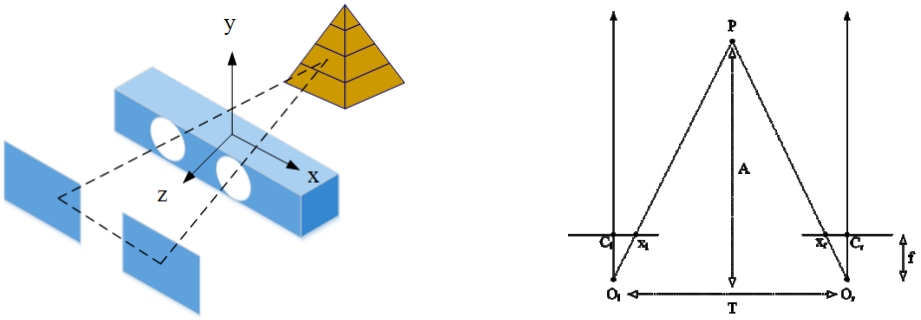
$$y = fY/Z \tag{1-2}$$

Equations (1-1) and (1-2) show that the object location (X, Y, Z) cannot be retrieved from the information in a single image, since there are three unknown variables (X, Y, Z) with

only two measurements. In this paper, Equations (1-1) and (1-2) are applied in stereovision in order to calculate the object location points when the rays from each pair of corresponding points intersect at a common 3-D scene point. Finding the corresponding points in stereo image pairs is discussed next.

1.3 Stereo Reconstruction in the Simplified Epipolar Geometry

We now review the geometrical arrangement of stereo images measured in the simplified epipolar geometry. Figure 1.3 (a) shows a typical set up for a pair of stereo cameras where the cameras have only horizontal shift. Figure 1.3 (b) shows how the location of the image of the same point appears in both images, where c_l and c_r are the image centers for the left and the right images respectively, O_l and O_r are the camera centers for the left and the right cameras respectively, x_l and x_r are the image locations of P in the left and the right images respectively, T is the horizontal distance between the two cameras, f is the focal length, and A is the distance between the object point and the cameras.



(a) Stereo camera geometry (b) Triangular geometry

Figure 1.3 Stereovision in the epipolar geometry

The disparity value d is the difference in the position of two points between the left and the right image planes where the right image is the reference image, which is

$$d = x_r - x_l \tag{1-3}$$

Each corresponding points in a pair of images are minimum disparity value over the search region. After finding all corresponding points, the disparity map is created. The disparity map is an image where every pixel contains the disparity value of each corresponding points. From similar triangles in Figure 1.3 (b), the depth information Z for an object point is calculated by

$$Z = (T * f)/d \tag{1-4}$$

Equations (1-1) and (1-2) are used to calculated the X, Y information for the object point given by

$$X = (x_l * Z)/f \tag{1-5}$$

$$Y = (y_l * Z)/f \tag{1-6}$$

where T is the horizontal distance between the two cameras, f is the focal length, (x_l, y_l) is the image locations of P in the left image, and d is the disparity value.

Finding the corresponding points in the pairs of images is the key to successful stereovision reconstruction. We assume that the stereo images are rectified, which means that the corresponding lines are horizontal and on the same height in the left and the right images. Block-matching is one of the techniques to find the corresponding points between the left and the right images. A widely used block-matching algorithm is the Sum of Absolute Difference (SAD) [42]. In equation (1-7), the SAD is calculated by taking the absolute difference between each pixel in a square block of certain size around the interested pixel

in the right image (reference image) and finding the corresponding pixel within the square block in the left image, while moving along the corresponding scan line or the search region. For each pixel in the right image, there should be only one best pair of corresponding points between the left and the right images when the value of the SAD is minimum over the search region. The calculation is repeated for each pixel in the right image until all corresponding pixels in the left image are found.

$$\underset{x_l,y_l \in SR}{\operatorname{argmin}} \mathrm{SAD}(x_l, y_l, x_r, y_r) = \sum_{j=\left(-\frac{B-1}{2}\right)}^{\left(\frac{B-1}{2}\right)} \sum_{i=\left(-\frac{B-1}{2}\right)}^{\left(\frac{B-1}{2}\right)} |I_l(x_l + i, y_l + j) - I_r(x_r + i, y_r + j)| \quad (1\text{-}7)$$

where B is the block size, SR is the search region, (x_l, y_l) is the candidate corresponding pixel in the left image, (x_r, y_r) is the interested pixel in the right image, I_l and I_r are the pixel intensities in the left and the right images, respectively.

1.4 Scale Invariant Feature Transform (SIFT)

We now describe the SIFT [40] algorithm and its application here to the problem of rectifying images when the cameras are not in the simplified epipolar geometry. Developing methods for working with cameras in more general geometries will significantly expand the physical camera arrangements from which three-dimensional information can be extracted. The use of a generalized camera geometry complicates the correspondence problem by requiring one of the images to be rescaled and to be rectified. In this section, we discuss the use of the SIFT algorithm to address these problems. Here, the SIFT algorithm is applied to find the matching points between the stereo images when a pair of stereo images have different scales. The matching point information is used to rescale one of the images. After the pair of stereo images has the same scale, the SIFT

algorithm is reused to find matching points between the stereo images. The new matching point information is used to rectify the images, and then the SAD algorithm is used to compute the disparities. The SIFT algorithm consists of four steps: scale-space extrema detection, key point localization, orientation assignment and key point descriptor.

The first step of the SIFT algorithm is scale-space extrema detection. To create the scale-space of an image, the Difference of Gaussian (DoG) images is computed as an approximation of scale invariant of the Laplacian of Gaussian from the difference of two nearby scales separated by a constant number k. The DoG images are given by

$$D_{(l,n)}(x,y,\sigma) = \left(G\left(x,y,k^{(n)}\sigma\right) - G\left(x,y,k^{(n-1)}\sigma\right)\right) * I(x,y) \tag{1-8}$$

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\left(x^2+y^2\right)/2\sigma^2} \tag{1-9}$$

where $D_{(l,n)}(x,y,\sigma)$ is DoG, k is a constant multiplicative factor, l is the level number, n is the scale space image range $[1, s+2]$ which $s = \log_k 2$, σ is the scale parameter, $G(x,y,\sigma)$ is a variable-scale Gaussian kernel, $I(x,y)$ is a grey-level input image, and * is the convolution operation.

After the DoG images are calculated, all locations and scales are processed to find key point candidates. The key point candidate is a pixel where it is the greatest or least of all neighboring points. In 3-by-3 sub-regions, a key point candidate is found by comparing its to eight neighbors in the current DoG image, and nine neighbors in the above and the below DoG images.

The next step of the SIFT algorithm is key point localization. Not all key point candidates are useful as features because some of them have low contrast, or lie along an edge. Thus, some key point candidates are rejected to increase the efficiency and robustness of the algorithm by using a Taylor expansion. The Taylor expansion for the DoG image is given by

$$S(X) = D + \frac{\partial D^T}{\partial X} X + \frac{1}{2} X^T \frac{\partial^2 D^T}{\partial a^2} X \tag{1-10}$$

where $S(X)$ is the Taylor expansion for the DoG image, D is the DoG, and $X = (x, y, \sigma)^T$ is the current key point candidate.

The location of extremum is calculated by taking the derivative of Equation (1-10) with respect to X and setting it to zero, giving

$$\hat{X} = -\left(\frac{\partial T}{\partial X}\right)\left(\frac{\partial^2 T}{\partial X^2}\right)^{-1} \tag{1-11}$$

where $\hat{X}$ is the location of extremum. If $\hat{X}$> 0.5 in any dimension, then it means that the extremum lies closer to a different key point.

The low contrast key points are rejected when $|S(\hat{X})|$ < 0.03. The low contrast key point is calculated by substituting equation (1-11) into (1-10), giving

$$S(\hat{X}) = D + \frac{1}{2}\frac{\partial D^T}{\partial X}\hat{X} \tag{1-12}$$

To increase stability, key point candidates that lie along an edge need to be rejected. If Ratio $> (r+1)^2/r$ when the SIFT uses r =10 then the key point candidate is deleted because it lies along the edge. Ratio is calculated by

$$\text{Ratio} = \frac{\text{Tr(H)}^2}{\text{Det(H)}} \tag{1-13}$$

where Tr(H) is the Trace of Hessian Matrix and Det(H) is the Determinant of Hessian Matrix. They are given by

$$\text{Tr(H)} = D_{xx} + D_{yy} \tag{1-14}$$

$$\text{Det(H)} = D_{xx} D_{yy} - (D_{xy})^2 \tag{1-15}$$

where D is the DoG and H is the Hessian Matrix in which second order partial of derivatives are estimated by taking differences of neighboring sample points (i.e. D_{xx} is second order partial of derivatives of x), which it is given by

$$\text{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{1-16}$$

The following step of the SIFT algorithm is orientation assignment by using accurate key points which have been tested to be scale invariance from the previous step. Each key point is assigned both gradient magnitudes and gradient orientations from the Gaussian blurred image to provide one or more orientations' invariance. The gradient orientations of the neighborhood pixels are then accumulated together in a histogram bar, which is divided into a 36-histogram bar. The gradient magnitude and the gradient orientation are calculated by

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y)^2 + (L(x,y+1) - L(x,y-1)^2} \tag{1-17}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{(L(x,(y+1) - (L(x,y-1))}{(L(x+1,(y) - (L(x-1,y))}\right) \tag{1-18}$$

where $m(x, y)$ is the gradient magnitude, $\theta(x, y)$ is the gradient orientation which ranges from 0 to 360 degrees, and $L(x, y, k\sigma)$ is the Gaussian-smoothed image blurred given by

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \tag{1-19}$$

$$G(x, y, k\sigma) = \frac{1}{2\pi k\sigma^2} e^{-(x^2+y^2)/2k\sigma^2} \tag{1-20}$$

where $G(x, y, \sigma)$ is a variable-scale Gaussian kernel, σ is the scale parameter, k is a constant multiplicative factor, $I(x, y)$ is a grey-level input image, and * is the convolution operation.

The final step of the SIFT algorithm is a key point descriptor. After the orientation assignment, each key point includes details about an image location, scale, and orientation, which are described by 16-by-16 windows. To identify each key point correctly, a unique key point descriptor needs to be created. When comparing two different images, key point descriptors are never exactly the same. In order to create the key point descriptors, 16-by-16 window of each key point are divided into sixteen 4-by-4 windows. For each 4-by-4 window, there are 16 orientation samples, which result in an 8-histogram bar and each bar has a range of 45 degrees. Therefore, from 16 orientation samples with 4-by-4 windows, dimensional vectors are created and are equal to 128. This feature vector is a unique key point descriptor.

1.5 RANdom SAmple Consensus (RANSAC)

To improve reconstructing a 3-D image, RANSAC [41] is applied to eliminate incorrect matching points after using SIFT. RANSAC is a learning technique to estimate the parameters of a mathematical model from SIFT's matching points data. First of all, we

need to find a mathematical model that fits to the set of hypothetical correct matching points. While estimating the parameters of a mathematical model, the outlier data is found when data does not fit the model because of an extreme value of the noise, erroneous measurement, or incorrect hypotheses about the interpretation of the data. The algorithm is an iterative method that consists of two steps. The first step is randomly selected a data subset from the original data. A fitting model and the corresponding model parameters are computed using only the selected data set. Secondly, all other data except selected data set are tested. If it does not fit the fitting model, a data element is considered as an outlier. The algorithm repeats these two steps until the algorithm reaches the maximum number of iterations allowed. Finally, the model that has the largest inliers would be used to eliminate the outlier.

1.6 Approach

In this dissertation, I develop triangulation-based geometric equations for reconstructing 3-D images. The triangular geometry based on the relaxed camera position is shown in Figure 1.4. The experiment is set up to test the algorithm by using two cameras. First, the images of the scenes with and without the target object are be acquired with the left and the right cameras respectively to allow the background to be removed, allowing us to do detailed engineering analysis of the output later in the paper. Our objective is to represent only the target object with the 3-D image reconstruction for demonstration process, and to assess performance. The SIFT and RANSAC algorithm is then used to find a set of matching points in order to rescale and to rectify the images. Next, a set of 3-D point positions in an object space is calculated for each pixel between two rectified stereo images by using a block-matching algorithm and a derived set of equations for the geometry.

Finally, the 3-D images are reconstructed from the set of 3-D point positions.

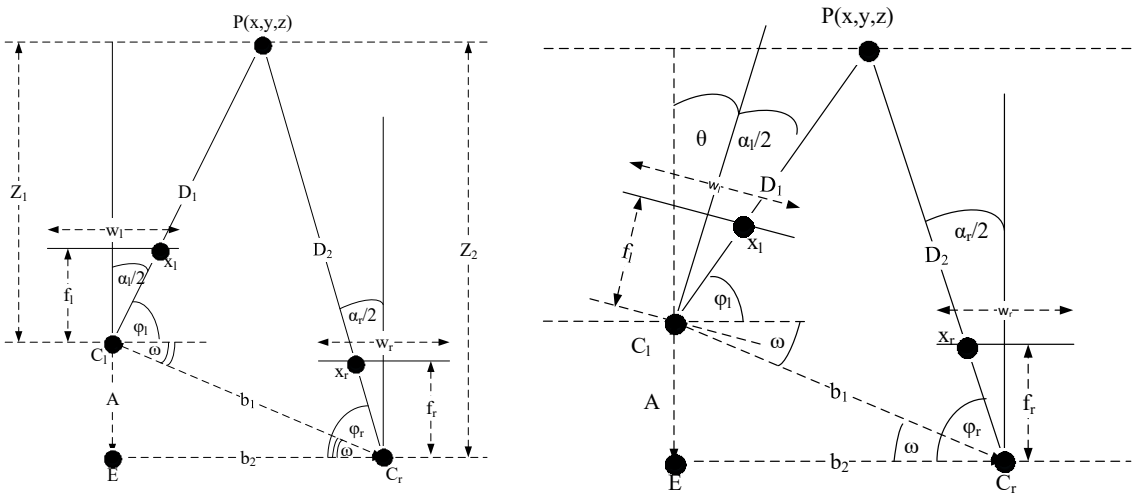


Figure 1.4 Triangular geometry

(a) both optical axis of cameras is parallel

(b) the optical axis of the left camera is rotated around the y-axis

1.7 Summary of Key Results

In this dissertation, the triangulation-based geometric algorithm was developed for 3-D reconstruction from 2-D overlapping images with two relaxed scientific camera positioning requirements. This algorithm exploited the SIFT and RANSAC techniques to rescale and rectify the stereo images, and the SAD block matching was applied to find the corresponding points between the left and the right images which were inputs for the triangulation-based geometric algorithm. This algorithm was tested with four different experiments: both cameras were parallel in the z-axis and the left camera was moved forward, both cameras were parallel in the z-axis and the left camera was moved backward, the left camera was rotated clockwise around the y-axis and was moved forward, and the

left camera was rotated clockwise around the y-axis and was moved backward. The 3-D reconstruction results show that the algorithm could be used to extract the 3-D information with high accuracy with RMSE = 1.265.

An opportunity arose in the monitoring of the failed 3-D printing in self-replicating rapid prototype (RepRap) 3-D printers research area, and this algorithm was improved and tested with the low-cost open-source RepRap 3-D printer developed by the Michigan Tech's Open Sustainability Technology Lab. To improve reliability of error detection, the algorithm to detect the shape error had been added to the approach. The results showed that these algorithms can detect failed printing close to 100%.

To make the approach a low-cost and open source reliable monitoring, the code is converted from MATLAB to Python, and it was tested with three pair of webcams setup around the printed part with 120 degrees apart. The quality of this approach using in experiments showed that the system was capable of a 100% rate for failure and error detection with 3X faster computation time for the shape technique comparing with code written in MATLAB.

1.8 Organization

The remainder of this dissertation is comprised of content from the first journal articles published by the editors of International Journal of Modern Engineering Research (IJMER) and the other articles have been completed and will be published. Chapter 2 is derived from "An Algorithm for Reconstructing Three Dimensional Images from Overlapping Two-Dimensional Intensity Measurements with Relaxed Camera Positioning Requirements" which was published online September, 2016, in International Journal of Modern

Engineering Research (IJMER). The paper provides the algorithm to reconstruct 3-D images for relaxed camera positions in MATLAB then applied for detecting an error in 3-D printing describe in chapter 3. The content in chapter 3 will be published under the title "Factors Effecting Real Time Optical Monitoring of Fused Filament 3-D Printing." To make this faster and open for everyone, the algorithm and the shape algorithm is ported to Python. The shape technique for single camera setup can be found in chapter 4. The content in chapter 4 will be published under the title "An Open Source Algorithm for Reconstructing 2-D Images of 3-D object being Fabricated for Low-cost, Reliable Real-Time Monitoring of FFF-based 3-D Printing." In chapter 5, the 3-D reconstruction technique for double camera setup is described and will be published under the title "360 Degree Real-time Monitoring of 3-D Printing Using Computer Analysis of Two Camera Views."

1.9 References

1. Stoyanov, Danail, et al. "Real-time stereo reconstruction in robotically assisted minimally invasive surgery." Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010. Springer Berlin Heidelberg, 2010. 275-282.
2. B. Andre, J. Dansereau, H. Labelle, "Optimized vertical stereo base radiographic setup for the clinical three-dimensional reconstruction of the human spine", J Biomech, 27 (8) (1994), pp. 1023–1035
3. Don Murray and James J. Little. "Using Real-Time Stereovision for Mobile Robot Navigation.", Autonomous Robots 8, (2000), pp.161–171.
4. Grosso, Enrico, Giulio Sandini, and Massimo Tistarelli. "3D object reconstruction using stereo and motion." Systems, Man and Cybernetics, IEEE Transactions on 19.6 (1989): 1465-1476.
5. Cardenas-Garcia, J. F., H. G. Yao, and S. Zheng. "3D reconstruction of objects using stereo imaging." Optics and Lasers in Engineering 22.3 (1995): 193-213.
6. Kim, Hansung, Seung-jun Yang, and Kwanghoon Sohn. "3D reconstruction of stereo images for interaction between real and virtual worlds." Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on. IEEE, 2003.

7. Suhr, Jae Kyu, et al. "Automatic free parking space detection by using motion stereo-based 3D reconstruction." Machine Vision and Applications 21.2 (2010): 163-176.
8. Baillard, Caroline, et al. "Automatic line matching and 3D reconstruction of buildings from multiple views." ISPRS Conference on Automatic Extraction of GIS Objects from Digital Imagery. Vol. 32. 1999.
9. Pollefeys, Marc, et al. "Detailed real-time urban 3d reconstruction from video. "International Journal of Computer Vision 78.2-3 (2008): 143-167.
10. S. I. Olsen. Epipolar line estimation for binocular stereovision. Proc. of the Nordic Summer School on Active Vision and geometric modellings Science. Aalborg : Aalborg Universitetsforlag, 1992. p. 143-149.
11. S. I. Olsen. Epipolar line estimation. Proc. 2 European Conference on Computer Vision, Lecture Notes in Computers Science. Berlin, Tyskland : Springer, 1992. pp. 307-311.
12. E. Nishimura, G. Xu, and S. Tsuji. Motion segmentation and correspondence using epipolar constraint. In Proc. 1st Asian Conf. Computer Vision, Osaka, Japan, 1993, pp. 199-204.
13. G. Xu, E. Nishimura, and S. Tsuji. Image correspondence and segmentation by epipolar lines: Theory, algorithm and applications. Technical report, Dept. of Systems Engineering, Osaka University, Japan, July 1993.
14. D.V. Papadimitriou and T.J. Dennis, "Epipolar line estimation and rectification for stereo image pairs", in Proceedings of the International Workshop on Stereoscopic and Three Dimensional Imaging (IWS3DI), Santorini, Greece, pp. 128-133, 1995
15. Z. Zhang and T.Kanade, “Determining the Epipolar Geometry and its Uncertainty: A Review”, IJCV, 27(2), 1998, pp.161-195.
16. O. Faugeras and Q.-T. Luong, “The Geometry of Multiple Images”, MIT Press: Cambridge, MA.
17. K. LU, X. Wang, Z. Wang and L. Wang, “Binocular stereovision based on opencv”, ICSSC, 2011, pp.74-77.
18. T. Surgailis, A. Valinevicius, V. Markevicius, D. Navikas and D. Andriukaitis, “Avoiding forward car collision using stereovision system”, ELEKTRONIKA IR ELEKTROTECHNIKA, ISSN 1392-1215, VOL. 18, NO. 8, 2012, pp. 37-40.
19. J. Vanne, E Aho, T D Hamalainen and K Kuusilinna, “A High-Performance Sum of Absolute Difference Implementation for Motion Estimation”, IEEE TCSVT, v. 16, n. 7, Jul. 2006, pp. 876-883.
20. O. Faugeras, B. Hotz, H. Matthieu, T. Vieville, Z. Zhang, P. Fua, E. Theron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy, “Real Time Correlation-Based Stereo: Algorithm, Implementations and Applications,” INRIA Technical Report 2013, 1993.
21. D.N. Bhat and S.K. Nayar, “Ordinal Measures for Image Correspondence,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, pp. 415-423, 1998.
22. B.D. Lucas and T. Kanade, “An Iterative Image Registration Technique with an Application to Stereovision,” Proc. Int’l Joint Conf. Artificial Intelligence, pp. 674-679, 1981.
23. V.S. Kluth, G.W. Kunkel, and U.A. Rauhala, “Global Least Squares Matching,” Proc. Int’l Geoscience and Remote Sensing Symp., vol. 2, pp. 1615-1618, 1992.

24. S. Randriamasy and A. Gagalowicz, “Region Based Stereo Matching Oriented Image Processing,” Proc. Computer Vision and Pattern Recognition, pp. 736-737, 1991.
25. V. Venkateswar and R. Chellappa, “Hierarchical Stereo and Motion Correspondence Using Feature Groupings,” Int’l J. Computer Vision, vol. 15, pp. 245-269, 1995.
26. F. Bigone, O. Henricsson, P. Fua, and M. Stricker, “Automatic Extraction of Generic House Roofs from High Resolution Aerial Imagery,” Proc. European Conf. Computer Vision, pp. 85-96, 1996.
27. S. Birchfield and C. Tomasi, “Depth Discontinuities by Pixel-to- Pixel Stereo,” Proc. IEEE Int’l Conf. Computer Vision, pp. 1073-1080, 1998.
28. Y. Ohta and T. Kanade, “Stereo by Intra- and Intra-Scanline Search Using Dynamic Programming,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 7 pp. 139-154, 1985.
29. S.S. Intille and A.F. Bobick, “Incorporating Intensity Edges in the Recovery of Occlusion Regions,” Proc. Int’l Conf. Pattern Recogni- tion, vol. 1, pp. 674-677, 1994.
30. I.J. Cox, S.L. Hingorani, S.B. Rao, and B.M. Maggs, “A Maximum Likelihood Stereo Algorithm,” Computer Vision and Image Under- standing, vol. 63, pp. 542-567, 1996.
31. C. S. Park, H. W. Park, A robust stereo disparity estimation using adaptive window search and dynamic programming search, Pattern Recognition (2000).
32. H. Zhao, “Global Optimal Surface from Stereo,” Proc. Int’l Conf. Pattern Recognition, vol. 1, pp. 101-104, 2000.
33. Y. Boykov and V. Kolmogorov, “An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision,” Proc. Third Int’l Workshop Energy Minimization Methods in Computer Vision and Pattern Recognition, 2001.
34. Y. Boykov, O. Veksler, and R. Zabih, “Fast Approximate Energy Minimization via Graph Cuts,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 11, pp. 1222-1239, Nov. 2001.
35. V. Kolmogorov and R. Zabih, “Computing Visual Correspon- dence with Occlusions Using Graph Cuts,” Proc. Int’l Conf. Computer Vision, 2001.
36. J. Sun, H.-Y. Shum, and N.-N. Zheng, “Stereo Matching Using Belief Propagation,” Proc. European Conf. Computer Vision, pp. 510-524, 2002.
37. http://www.thefoundry.co.uk/products/ocula/features/. Accessed 2015, September 10
38. http://www.ptgrey.com/support/downloads/documents/TAN2008005_Stereo_Vision_Introduction_and_Applications.pdf. Accessed 2015, September 10
39. D. Lowe, Distinctive image features from scale-invariant key points, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.
40. D. Lowe, Object recognition from local scale-invariant features, International Conference on Computer Vision, Corfu, Greece (September 1999), pp. 1150-1157.
41. Fischler, M.A. and R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24, 6 (1981), p. 381-395.

42. T. Tao, J. C. Koo, H. R. Choi, “A fast block matching algorithm for stereo correspondence,” in Proc. IEEE Int. Cyber. Intell. Syst., Sep. 2008, pp. 38–41.

Chapter 2: An Algorithm for Reconstructing Three Dimensional Images from Overlapping Two-Dimensional Intensity Measurements with Relaxed Camera Positioning Requirements[1]

2.1 Abstract

This paper proposes and demonstrates an algorithm to generate three-dimensional (3-D) reconstructions using images from a stereo vision of two-dimensional (2-D) surveillance camera without calibration. In the surveillance of public environment, the cameras are not set up for a binocular stereo system for a 3-D reconstruction, but here they can be used when there is an overlapped scene. When the field of view of multiple cameras overlap, the potential exists for computing the 3-D location of surfaces in the overlapping regions of the images. In this paper, we apply the Scale Invariant Feature Transform (SIFT), the RANdom SAmple Consensus (RANSAC), and the Sum of Absolute Differences (SAD) to reconstruct 3-D image from two overlapping images. The camera parameters and the geometry of the cameras are known; however, they do not correspond to conventional stereo image measurements. The process consists of two steps: image preparation and 3-D reconstruction. Image preparation involves rescaling, rectifying, and finding the corresponding points between the left and the right stereo images. The SIFT and the RANSAC algorithm are applied to find the difference of object size between the images and then to rescale and rectify the images. The corresponding points on the two images are

[1] This chapter has been published as an article in *International Journal of Modern Engineering Research (IJMER)*. Citation: Nuchitprasitchai S, Roggemann M, & Havens T (2017). An Algorithm for Reconstructing Three Dimensional Images from Overlapping Two Dimensional Intensity Measurements with Relaxed Camera Positioning Requirements. *IJMER*, 6(9):69–81. Available online September 2016 http://www.ijmer.com/papers/Vol6_Issue9/Version-2/J9226981.pdf.

found with a block matching method using the SAD technique. For 3-D reconstruction, a set of prototype geometric equations is introduced to calculate the 3-D locations (x,y,z) for each corresponding point. This algorithm for 3-D reconstruction was evaluated using different camera geometries, and using different objects. The results show that the target dimension estimated from the 3-D images has a small Root-Mean-Square-Error (RMSE) as compared to the actual dimension of the target.

2.2 Introduction

Security has become more important in both private and public areas. Camera surveillance systems are widely used for security purposes [1-5]. In order to cover the area of interest, there are often multiple cameras present that have overlapping fields of view. These digital images of the same scene can be used to extract three-dimensional (3-D) information of the objects in the overlapping fields of view, such as the height of the person, the size of an object in that scene, or object distance [6-9]. 3-D image reconstruction from sets of two-dimensional (2-D) images using stereo vision has been an area of active research for many decades, and has been applied in many fields, such as medical imaging [10], robot navigation [11], image analysis [12], machine vision [13], and architecture [14]. In most cases, the geometry of the stereo cameras and the scene are carefully controlled to make the processing straightforward. In this geometry, the spatial scale of the two images is guaranteed to be the same, and the stereo reconstruction problem is straightforward [15]. In stereo vision, disparities between corresponding points in the two images can be found by using the following techniques: block-matching [16], gradient-based optimization [17], feature matching [18], dynamic programming [19], graph cuts [20], and belief propagation [21]. These techniques have been successfully demonstrated, and are used in commercially

available products when the camera and the target geometries can be controlled.

In this paper, we propose a triangulation method based on the SIFT algorithm as a means of expanding the range of camera geometries from which 3-D information can be extracted. Our camera geometries are more flexible compared to standard stereo vision [22]. The optical axes of the cameras do not need to be parallel and the cameras do not need to have the same distance from object. The 3-D reconstruction process consists of two steps: preparing the images and reconstructing the 3-D image. For the first step, the Scale Invariant Feature Transform (SIFT) [23] is applied to rescale and rectify the images. Some candidate matching points output by SIFT are incorrect, and including them in subsequent processing has negative effects on the 3-D image reconstruction. These incorrect matching points (outliers) are eliminated by using the RANdom SAmple Consensus (RANSAC) algorithm [24]. RANSAC is an iterative method to create a mathematical model fit to remove outlier data. In the next step, the Sum of Absolute Differences (SAD) block matching technique [25, 26] is used to find the corresponding points between the left and the right images. Triangulation-based geometric equations are used to calculate the 3-D location of each corresponding point. This 3-D data may be used to extract detailed shape information of objects in the scene. A comparison of the 3-D information with measurements of the target shows that the result is accurate to within small errors on the order of a few centimeters. The errors are evaluated more completely in the experimental section.

The remainder of the paper is organized as follows. Image preparation and triangulation-based geometric 3-D reconstruction regarding the proposed geometry, our approach for

calculating the 3-D object point locations, is presented in Section 2. Experimental results showing 3-D image reconstructions and the errors between the actual size of the object and the measured size of the object are presented in Section 3. Conclusions are discussed in Section 5.

2.3 Image Preparation and Triangulation-Based Geometric 3-D Reconstruction

Our approach is to extracting 3-D information from 2-D overlapping images taken by two cameras that do not need to be on the same baseline, and do not need to be parallel like the standard stereo vision [22]. The cameras can also be rotated around the axes and have the different distance from the object. The different distances from the object results in that the camera positions may have z-axis displacement from each other; so, the cameras do not need to be on the epipolar line as in conventional stereo vision. For example, the left camera can be closer to the object than the right camera, or vice versa. The 3-D model is created by finding pixels in one 2-D image that can be identified as originating from the same point in another 2-D image. This is referred to as the correspondence problem [27] in stereo reconstruction. To solve the correspondence problem, 2-D images need to be prepared using SIFT, RANSAC, and SAD. This preparation is now explained.

2.3.1 Image Rescaling and Rectification

We employ a camera model based on the single thin lens camera. The single thin lens camera [22] describes the mathematical relationship between the 3-D object points and the image points. In the thin lens model, the rays of light emitted from a point travels along paths through the lens, converging at a point behind the lens. In geometric optics, a ray

passing through the center of a thin lens is called the chief ray, which is not deflected by the lens. The image is inverted in the image plane. Figure 2.1 shows a chief ray in the thin lens camera model, except that the image plane is moved to the front of the lens instead of behind it, and in this case the image is not inverted. The perspective model explains the projection of an object point at location P to the point P′, where it is imaged as defined by a chief ray traced from P to P′ through the center of the lens.

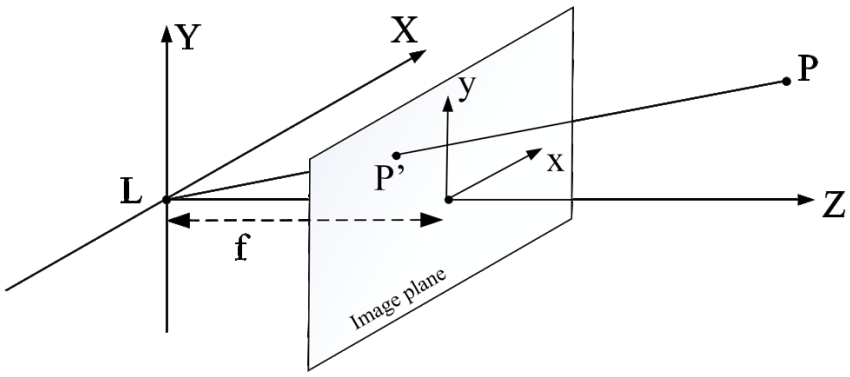
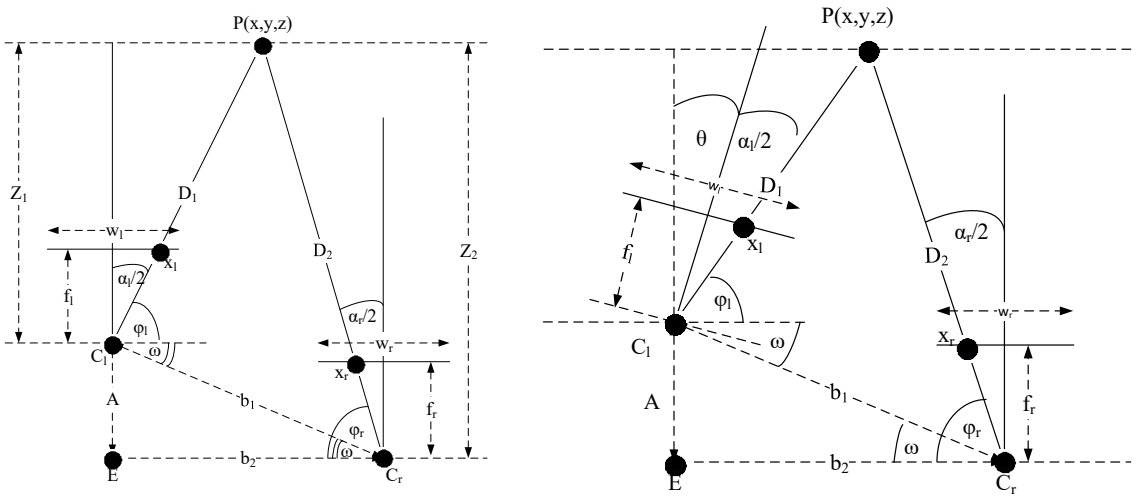


Figure 2.1 The equivalent of single thin lens camera geometry

We apply the equivalent of single thin lens camera geometry from Figure 2.1 to the geometry shown in Figure 2.2. The left and the right camera positions have z-axis displacement as shown in Figure 2.2 (a) when both optical axis of cameras are parallel and in Figure 2.2 (b) when the optical axis of the left camera is rotated around the y-axis; therefore, the target objects in the left and the right images have different scales and aspects.



b) b)

Figure 2.2 Triangular geometry

(a) both optical axis of cameras is parallel

(b) the optical axis of the left camera is rotated around the y-axis

Here, the SIFT algorithm [23] is applied to find the matching points between stereo images when a pair of stereo images have different translation and scales. However, there can be many incorrect matching points or outliers in the result of SIFT that will cause problems for the 3-D reconstruction unless some means of correcting for this effect is implemented. Here we use the RANSAC [24] algorithm to eliminate incorrect matching points after using SIFT. 2-D intensity from the left and the right images saved in 2-D array are used in the SIFT algorithm to find the matching points and then the matching point information is filtered by the RANSAC algorithm to eliminate the outliers. The remain matching point information is used to rescale one of the images. After the pair of images have the same scale, the SIFT and the RANSAC algorithms are used again to find correct matching points between the images. The new matching point information is used to rectify one of the

images. In this paper, we used SIFT and RANSAC code from an open source library called VLFeat [28]. Now both left and right images are ready to find the corresponding pixel in the next step.

2.3.2 Sum of Absolute Difference Algorithm

From a rescaled and rectified image pair, we acquire corresponding points by employing a block-matching algorithm using the SAD algorithm [25, 26]. The SAD value is computed by

$$\underset{x_l, y_l \in SR}{\operatorname{argmin}} \mathrm{SAD}(x_l, y_l, x_r, y_r) = \sum_{j=\left(-\frac{B-1}{2}\right)}^{\left(\frac{B-1}{2}\right)} \sum_{i=\left(-\frac{B-1}{2}\right)}^{\left(\frac{B-1}{2}\right)} |I_l(x_l + i, y_l + j) - I_r(x_r + i, y_r + j)| , \quad (2\text{-}1)$$

where B is the block size, SR is the search region, (x_l, y_l) is the candidate corresponding pixel in the left image, (x_r, y_r) is the interested pixel in the right image, and I_l and I_r are the pixel intensities in the left and the right images, respectively.

In Equation (1), SAD is calculated by taking the absolute difference between each pixel in a square block of certain size around the pixel of interest in the right image (reference image) and finding the corresponding pixel within the square block in the left image, while moving along the corresponding scan line or the search region. There should be only one best pair of corresponding points between the left and the right images that are determined when the value of SAD is minimum over the search region. When each pair of corresponding points between the left and the right images is found, the 3-D object point will be calculated as described in the next step until cover all corresponding points.

2.3.3 Depth of Triangulation

The typical stereo vision system [22] is set up with two cameras positioned parallel to each other, observing an object placed along the axis perpendicular to a line connecting the cameras, and centered between the cameras. In this case, the standard stereo vision geometry yields a straightforward result for finding the 3-D object points from stereo images. However, in a security system, the camera positions are relaxed. It will not always be in the parallel position, and the target will not always be on a line bisecting the cameras. In these cases, the requirements of the standard stereo vision geometry will not be satisfied. Here, we propose a new approach for reconstructing a 3-D image from a pair of cameras that are not parallel, while some parts of the images overlap with each other. Our method for calculating a set of 3-D object point positions is presented here with the geometry shown in detail in Figure 2.3.

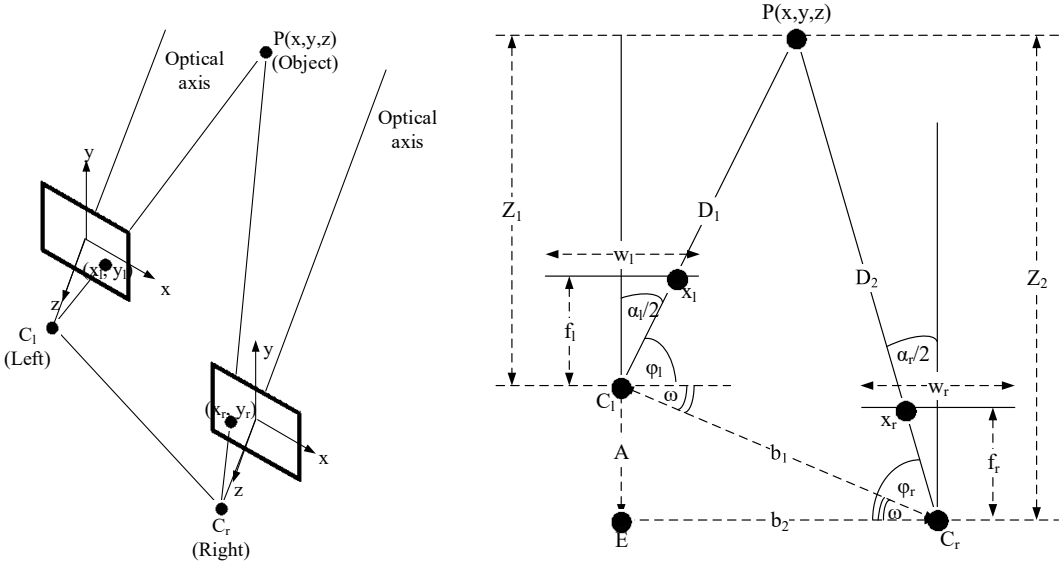


Figure 2.3 The relaxed camera positioning geometry

(a) camera Positions (b) triangular geometry

In Figure 2.3 (b), the optical axes of both cameras are parallel but the camera positions have z-axis displacement. The left camera is closer to the object than the right camera. Each best pair of corresponding points between the left and the right images from the last step will be used to calculate each 3-D object point here. The angle between the interest point and the camera position in x-axis in the left and in the right images, φ_l and φ_r, are calculated by

$$\varphi_l = \frac{\pi}{2} - \frac{\alpha_l}{2} \quad \text{and} \quad \varphi_r = \frac{\pi}{2} - \frac{\alpha_r}{2} \ , \tag{2-2}$$

where α_l and α_r are the angle between the optical axis and the interested point in the left and in the right images as calculated by

$$\frac{\alpha_l}{2} = \tan^{-1}\left(\frac{d_l}{2*f_l}\right) \quad \text{and} \quad \frac{\alpha_r}{2} = \tan^{-1}\left(\frac{d_r}{2*f_r}\right), \tag{2-3}$$

where f_l and f_r are the focal length of the lens of the left and the right camera, and d_l and d_r are the size in the left and the right image from the middle of the image to the interested point as calculated by

$$\frac{d_l}{2} = (\, x_l - \frac{w_l}{2}) \times \vartheta_l \quad \text{and} \quad \frac{d_r}{2} = (\, \frac{w_r}{2} - \, x_r \,) \times \vartheta_r \ , \tag{2-4}$$

where x_l and x_r are the points of interest in the left and the right image that represents point P of the object, w_l and w_r are the width of the left and the right image size, and ϑ_l and ϑ_r are the pixel size of the left and the right image.

In another case of the relaxed camera position, the cameras are moved arbitrarily as shown in Figure 2.4. From Figure 2.3 (b), when the left camera is rotated θ degrees clockwise

around the y-axis, the triangular geometry would be changed as shown in Figure 2.4 (a). The additional geometric considerations to accommodate this situation are as follows.

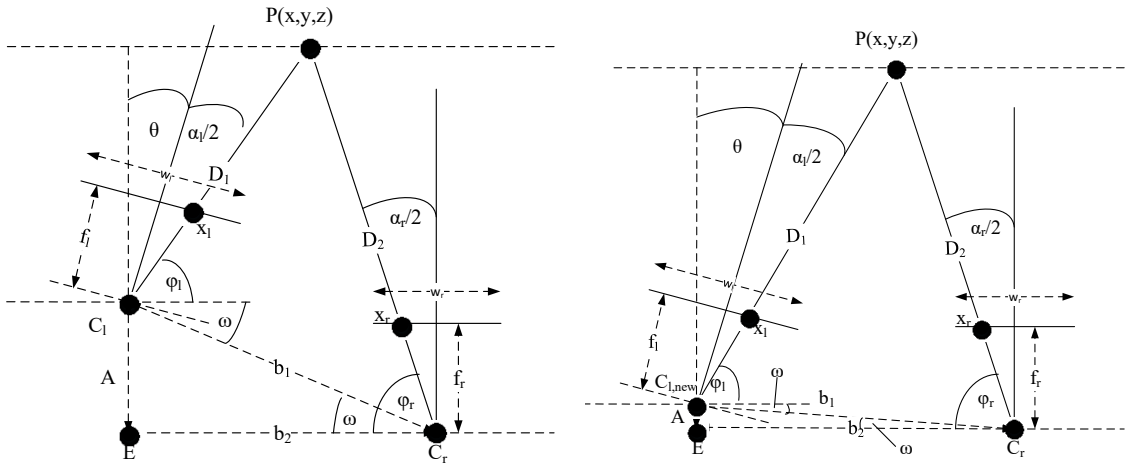


Figure 2.4 Triangular geometry used in calculations after rotating the left camera (a) before rescaling the image, and (b) after rescaling the image

From Figure 2.4 (b), after rescaling, the new left camera position $C_{l,new}$ is calculated by

$$C_{l,new} = T_2 * R_y * T_1 * C_{l,old}, \tag{2-5}$$

where $C_{l,new} = \begin{bmatrix} C_{l,new,x} \\ C_{l,new,z} \\ 1 \end{bmatrix}$, $C_{l,old} = \begin{bmatrix} C_{l,old,x} \\ C_{l,old,z} \\ 1 \end{bmatrix}$, and R_y is the rotation matrix around the y-axis with a rotation of θ degrees, T_1 is the translation matrix from the original left camera position to the origin, and T_2 is the translation matrix from the origin back to the origin of the left camera position.

For each best pair of corresponding points, a 3-D object point is calculated. When the left camera is rotated θ degrees, Equation (2-2) needs to be altered to

$$\varphi_l = \frac{\pi}{2} - \frac{\alpha_l}{2} + \theta \quad \text{and} \quad \varphi_r = \frac{\pi}{2} - \frac{\alpha_r}{2}, \tag{2-6}$$

where φ_l and φ_r are the angle between the interest point and the camera position in the left and in the right images, α_l and α_r are the angle between the optical axis and the interested point in the left and in the right images as calculated from Equations (3-4), and θ is the degree of the left camera rotation.

D_1 and D_2 are calculated by

$$D_1 = (b_1 \times \sin(\beta_r))/\sin\phi, \tag{2-7}$$

$$D_2 = (b_1 \times \sin(\beta_l))/\sin\phi, \tag{2-8}$$

where ϕ is calculated by

$$\phi = \pi - \beta_l - \beta_r, \tag{2-9}$$

and $\beta_l = \varphi_l + \omega$ and $\beta_r = \varphi_r - \omega$. The quantity ω is calculated by

$$\omega = \sin^{-1}((A \times \sin(\pi/2))/b_1), \tag{2-10}$$

where $b_1 = \sqrt{A^2 + b_2^2}$; $A = 0$ after rescaling because both camera positions became parallel in virtual scene.

The X and Z information for object points are calculated by

$$D_1 = \sqrt{(C_{lx} - X)^2 + (C_{lz} - Z)^2}, \tag{2-11}$$

$$D_2 = \sqrt{(C_{rx} - X)^2 + (C_{rz} - Z)^2}. \tag{2-12}$$

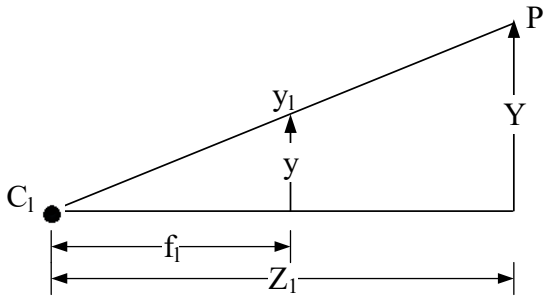


Figure 2.5 Height triangular geometry

From Figure 2.5, the Y information for each object point from the left camera is calculated by

$$Y = (y/f_l)\times Z, \tag{2-13}$$

$$y = ((h/2) - y_l)\times \vartheta_l, \tag{2-14}$$

where f_l is the focal length of the left camera, Z is the depth value from point P of the object to C_l, h is the height of the left image size (height by width), y_l is the image point in the left image that represents the location of P, and ϑ_l is the pixel size of the left image. The Y information for the object point can be calculated by using the parameters of the right camera in the same way.

The object point (X, Y, Z) calculation is repeated until all corresponding pixels are calculated. Finally, the 3-D images were displayed from the set of 3-D object points using a 3-D scatter plot in MATLAB.

2.4 Experimental Results

In this section, we describe experiments to demonstrate 3-D image reconstruction using the geometry described in Section 3. The cameras used in this study were two identical 1394a Firefly MVs, with an image size of 480-by-640 (height-by-width), pixel size ϑ=6 μm with square pixels, and a focal length of 16.6 mm. The left and the right camera positions had z-axis displacement as shown in Figure 2.3 when both cameras were parallel and in Figure 2.4 when one camera was rotated around the y-axis; therefore, the target objects in the left and the right images had different scales and aspects. In order to rescale and to rectify the target object in the image, the SIFT algorithm was used to find a set of matching points between the left and the right gray scale images. Color images needed to be converted to be gray scale images before using the SIFT algorithm. The output of SIFT was passed on to RANSAC to find and to exclude outliers from the matching set originally generated by SIFT. Next, a set of 3-D point positions in object space was calculated for each pixel between two rectified images. To find the corresponding pairs, a block-matching algorithm with the SAD in Equation (1) was used with a 67x67 block size and ± 15 pixels of search region size. After the corresponding points were found in the left image (x_l, y_l) and in the right image (x_r, y_r), the object points (X, Y, Z) were calculated by using Equations (2) - (14). Finally, the 3-D image was reconstructed from the set of 3-D object points.

To setup the experiment, there were two different conditions for the cameras' settings. The first condition is that both cameras remained parallel in the z-axis, whereas the second condition is that the left camera was rotated θ degrees clockwise around the y-axis and the right camera remained the same. For each condition, the left camera was both moved

forward and backward compared to the right camera's position. In the four camera settings, the right camera position was referenced at $C_r = (0, 0, 0)$ and both cameras were positioned at the same height. The four experiment set ups are shown in Figure 2.6.

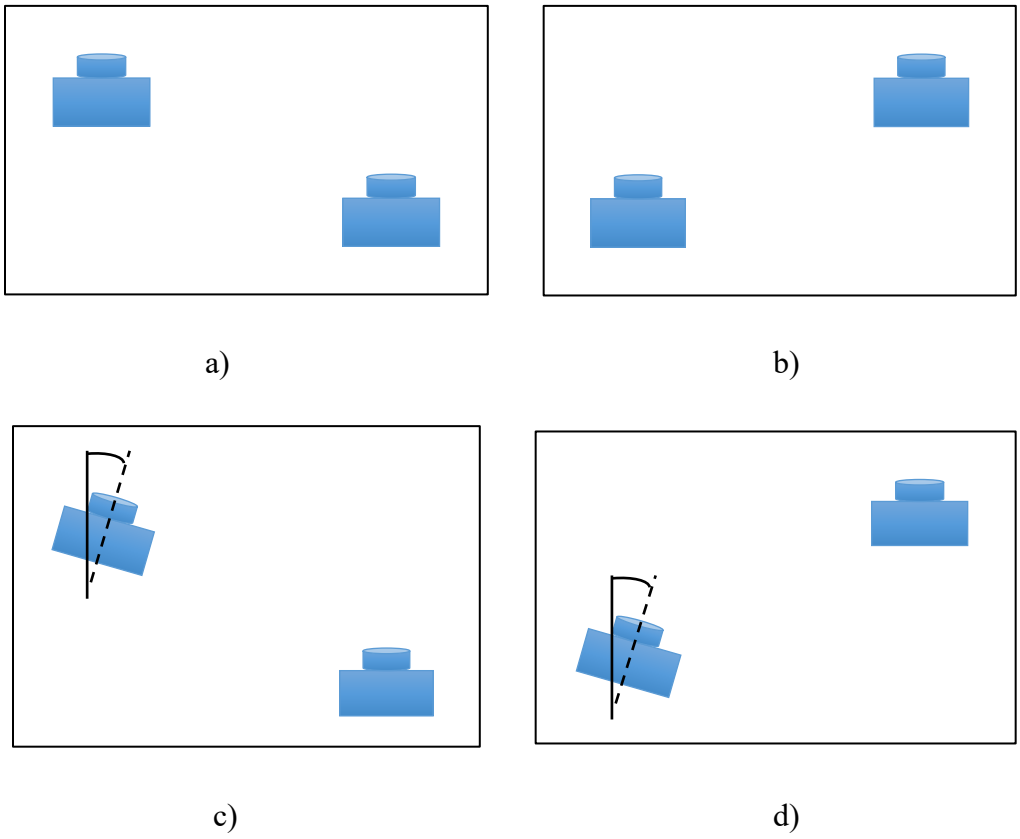


Figure 2.6 Four experiment setups from top view

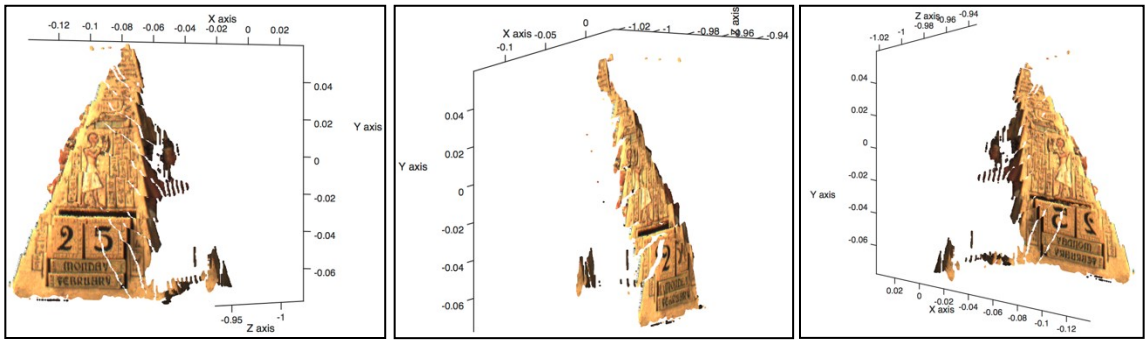
(a) both cameras were parallel in the z-axis and the left camera was moved forward.

(b) both cameras were parallel in the z-axis and the left camera was moved backward.

(c) the left camera was rotated 7 degrees clockwise around the y-axis and was moved forward.

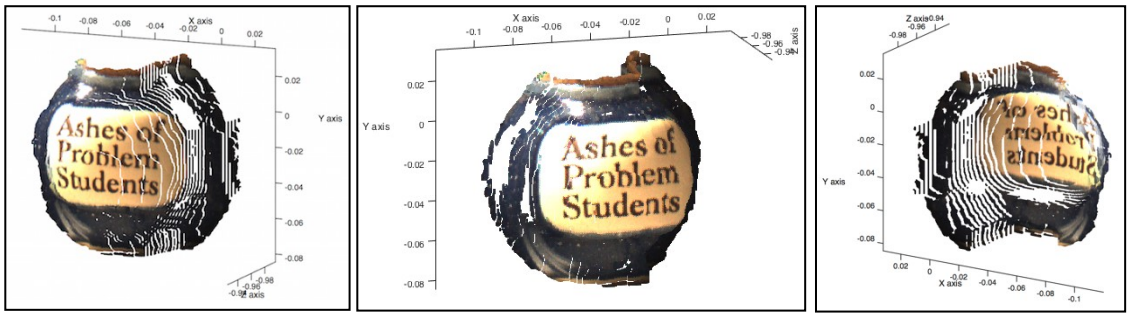
(d) the left camera was rotated 4.5 degrees clockwise around the y-axis and was moved backward.

The 3-D images were reconstructed from each pair of images taken from all for camera scenarios. Images were taken of five different objects in four experiments. The five objects were a jar, a fox, two dolls, an engine model, and a pyramid. There were two examples where the 3-D images were looked at from multiple viewpoints as shown in Figures 2.7 and 2.8. All 3-D image reconstructions created from 2-D images of the five objects in different conditions and cases are shown in Figures 2.9-2.12. The error between the object actual size and 3-D image reconstructions for all cases of different objects are calculated and shown in Table 2.1.



a) b) c)

Figure 2.7 3-D pyramid image from different viewpoints

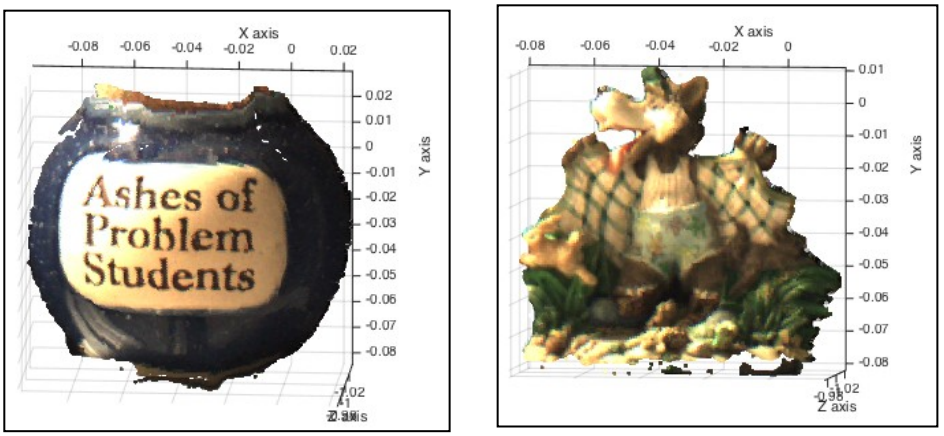


a) b) c)

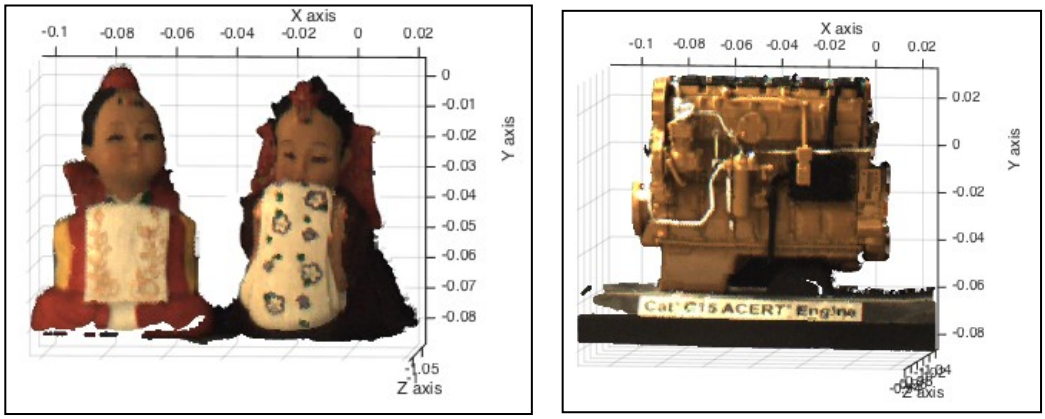
Figure 2.8 3-D jar image from different viewpoints

Figures 2.7 and 2.8 showed that the result of reconstruction of the pyramid and jar from different viewpoints had discontinuous surfaces because of the quantization noise when the cameras captured the real world objects into the pixels of the digital images. These pixels could not represent the continuity of the surface of the objects. This is similar to when converting analog to digital.

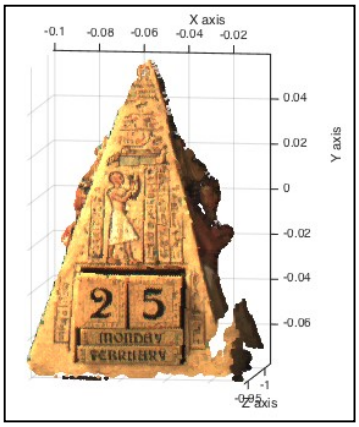
Figure 2.9 Setup one: both cameras were parallel in the *z*-axis and the left camera was moved forward.



a) b)



c) d)



e)

Figure 2.10 Setup two: both cameras were parallel in the z-axis and the left camera was moved backward.

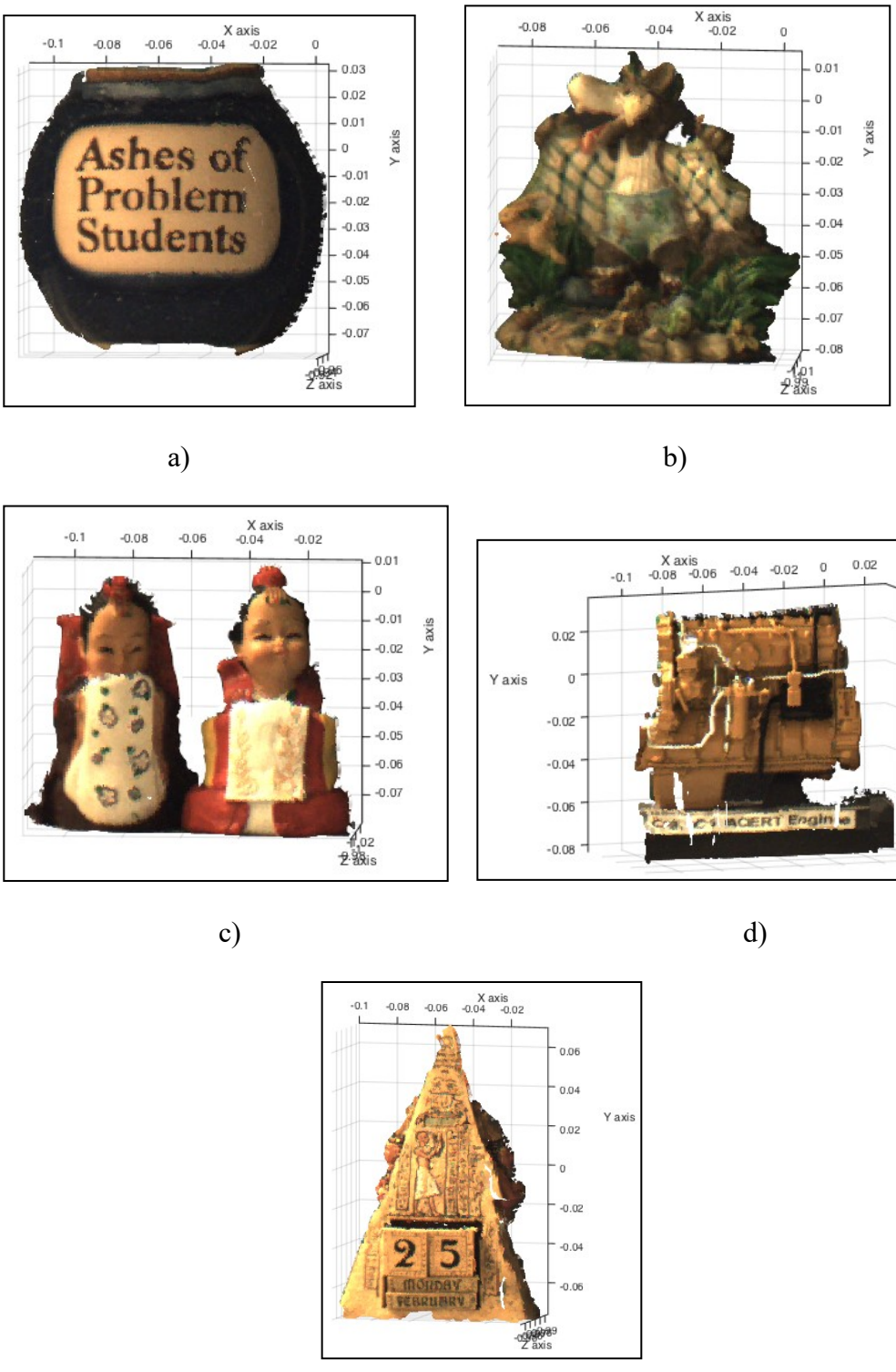


Figure 2.11 Setup three: the left camera was rotated 7 degrees clockwise around the y-axis and was moved forward.

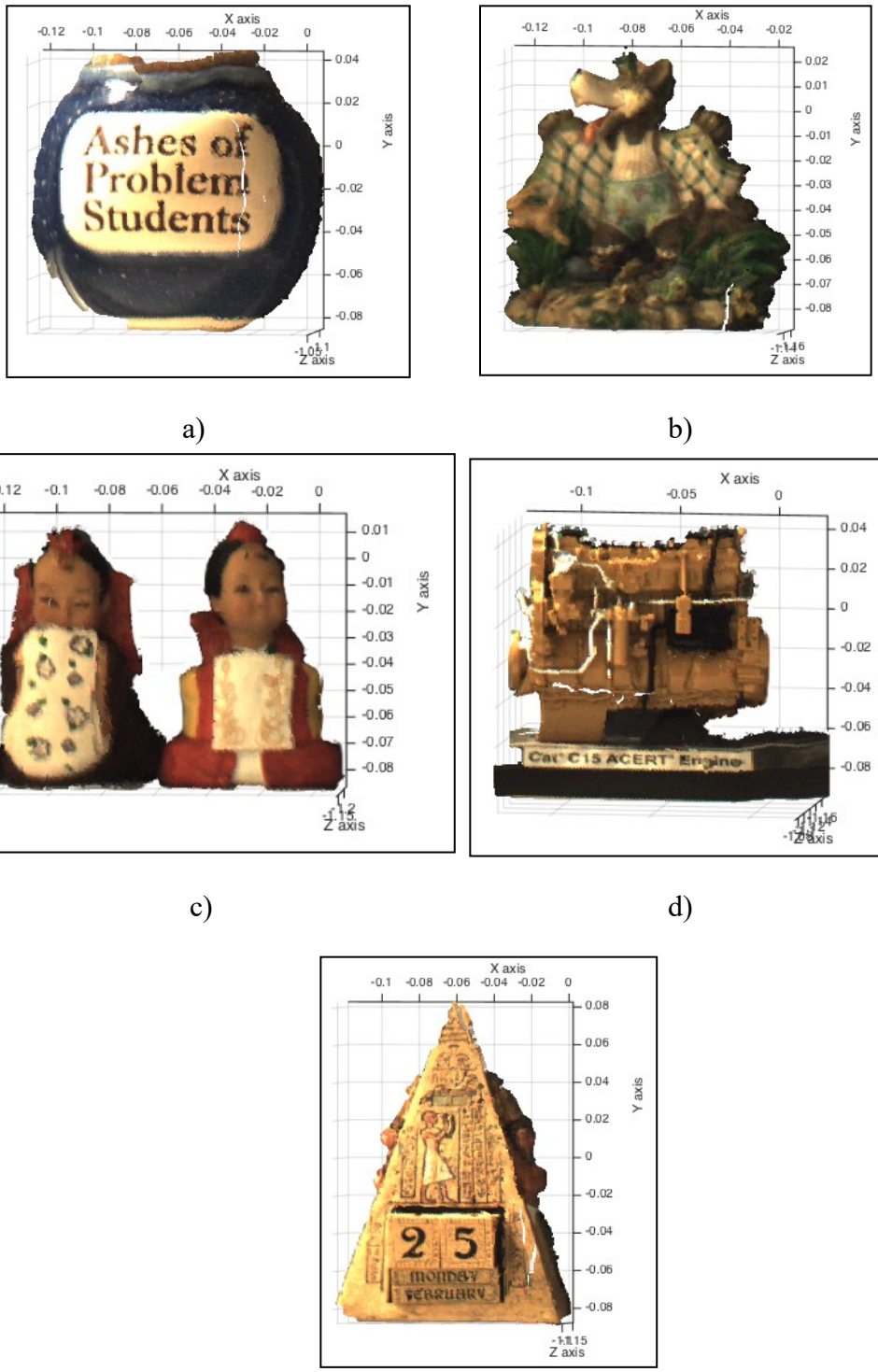


Figure 2.12 Setup four: the left camera was rotated 4.5 degree clockwise around the *y*-axis and was moved backward.

It can be noticed from Figures 2.9-2.12 that the 3-D images contain enough quality 3-D information to represent one side of the actual object. Table 2.1 shows the error values between the actual object size and 3-D image size for height and width for all conditions and cases. Experiments were performed to evaluate the prototype 3-D geometry algorithm by using RMSE. Table 2.2 shows RMSE measurement of object sizes between the actual size and the 3-D image size. There are some errors in 3-D reconstructions because there are incorrect matching points during block matching process. This can be attributed to errors in a search region—if the intensity of pixels is about the same, they will give similar results for SAD that lead to high probability of generating the incorrect matching points.

Table 2.1 The error values between the actual object size and the 3-D image size (unit: cm)

Object			First condition				Second condition			
Name	Size		First case		Second case		First case		Second case	
			3-D image	Error	3-D image	Error	3-D image	Error	3-D image	Error
Pyramid	Height	16.5	13.9	2.6	13.76	2.74	16	0.5	14	2.5
	Width	10.5	10	0.5	9.05	1.45	11	-0.5	9.7	0.8
Fox	Height	10.5	8.78	1.72	8.9	1.6	10	0.5	9.6	0.9
	Width	10	9.8	0.2	9	1	11.5	-1.5	9.5	0.5
Engine	Height	9.5	8.76	0.74	8.98	0.52	7.6	1.9	9.21	0.29
	Width	11.5	10.24	1.26	9.7	1.8	9.4	2.1	10.96	0.54
Doll	Height	9.5	8.6	0.9	8	1.5	9.75	-0.25	8.89	0.61
	Width	6	6.1	-0.1	5.48	0.52	6.4	-0.4	6	0
Jar	Height	12	10.37	1.63	9.76	2.24	11.9	0.1	10.13	1.87
	Width	12	12	0	11	1	11	1	11.4	0.6

Table 2.2 Qualitative results for the 3-D images for all cases (unit:cm)

	First condition		Second condition	
	First case	Second case	First case	Second case
RMSE	1.249	1.588	1.106	1.12

2.5 Conclusions

In this study, we proposed a triangular geometry to calculate 3-D information objects. This set of equations were used with processed images when the two cameras had x and z-displacement shift, and when one camera was rotated around the y-axis. Therefore, a set of 3-D object points could be calculated.

The findings of the study showed that the 3-D information captured in this manner has enough quality to represent one side of the actual object. The RMSE between the actual size and the measured 3-D image result in the first case when the left camera was moved backward for both conditions are less than the second case when the left camera was moved forward. The average RMSE is equal to 1.265. The results indicated that our set of prototype geometric equations could be used to calculate the 3-D information that can build a 3-D image with high reliability.

2.6 References

1. Robert-Inacio, F., A. Raybaud, and E. Clement, *Multispectral target detection and tracking for seaport video surveillance.* Proceedings of the IVS Image and Vision Computing New Zealand, 2007: p. 169-174.
2. Hampapur, A., et al., *Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking.* Signal Processing Magazine, IEEE, 2005. 22(2): p. 38-51.
3. Valera, M. and S.A. Velastin. *Intelligent distributed surveillance systems: a review*. in *Vision, Image and Signal Processing, IEE Proceedings-*. 2005. IET.

4. Bird, N., et al. *Real time, online detection of abandoned objects in public areas*. in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. 2006. IEEE.
5. Singh, A., et al. *An abandoned object detection system based on dual background segmentation*. in *Advanced Video and Signal Based Surveillance, 2009. AVSS'09. Sixth IEEE International Conference on*. 2009. IEEE.
6. Cucchiara, R. *Multimedia surveillance systems*. in *Proceedings of the third ACM international workshop on Video surveillance & sensor networks*. 2005. ACM.
7. Calderara, S., et al. *Entry edge of field of view for multi-camera tracking in distributed video surveillance*. in *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*. 2005. IEEE.
8. Mustafah, Y.M., A.W. Azman, and M.H. Ani. *Object Distance and Size Measurement Using Stereo Vision System*. in *Advanced Materials Research*. 2013. Trans Tech Publ.
9. Nedevschi, S., et al. *High accuracy stereo vision system for far distance obstacle detection*. in *IEEE Intelligent Vehicles Symposium*. 2004.
10. Stoyanov, D., et al., *Real-time stereo reconstruction in robotically assisted minimally invasive surgery*, in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010*. 2010, Springer. p. 275-282.
11. Murray, D. and J.J. Little, *Using real-time stereo vision for mobile robot navigation.* Autonomous Robots, 2000. 8(2): p. 161-171.
12. Kim, H., S.-j. Yang, and K. Sohn. *3d reconstruction of stereo images for interaction between real and virtual worlds*. in *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*. 2003. IEEE.
13. Suhr, J.K., et al., *Automatic free parking space detection by using motion stereo-based 3D reconstruction.* Machine Vision and Applications, 2010. 21(2): p. 163-176.
14. Pollefeys, M., et al., *Detailed real-time urban 3d reconstruction from video.* International Journal of Computer Vision, 2008. 78(2-3): p. 143-167.
15. Dhond, U.R. and J.K. Aggarwal, *Structure from stereo-a review.* IEEE transactions on systems, man, and cybernetics, 1989. 19(6): p. 1489-1510.
16. Tao, T., J.C. Koo, and H.R. Choi. *A fast block matching algorthim for stereo correspondence*. in *Cybernetics and Intelligent Systems, 2008 IEEE Conference on*. 2008. IEEE.
17. Lucas, B.D. and T. Kanade. *An iterative image registration technique with an application to stereo vision*. in *IJCAI*. 1981.
18. Venkateswar, V. and R. Chellappa, *Hierarchical stereo and motion correspondence using feature groupings.* International Journal of Computer Vision, 1995. 15(3): p. 245-269.
19. Park, C.S. and H.W. Park, *A robust stereo disparity estimation using adaptive window search and dynamic programming search.* Pattern Recognition, 2001. 34(12): p. 2573-2576.
20. Boykov, Y., O. Veksler, and R. Zabih, *Fast approximate energy minimization via graph cuts.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2001. 23(11): p. 1222-1239.

21. Sun, J., N.-N. Zheng, and H.-Y. Shum, *Stereo matching using belief propagation.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2003. 25(7): p. 787-800.
22. Hartley, R. and A. Zisserman, *Multiple view geometry in computer vision*. 2003: Cambridge university press.
23. Lowe, D.G., *Distinctive image features from scale-invariant keypoints.* International journal of computer vision, 2004. 60(2): p. 91-110.
24. Fischler, M.A. and R.C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography.* Communications of the ACM, 1981. 24(6): p. 381-395.
25. Bhaskaran, V. and K. Konstantinides, *Image and video compression standards: algorithms and architectures. 1997*. Kluwer Academic Publishers, Norwell, MA, USA.
26. Vassiliadis, S., et al. *The sum-absolute-difference motion estimation accelerator*. in *Euromicro Conference, 1998. Proceedings. 24th*. 1998. IEEE.
27. Scharstein, D. and R. Szeliski, *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms.* International journal of computer vision, 2002. 47(1-3): p. 7-42.
28. Fulkerson, A.V.a.B. *VLFEAT: An Open and Portable Library of Computer Vision Algorithms*. 2008; Available from: http://www.vlfeat.org/.

Chapter 3: Factors Effecting Real Time Optical Monitoring of Fused Filament 3-D printing[2]

3.1 Abstract

This study analyzes a low-cost reliable real-time optimal monitoring platform for fused filament fabrication-based open source 3-D printing. An algorithm for reconstructing 3-D images from overlapping 2-D intensity measurements with relaxed camera positioning requirements is compared with a single camera solution for single side 3-D printing monitoring. The algorithms are tested for different 3-D object geometry and filament colors. The results showed that both algorithms with a single and double camera system were effective at detecting a clogged nozzle, incomplete project, or loss of filament for a wide range of 3-D object geometries and filament colors. The combined approach was the most effective and achieves 100 percent detection rate for failures. The combined method analyzed here has a better detection rate and a lower cost compared to previous methods. In addition, this method is generalizable to a wide range of 3-D printer geometries, which enables further deployment of desktop 3-D printing as wasted print time and filament are reduced, thereby improving the economic advantages of distributed manufacturing.

3.2 Introduction

As the Stratasys patent [1] expired on fused deposition modeling (FDM) in 2009, a more generalized form (fused filament fabrication (FFF)) enabled the self-replicating rapid prototyper (RepRap) 3-D printer project [2-4] to develop and scale. The RepRap project

[2] This chapter has been completed as an article to submit. Citation: Nuchitprasitchai S, Roggemann M, & Pearce J (2017). Factors Effecting Real Time Optical Monitoring of Fused Filament 3-D Printing

was developed using open source hardware protocols [5] around the Arduino microcontroller [6-7]. The expected rapid innovation in the open source community [8] succeeded, and dropped the cost of FFF 3-D printers by several orders of magnitude [9], spawned dozens of 3-D printer startup companies and brought 3-D printing to prosumers at a rapid rate [10]. This change had helped to lower the cost of an open source RepRap style 3-D printer first under US$1000 and now to under US$500 in parts, which makes it economically viable for average consumers to offset purchases with 3-D printing [11]. RepRaps can reproduce more than half of their own components and can self-upgrade, which make them attractive for a wide range of applications including sustainable development and farming [12-14], education [15-19], rapid prototyping standard products [20,21] to microfluidics [22,23] small business manufacturing [24-27], as well as scientific tools [28-31]. However, these low-cost printers are still short of the reliability standards [32-35] that consumers are accustomed to with other consumer products. Some work has estimated a 20% failure rate for inexperienced 3-D printer users on DIY machines [11]. This is primarily due to inherent challenges of FFF printing, which although far improved in the last several years [36] persist, including: warping, elephant foot, more first layer problems, lower parts shrink, skewed prints/ shifted layers, layer misalignment, missing layers, cracks in tall objects, pillowing, stringing, under-extrusion, over-extrusion, gaps in the top layers, visible lines in the bottom layers, scars on the top surface, or no filament comes out of the nozzle [37]. These errors cost money and waste time as they reduce prosumer use due to frustration and reduce the environmental benefits of distributed manufacturing [38-42].

Several attempts have been made to improve the reliability of 3-D printers using high resolution imaging. However, the majority of this work has been based on high-cost, high resolution laser based 3-D printing systems. Kleszczynski, et al. [43] presented an overview of an error detection in an EOS INT M 270 Laser Beam Melting System with a monochrome CCD camera system, a tilt and shift lens to reduce perspective distortion, and an adjustable tube for changing height or reducing the distance between the lens and the object. Similarly, Jacobsmühlen, et al. [44] successfully applied their images to inspect a powder bed AM process result on a microscopic scale for flaw detection, missing powder or low energy input, surface quality, and measurements of part geometries. Later, Jacobsmühlen, et al. [45] showed that for providing high-resolution image-based measurements, calibration of perspective correction need to be done by using their template matching approach based on the experiment setup from [44]. Kleszczynski, et al. [46] presented two approaches to improve 3-D printing process stability including 1) using a high-resolution imaging setup and 2) an enhanced version with a proximity sensor. In addition, several commercial systems based on proprietary computer visions systems are also available for high-end printers. For example, the price for a Sintavia relies on Concept Laser for 3-D process monitoring in real-time for metal additive manufacturing system, costs around US$800,000 [47].

Relatively little work has investigated error detection in prosumer desktop FFF-based 3-D printing. This work is based primarily on monitoring Makerbot branded derivatives [48-53] of the RepRap project. Baumann, et al. met with limited success that did not support flat objects and those with similar material color with the printer using and open source software approach with OpenCV [54] and Python [55] to detect errors including

detachment, missing material flow and deformed object with a Playstation eye cam [50]. Hurd et al. successfully applied a mobile device to remotely monitor internal and external errors with Samsung Galaxy Tab 3 [48]. Ceruti et al. met with limited success using and open source software approach with Augmented Reality toolkit (AR) [56], Speeded Up Robust Features (SURF) algorithm [57], and The RANdom SAmple Consensus (RANSAC) algorithm [58] to detect the differences between a reference 3-D model (CAD) and the 3-D printing model with a camera and Augmented Reality Wuzix glasses [51]. Faes et al. had nearly zero production failure in the z-direction to detect the deposited tracks and to determine the dimension of interest in a closed-loop feedback in an Extrusion based 3-D printing (E3DP) with a modular 2-D laser triangulation scanner [49]. Straub successfully applied a visible light scanning with a multi-camera system and open source software approach with C# and Dot Net Framework [59] to detect dry printing when filament is not applied and premature job termination when project is not complete with Raspberry Pi [60], five Raspberry Pi cameras, and a visible light 3-D scanning system [52]. Straub provide an overview on how to characterize an internal structures and covered surfaces defects of complex objects with a Raspberry Pi [60], a multi-camera system (five Raspberry Pi cameras), and a visible light 3-D scanning system [53]. Flexible plastic toys production line prototype systems with the integration of a 3-D printer, industrial robot and machine vision have been demonstrated in a laboratory environment [62]. Finally, Cummings, et al. [62] presented some preliminary results with the detection and correction of filament in closed loop control for 3-D printing using ultrasonic signals with limited success.

To build on this work in order to develop a low-cost reliable real-time optimal monitoring platform for FFF-based 3-D printing, this paper undertakes a detailed study of the use of an algorithm for reconstructing 3-D images from overlapping 2-D intensity measurements with relaxed camera positioning requirements [63]. For single side 3-D printing monitoring, single and double camera solutions are compared for the following variables: six different 3-D object geometry, five filament colors. The results are compared between the two camera setups as well as the results of previously published techniques. The limitations of this approached are detailed and future work is described. The results are then discussed and conclusions are drawn in the context of furthering the adoption of desktop 3-D printing for distributed manufacturing.

3.3 Methods

For this paper, experiments were setup in two different ways: 1) using one camera to capture a 2-D image from a single 3-D printing model to do a 2-D shape image, and 2) using two cameras to capture two 2-D images from a single 3-D printing model to do a 3-D reconstruction. A different algorithm is used for each experimental setup, but the same type of camera, printer and tested objects are used. Due to the distance between the camera and the printer for the experiment setup, the field of view for both cameras can cover the printed area of 70 mm in width and 60 mm in height. To eliminate the shadow on the object scene, there should be sufficient light sources. Both experimental setups used the same 3-D printer using a delta-style RepRap, Point Grey cameras, distance between the camera and the printer, distance between the light sources and the printer, blue printing base, and filament brand. The relation of geometry between the 3-D printer and the camera system need to be known for using camera calibration technique to calculate the intrinsic and

extrinsic parameters for a specific camera setup. These parameters will be used to correct for lens distortion and to determine the location of the camera in the scene.

A low-cost (<US$500 in parts) [16] open source delta-style polymer printing RepRap (MOST Delta) is used [64]. The MOST Delta is a RepRap [65] derived from the Rostock [66] printer with a cylindrical build volume 270 mm in diameter and 250 mm high and overall dimensions of 375 mm diameter and 620 mm high. The cameras are setup on 1 side of the printer 580 mm from the outer edge as shown in Figure 3.1. The cameras used in this study are two identical 1394a Firefly MVs, with an image size of 480-by-640 (height-by-width), pixel size is 6 μm with square pixels, and a focal length of 16.6 mm. The computer models chosen are a Tyrannosaurus rex skull, cube, twisted gear vase, rectangular prism, cylinder, and triangular prism are available [67] as shown in Figure 3.2. The printing parameters used are: layer height 0.2 mm, shell thickness 1 mm, unable retraction, bottom/top thickness 1mm, fill density 20%, print speed 60 mm/s (except the skull model, which used 20 mm/s), printing temperature 180^{o}C, diameter filament 1.94mm-1.98 mm, flow filament 100%, and nozzle size 0.5 mm. The PLA filament used in this experiment is Hatchbox 3-D PLA with dimensional accuracy +/- 0.05 mm on 1 kg spools, 1.75 mm diameter with red, pink, glow, black, and orange colors.

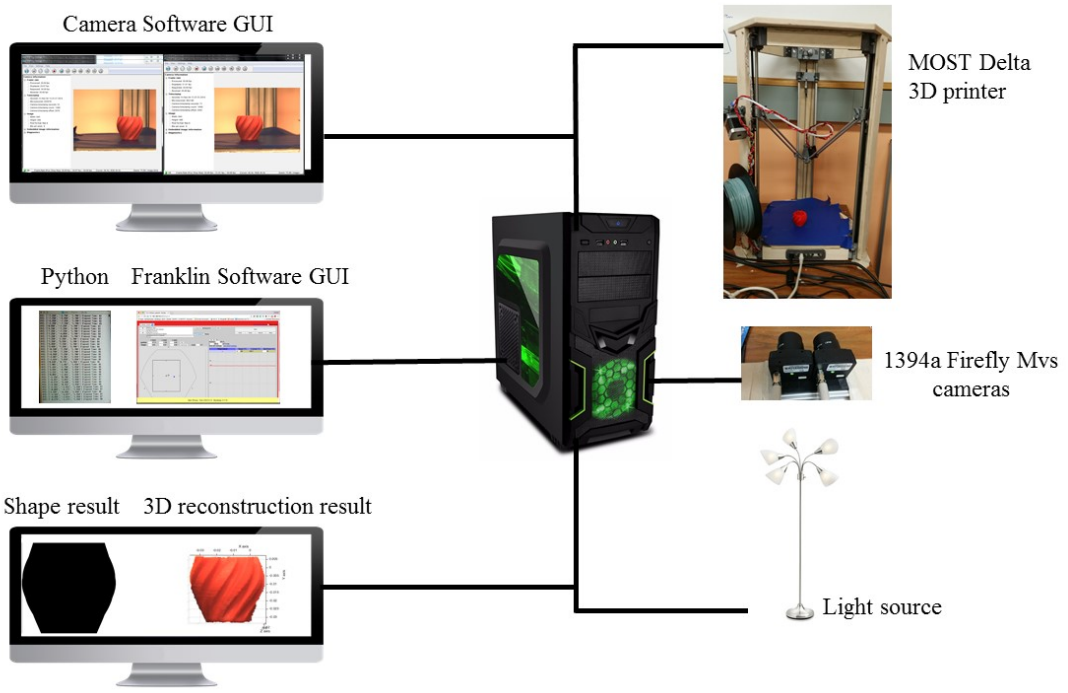


Figure 3.1 MOST Delta printer with optical monitoring experimental setup

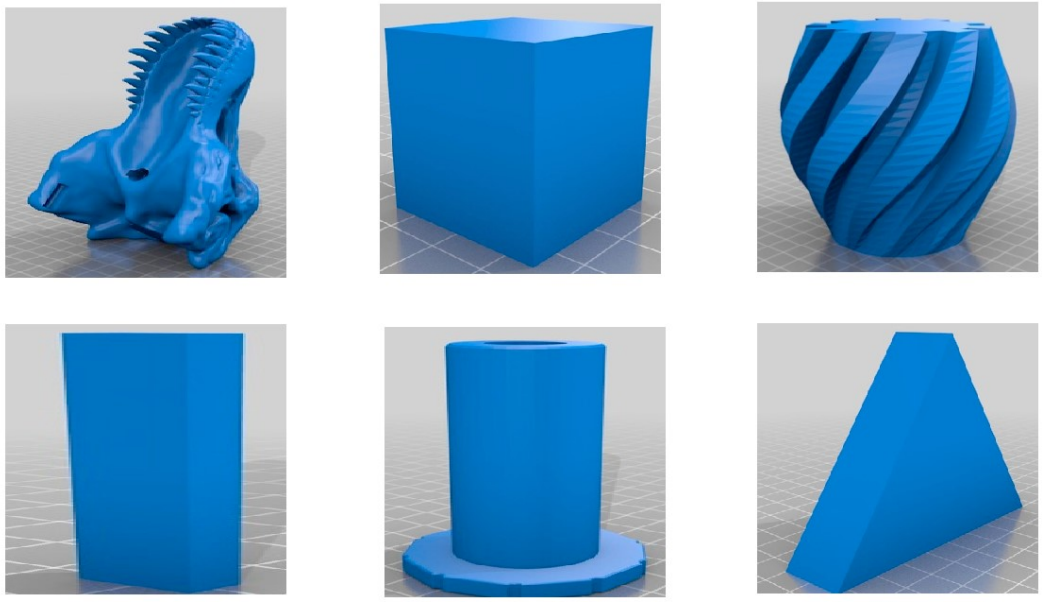


Figure 3.2 Rendering of STL models for testing: a) tyrannosaurus rex skull, b) cube, c) twisted gear vase, d) rectangular prism, e) cylinder, and f) triangular prism

3.3.1 Single Camera Setup

To detect an error from a single camera setup as shown in Figure 3.3, after simulating a 2-D shape image (cameraimage) of the 3-D object then comparing observation to the 2-D shape model (stlimage). To create a stlimage, a rendered 3-D model in OpenSCAD is saved into stl file (stlimage), then all data from stl file are plotted in x, y, z axes by using stlTools [68] to display the shape of the rendered 3-D model, which can be observed from different viewpoints. The position of the viewer for plotting the model needs to be set specify as the position of camera viewpoint while taking an image. Thus, in the right position of the viewer, the shape of the stlimage is saved as PNG image type on xz-plane. The cameraimage is created after capturing a 2-D image from the 3-D printing model. The background is then removed and rendered white. Distortion is removed from the image by intrinsic parameters from camera calibration [69] following the details in the method. Next a region of interest (ROI) is calculated from the image by converting the color image into a gray scale image, then converting it into binary image. The object area in the binary image is converted to be white used as the ROI, otherwise is converted to be black. The size of the object in stlimage and cameraimage are defined by edge detection, then the object size ratio between these two images can be found for rescaling. After rescaling, edge detection is applied again to find minimum and maximum positions of object in both images for rectification. After rectification, any errors in the process are detected by subtracting the simulated 3-D object image from the actual image. If the difference of subtraction is greater than 5%, there is an error, otherwise there is no error flagged.

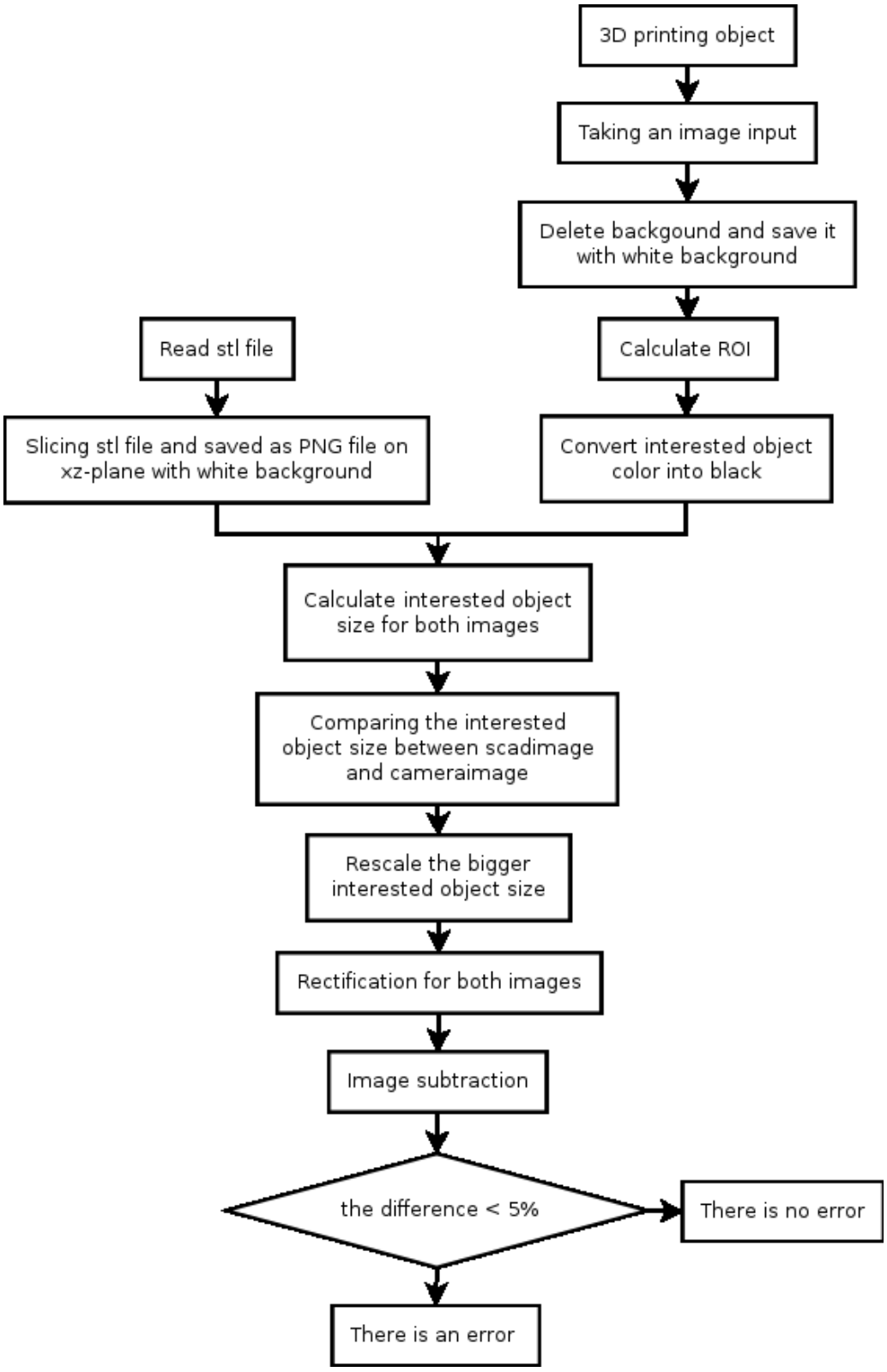


Figure 3.3 Error detection for single camera model flowchart

3.3.2 Two Camera Setup

To detect an error from the two-camera setup between a 3-D printed object and a 3-D reconstruction from two cameras the following process is used as shown in Figure 3.4 and 3.5. First, the background is removed and rendered white from the images taken from two cameras (leftimage and rightimage). Distortion and the ROI are calculated as above.

However, in the two-camera case there is another problem as points in 3-D space must be matched between the two images. To resolve this problem, the Scale Invariant Feature Transform (SIFT) [70] and the RANdom SAmple Consensus (RANSAC) [58] models are applied for rescaling and rectification. The algorithm for doing this has been described previously [63].

Next, the error detection is obtained by comparing the 3-D printed object and 3-D reconstruction image. If the difference between the two more than 5%, there is an error is identified, otherwise there is no action taken to stop the print.

3.3.3 Validation

The dimensions of the 3-D printed objects are measured with a digital caliper (+/-0.05mm). A 3-D reconstruction of the object is created from two images and the object size is calculated. Next, the size of both objects is compared to calculate size difference an error of the reconstruction. For validation of this approach six different test objects with different color filament are printed including a) Tyrannosaurus rex skull (pink), b) cube (black), c) twisted gear vase (red), d) rectangular prism (red), e) cylinder (glow), and f) triangular prism (orange).

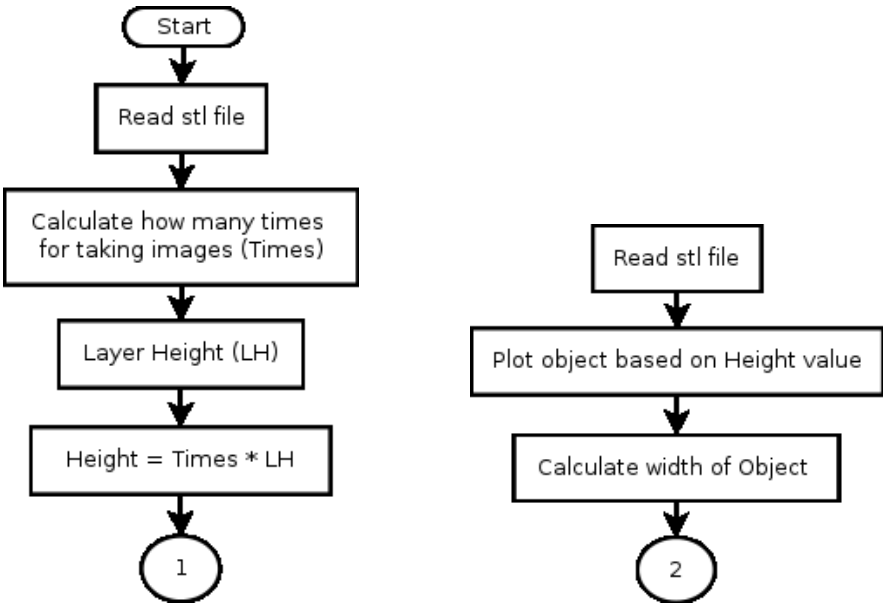


Figure 3.4 Error detection for two cameras model part 1 flowchart 1) checking 3-D object calculation, and 2) plotting stl file.

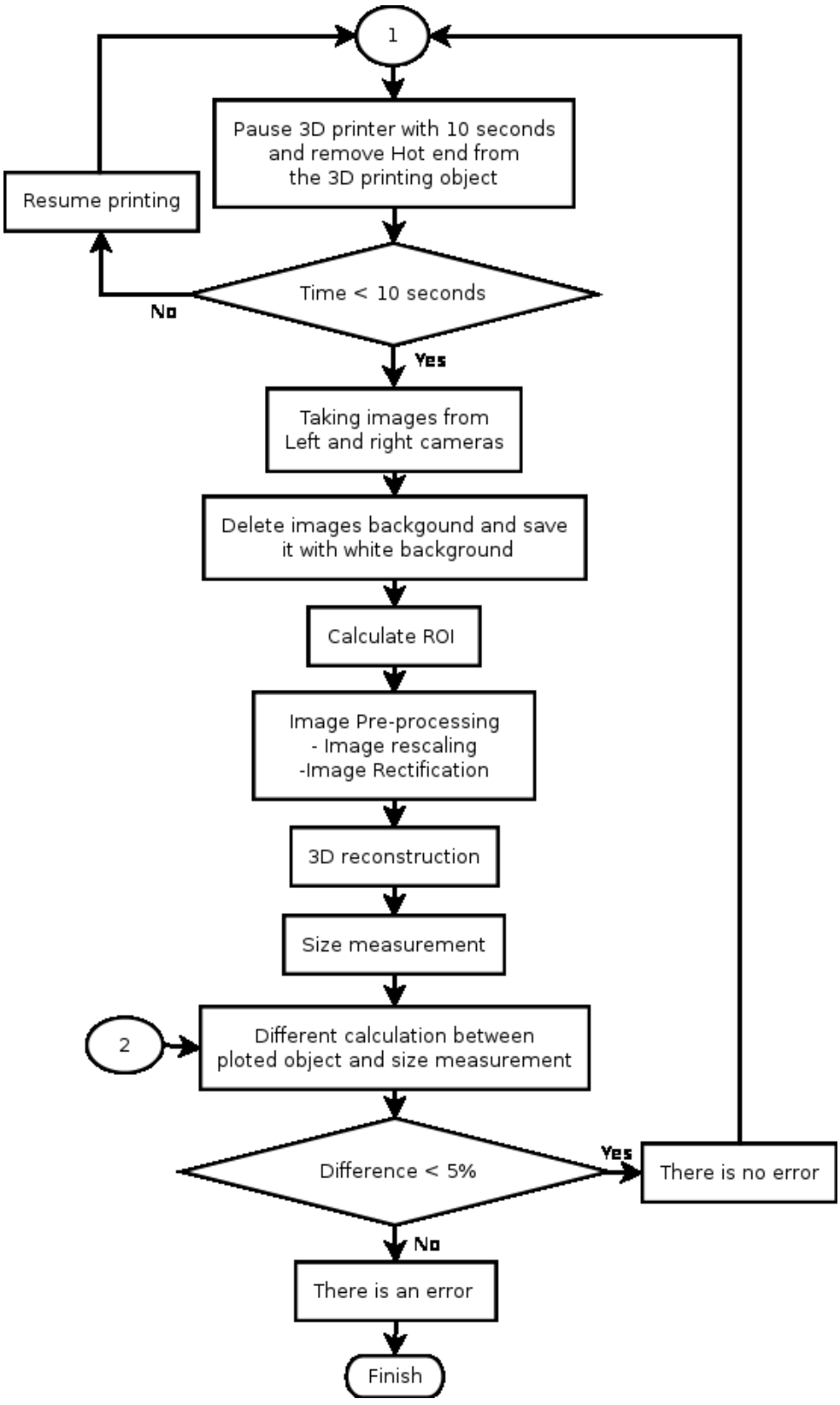


Figure 3.5 Error detection for two cameras model part 2 flowchart.

3.4 Results

The validation print images are shown in Figure 3.6. They are printed in order to detect missing material flow when the supply of filament is cut during a 3-D print.

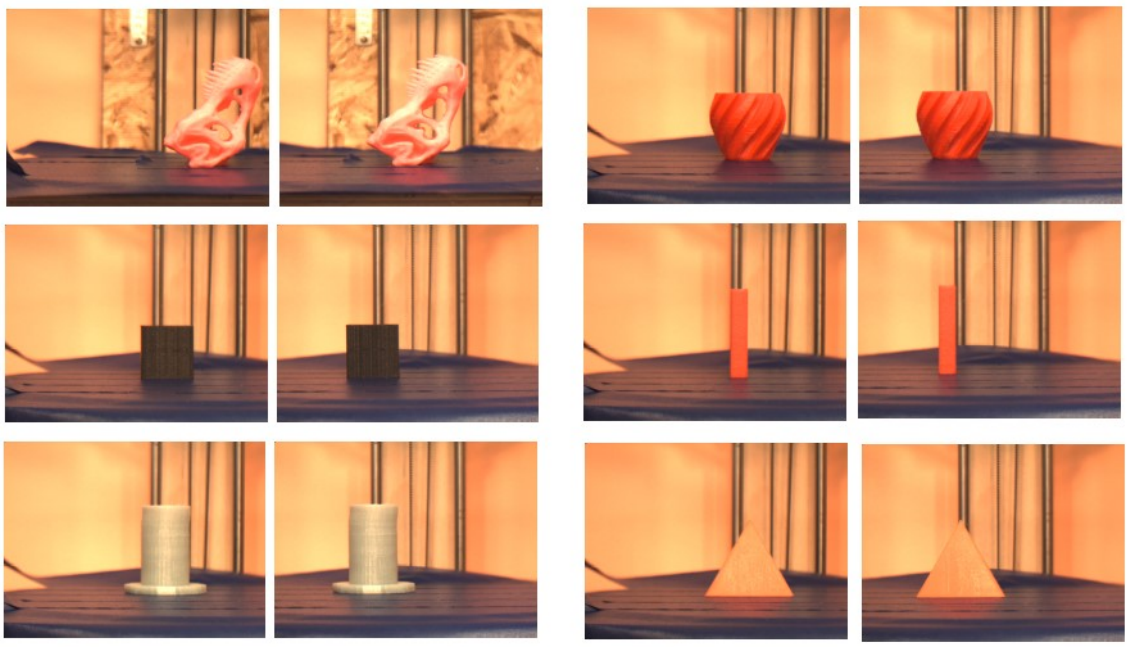


Figure 3.6 Original left and right image for different geometries with different color: a) tyrannosaurus rex skull (pink), b) cube (black), c) twisted gear vase (red), d) rectangular prism (red), e) cylinder (glow), and f) triangular prism (orange).

The error detection from one camera was tested with different geometries (Tyrannosaurus rex skull, cube, twisted gear vase, rectangular prism, cylinder, and triangular prism) with different filament colors (pink, black, red, glow, and orange) because different color gives both different 3-D printing results and can represent different challenges for image processing. The error detection system is tested with two different conditions: first is when the 3-D printer finish complete printing and second is when the 3-D printer fails and a print is incomplete. Printing is tested with different geometries are shown in Figure 3.7. Table 3.1 shows that the shape errors are between 0.984% and 2.987%. This error is acceptable

because the error of shape difference is less than 5%. The incomplete project has been tested with different geometries between the cameraimage and stlimage in different conditions as shown in Figure 3.8. Table 3.2 shows that the shape errors are greater than 5%. When the nozzle is clogged, or an incomplete project is caused by filament running out that effect the 3-D printing shapes so they are smaller than the STL models. The one exception in this case is the triangle model that is less than 5% between cameraimage (150 layers) and stlimage (200 layers) because the top of triangle has a small area.

Table 3.1 Single camera: error measurements for each geometry (W: Width, H: Height)

Object		skull		Cube		Twisted gear vase		Rectangle		Cylinder		Triangle	
Color		Pink		Black		Red		Red		Glow		Orange	
Size (mm)	Axes	W	H	W	H	W	H	W	H	W	H	W	H
	STL model	28.68	60.55	30.00	30.00	43.82	38.03	10.00	50.00	28.10	51.10	50.23	43.50
Shape error (%)		2.98		1.34		2.20		0.98		2.58		2.06	
Calculation time (sec.)		6.64		6.90		7.07		7.16		9.03		6.56	

Table 3.2 Single camera: example for error measurements when the printings fail in different layer heights

Layer heights		Error (%)					
cameraimage	stlimage	Skull	Cube	Vase	Rectangle	Cylinder	Triangle
50	100	18.68	15.87	12.71	19.36	15.06	13.76
100	150	13.82	9.97	11.97	11.54	12.19	7.60
150	200	12.95		6.78	10.04	10.13	3.39

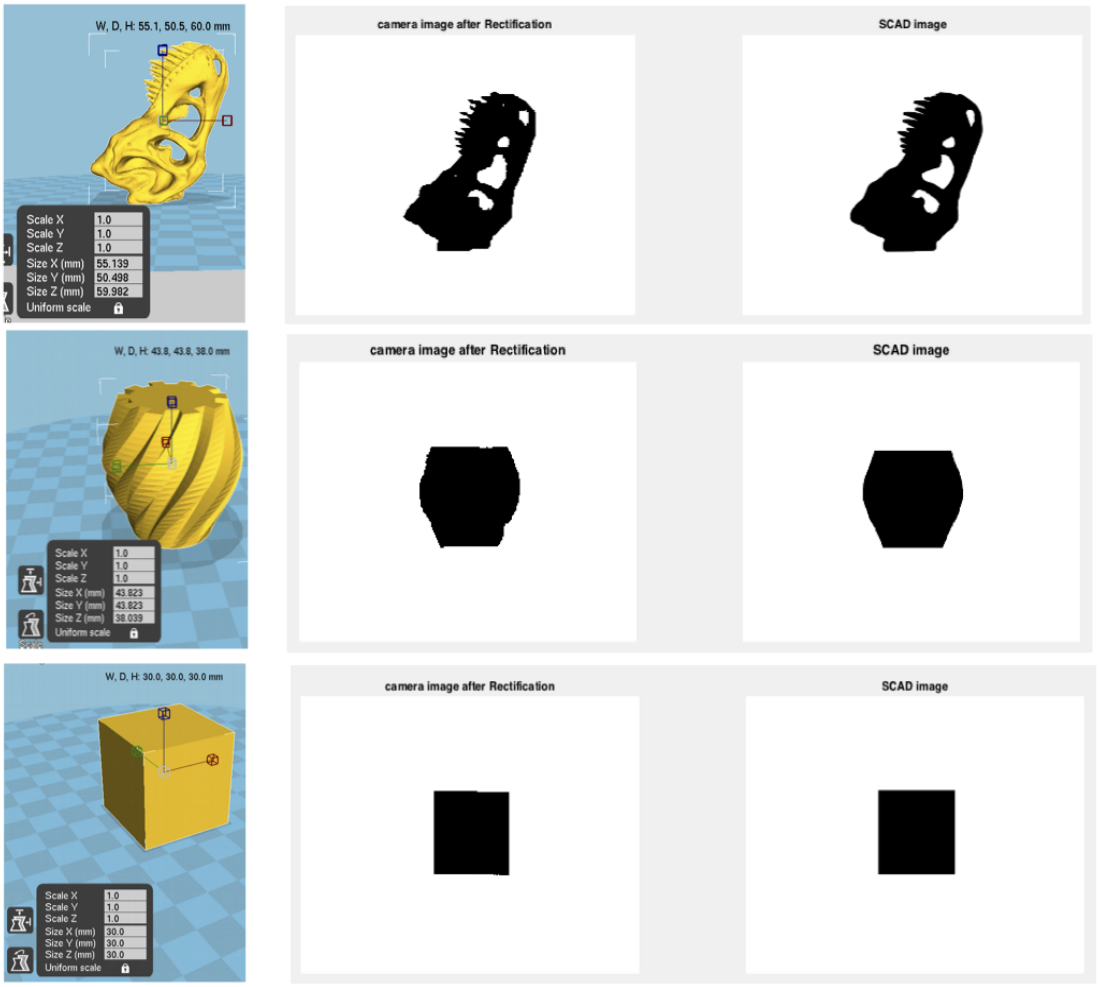


Figure 3.7 Single camera setup with different geometries: a) tyrannosaurus rex skull (pink), b) cube (black), c) twisted gear vase (red), d) rectangular prism (red), e) cylinder (glow), and f) triangular prism (orange).

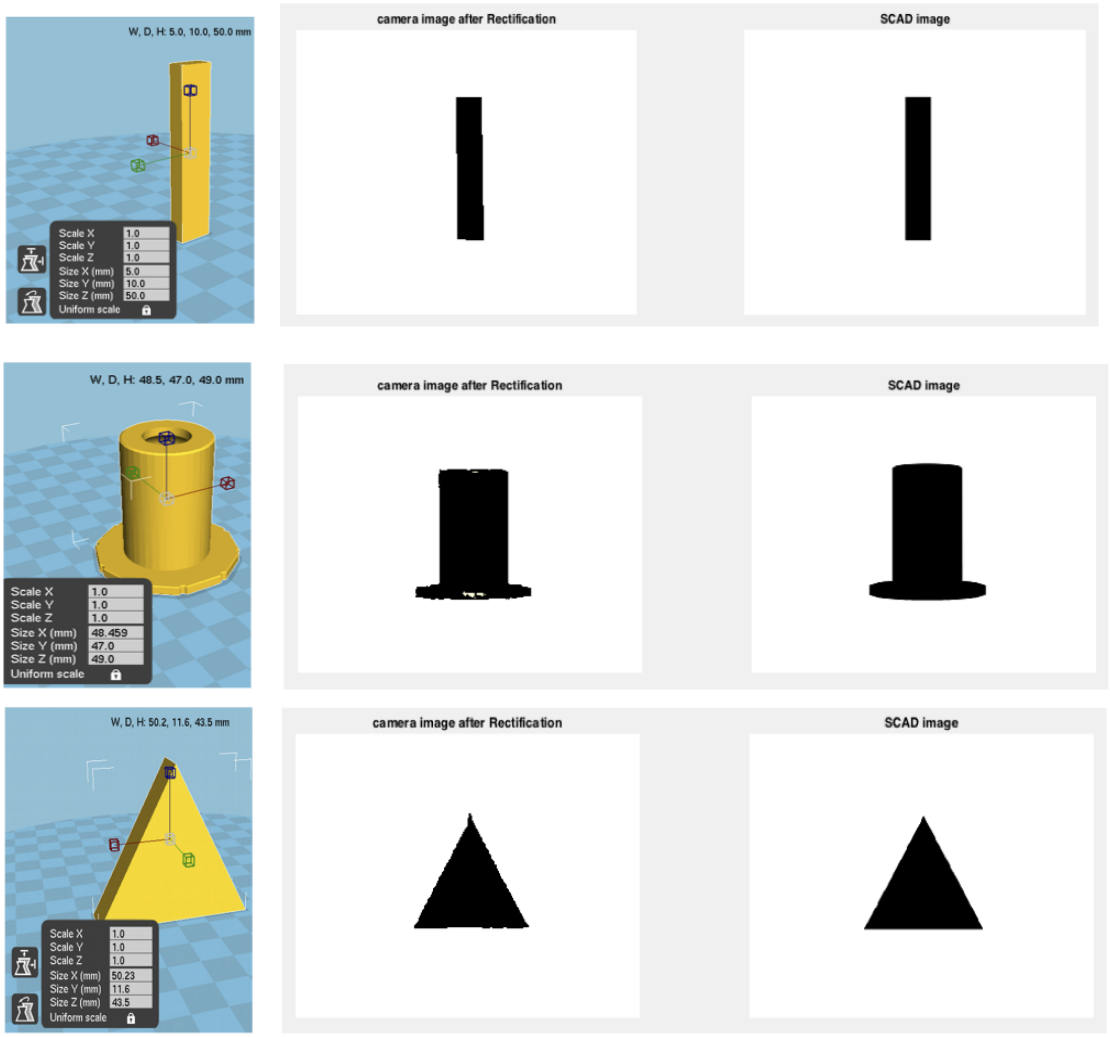


Figure 3.7 (cont.) Single camera setup with different geometries: a) tyrannosaurus rex skull (pink), b) cube (black), c) twisted gear vase (red), d) rectangular prism (red), e) cylinder (glow), and f) triangular prism (orange).

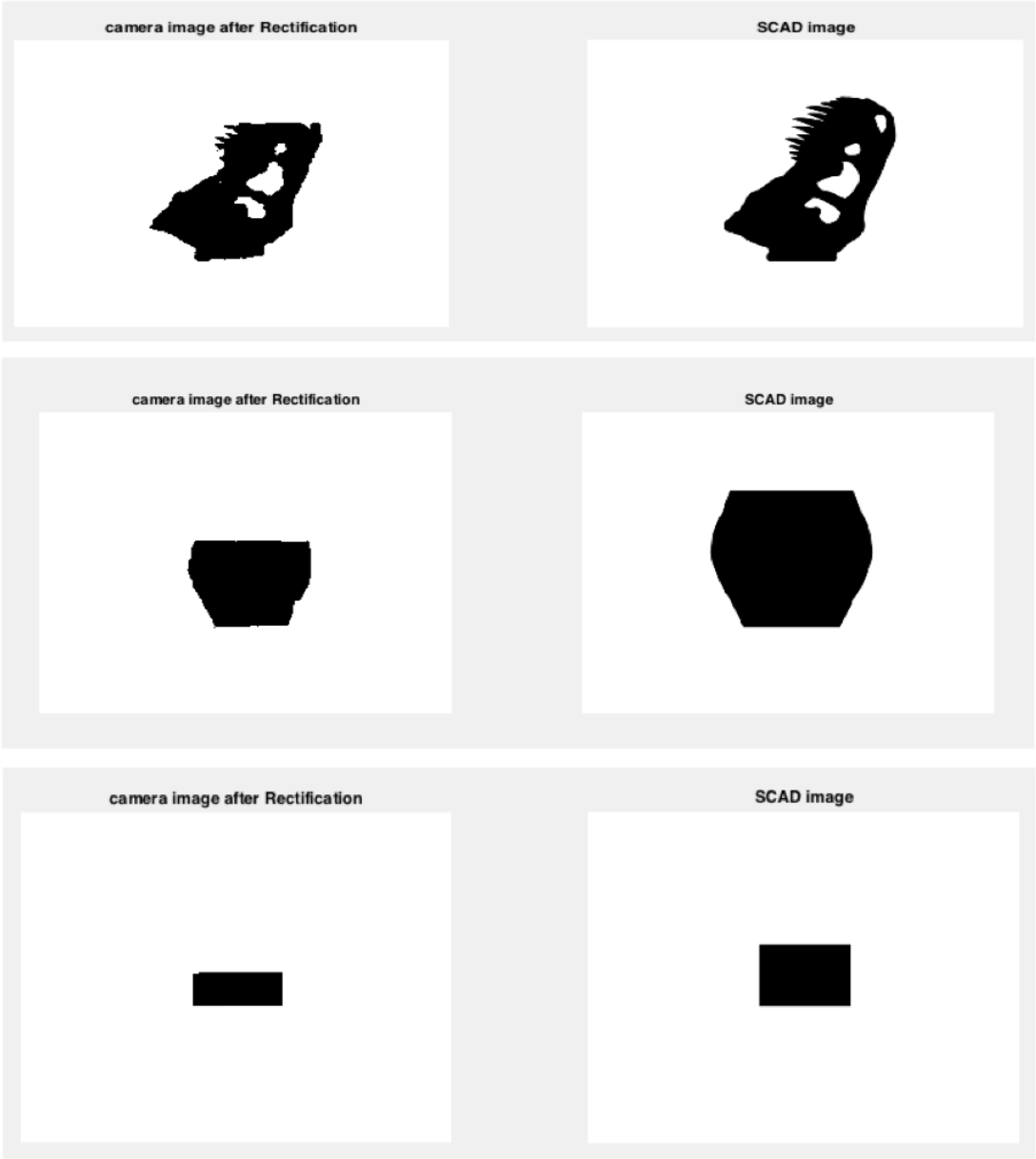


Figure 3.8 Single camera setup: error detection for different geometries between camera and STL image: a) skull model between 250 layers and full model, b) twisted gear vase model between 150 layers and full model, c) cube model between 150 layers and full model, d) rectangle model between 150 layers and 200 layers, e) cylinder model between 150 layers and full model, and f) triangle model between 100 layers and full model

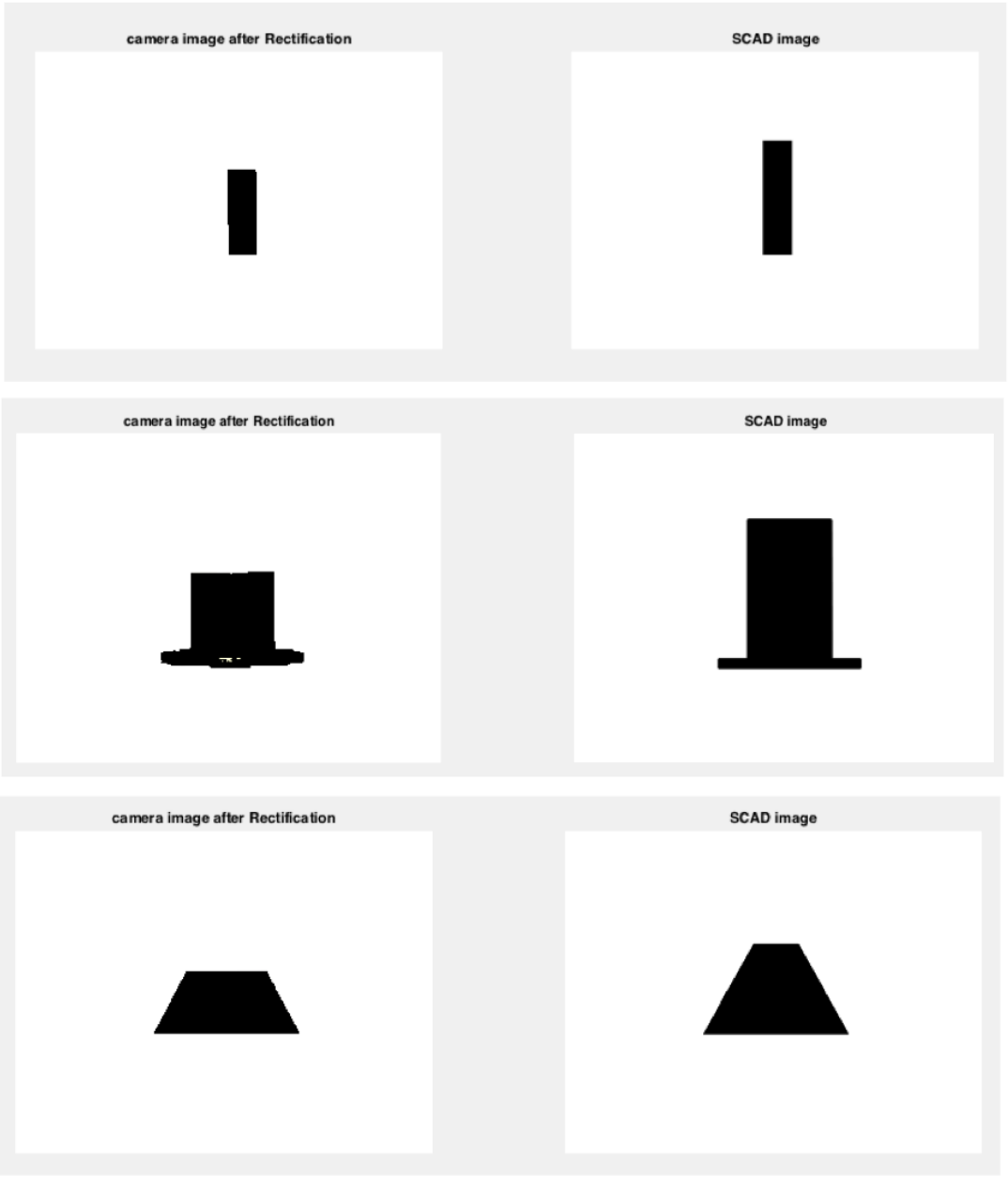


Figure 3.8 (cont.) Single camera setup: error detection for different geometries between camera and STL image: a) skull model between 250 layers and full model, b) twisted gear vase model between 150 layers and full model, c) cube model between 150 layers and full model, d) rectangle model between 150 layers and 200 layers, e) cylinder model between 150 layers and full model, and f) triangle model between 100 layers and full model

The error detection for the complete project from two cameras setup was tested with different geometries (Tyrannosaurus rex skull, cube, twisted gear vase, rectangular prism, cylinder, and triangular prism) with different filament colors (pink, black, red, glow, and orange) because different color gives us different 3-D printing results. A manual caliper is used to measure the width and the height of the real object size in millimeters (the 3-D model printing) as seen in Figure 3.9. The width and the height of the 3-D reconstruction is calculated after pointing those points on the image to get the x, y, z positions manually. The percentage of the error measurements for the complete project for each geometry with different colors for the width and the height are calculated after the difference in the width and height are found in millimeter. The 3-D reconstruction for different geometries are shown in Figure 3.9 -3.14 and the percentage of errors are less than 3.94% that there are acceptable because the error of size difference is less than 5% as shown in Table 3.3.

Table 3.3 Error measurements for complete project for each geometries with different color (W is Width and H is Height)

Object		Skull		Cube		Twisted gear vase		Rectangle		Cylinder		Triangle	
Color		Pink		Black		Red		Red		Glow		Orange	
Axes		W	H	W	H	W	H	W	H	W	H	W	H
Size (mm)	Real Object	28.68	60.55	29.86	30.23	44.01	38.50	9.59	50.14	28.65	49.48	50.73	43.58
	3-D recon-struction	28.70	61.1	30.39	30.00	44.35	38.19	9.52	51.2	29.78	50.98	51.00	44.00
Error	+/-	+0.02	+0.55	+0.53	-0.22	+0.34	-0.30	-0.07	+1.06	+1.13	+1.5	+0.27	+0.42
	%	0.07	0.90	1.80	0.75	0.77	0.79	0.70	2.11	3.94	3.03	0.54	0.96
Calculation Time (sec.)		73.88		45.66		67.05		38.73		58.78		51.56	

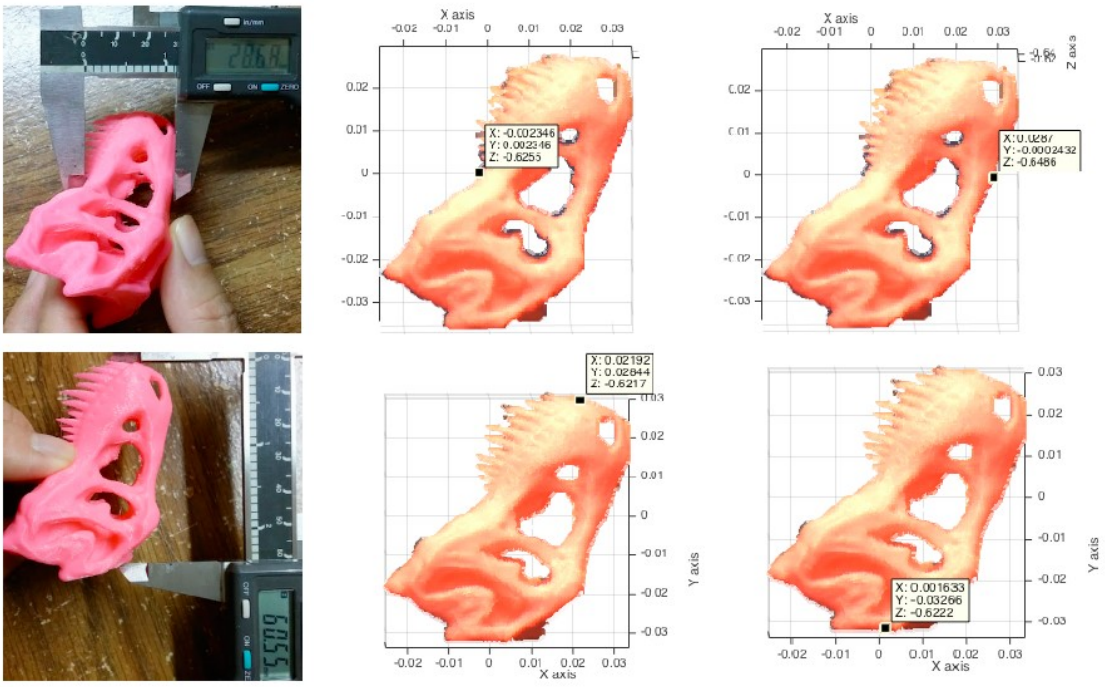


Figure 3.9 Tyrannosaurus rex skull (pink): a) width measurement and b) height measurement

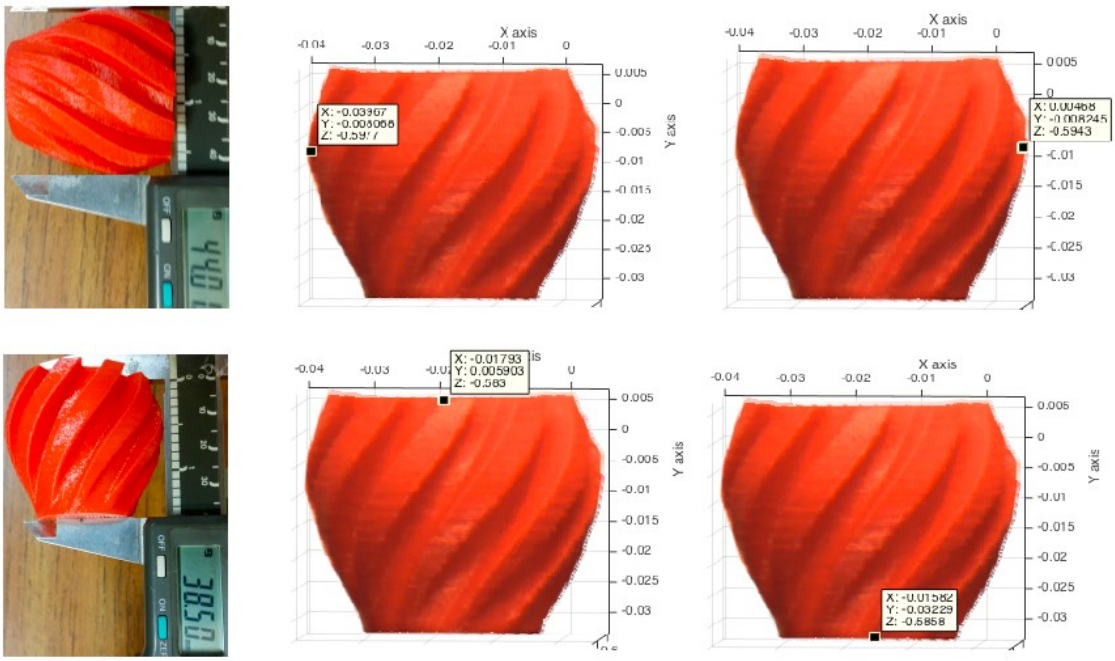


Figure 3.10 Twisted gear vase (red): a) width measurement and b) height measurement

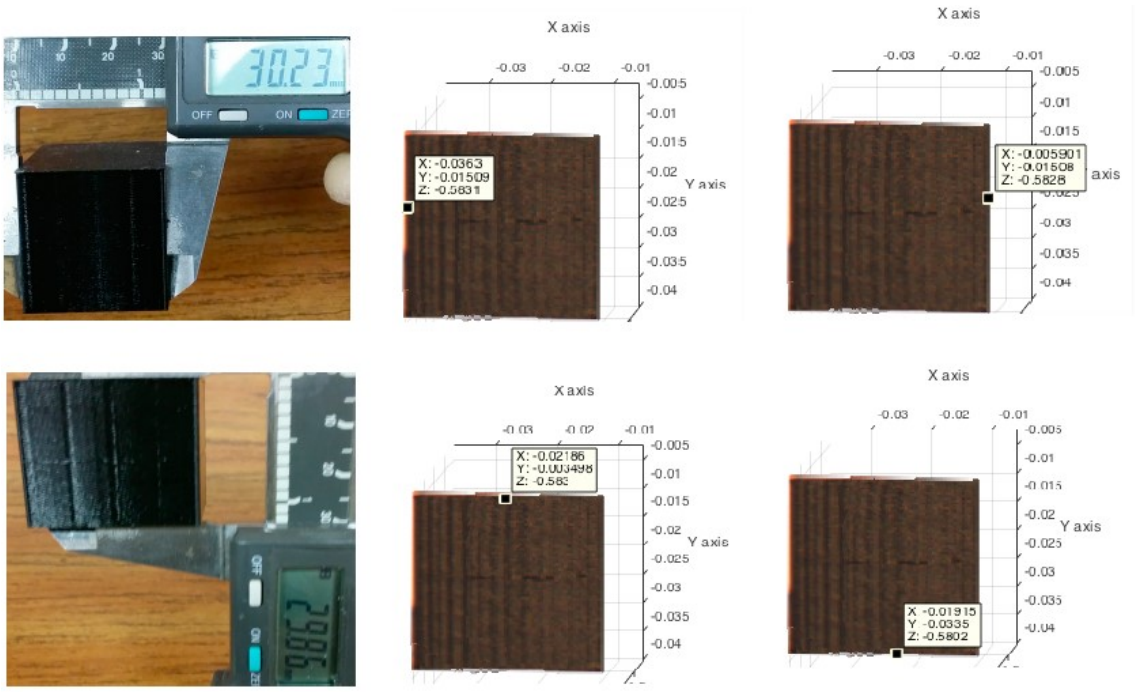


Figure 3.11 Cube (black): a) width measurement and b) height measurement

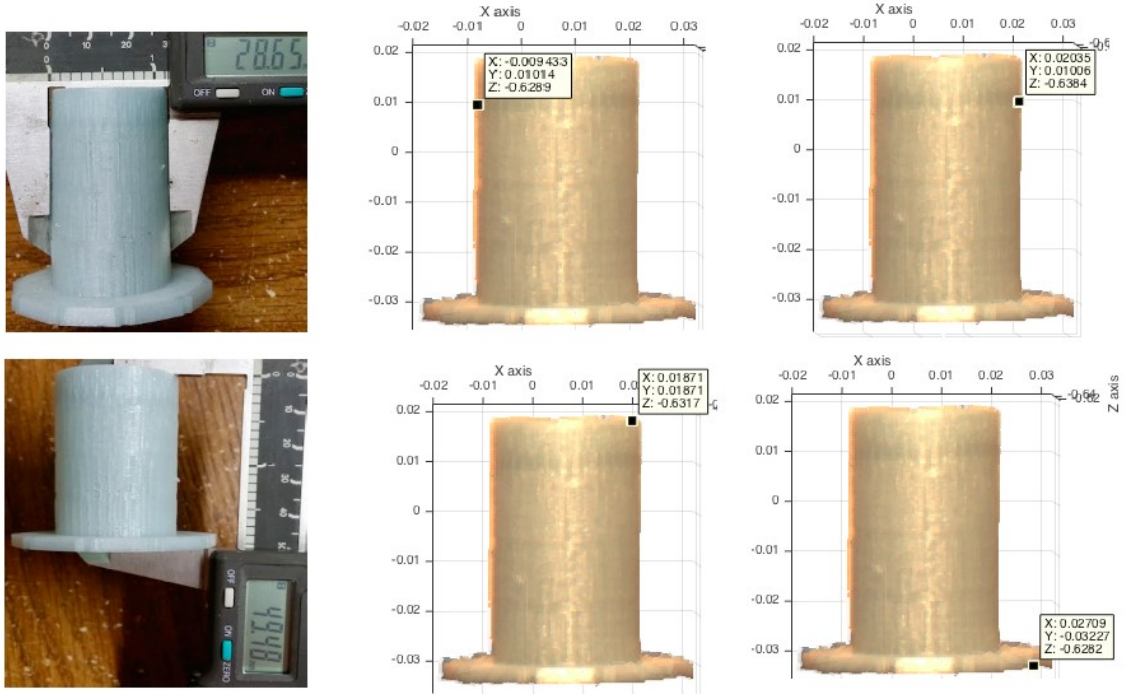


Figure 3.12 Cylinder (glow): a) width measurement and b) height measurement

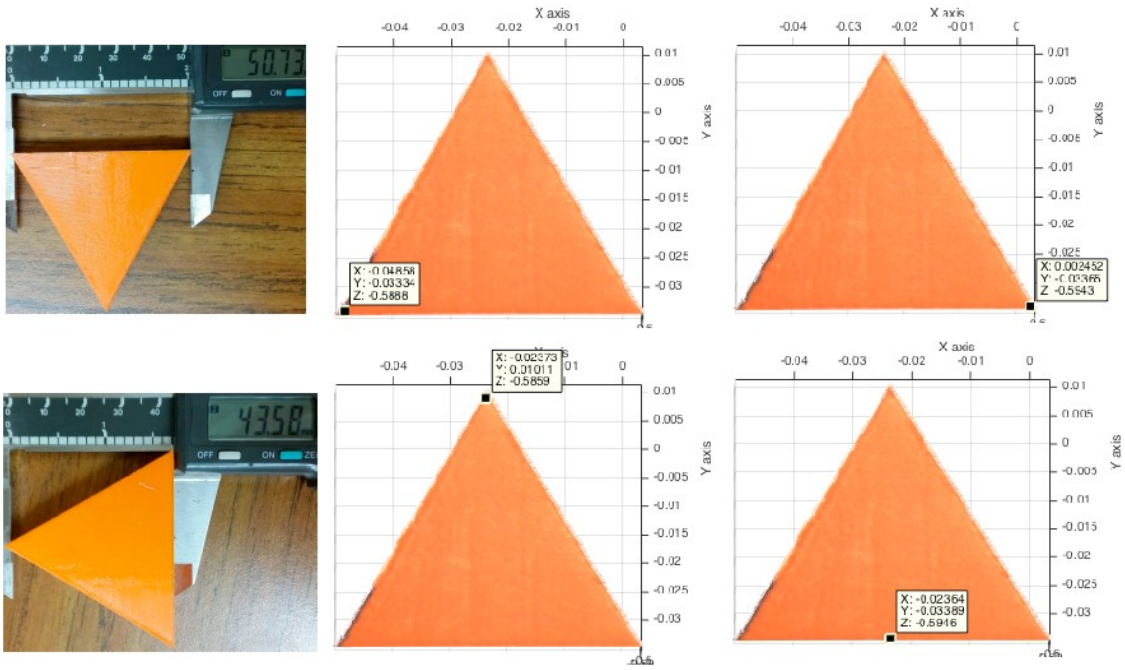


Figure 3.13 Triangle (orange): a) width measurement and b) height measurement

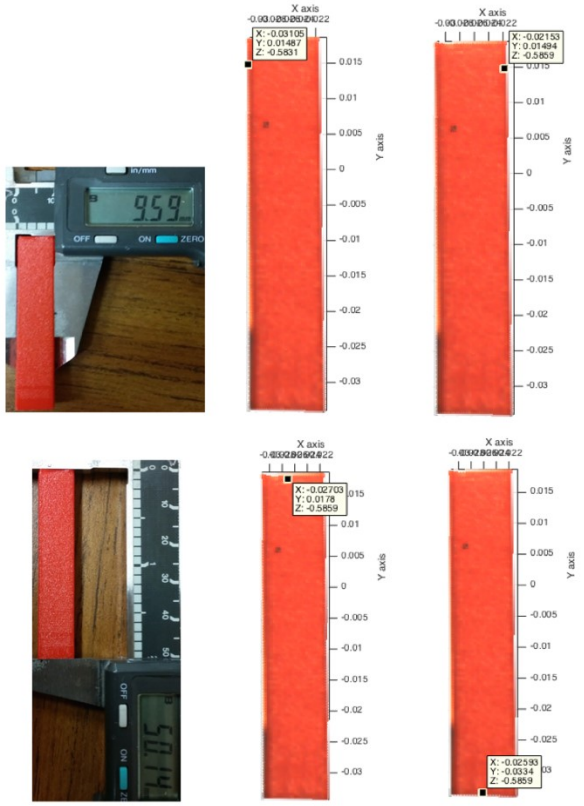


Figure 3.14 Rectangle (red): a) width measurement and b) height measurement.

3.5 Discussion

The experiments demonstrate that both the single and two cameras set up can be used detect a catastrophic FFF 3-D printing error such as clogged filament. Table 3.3 shows the percentage of error for complete printing between single and two camera setups. The size error percentage of two cameras is less than the shape error percentage of single camera. However, the calculation time of two cameras is greater than the single camera. For two cameras set up provided the width and height error. There are more error details for the double camera setup than the single camera provided only the total shape error.

The error detection system works as designed for both the single and two camera setups. To detect an error more accurately, the perspective view of stlimage needs to be set as the actual perspective view between cameras and the 3-D printing object. It should be noted, that a printed 3-D object usually has a small error when compared to its designed 3-D model because of the FFF process that impacts error detection calculation. These experiments show that the shape error detection can determine when the printing has failed because the 3-D printed objects are smaller than the STL models and the error percentage is greater than 5%. However, the error detection system will detect an error better than either process alone, when the single and two camera setups are combined to detect error together. While the 3-D printer is printing, the single camera system detects a shape error every N layers because the computation time is less than 10 seconds for the whole object. If a shape error is greater than 5%, it will report to the user. If there is no shape error, the two cameras system will start to detect a size error. If a size error is greater than 5%, it will report to user. If there is no size error, the 3-D printer will continue. This combined

method provides both the size and shape error detection with required accuracy in reasonable times for FFF printing.

Overall, the combination of the two methods (single and double camera) was found to be the most effective. The use of cameras can be less expensive than other methods used to determine the accuracy of a 3-D print such as a laser scanning or sensor [49]. Using the single camera method, the computation time (6.9 seconds for 9 square cm) is faster than both subtraction (fastest is 10 seconds for 6.25 square cm) and the searching (fastest is 12 seconds for 6.25 square cm) algorithm developed by Hurd et al. [48]. There are other methods to stop catastrophic failures. For example, Barker developed a system that works for delta-style RepRaps, which stops a print when electrical connections are broken if any of the linking rods are thrown [71]. In addition, to the increase in complexity for the 3-D printing system this is also not generalizable to other 3-D printers that do not have magnetic bearings (e.g. most Cartesian based printers). Early work has tried to determine ways to use relatively expensive ultrasonic sensors to detect errors with promise, but unreliable results [62]. This method (100 % detection) can detect an error better than vision based error detection for 3-D printing processes when missing material flow (80% detection) [50]. When the square model is tested printing every 10 layers when the layer height is 0.2 mm, the shape errors are greater than 5% when the nozzle is clogged, or an incomplete project. Using the single camera method can detect an error at 2mm in height which is smaller than 5 mm [51].

Other solutions to 3-D print failure provided in the RepRap community have had video monitor of printing [72], but the user has to stop the print manually if the user detects and error through continuous human surveillance. This obviously undermines one of the

primary benefits of bespoke automated fabrication with 3-D printers because of the necessary human involvement. The system described here overcomes that issue to allow for automatic error detection with no human oversight. However, the algorithm here still has two fundamental limitations. First, the finite (several seconds) of commutation time (as summarized in Table 3.4) does not allow every layer to be monitored in real time for small printed objects as the print speed is faster than the analysis time. For larger more complex prints this is a less of an issue and as the results have shown here sampling a printed object after several layers is adequate for catastrophic failures although it does not enable real time automatic error detection (and the potential for real time error correction). To get to that goal the computation time would need to be reduced. This may be possible by streamlining the computation and removing it from the MATLAB environment. Doing the latter, will also overcome one of the other primary challenges to the use of this method in the distributed manufacturing community. Specifically, although the algorithms provided here are open source [73]. They currently are run in the MATLAB environment which costs $2,150 [74]. This is not that expensive for research or in higher end 3-D printer applications, but represents a barrier to deployment in the low-cost prosumer printers used for distributed manufacturing, which generally cost in total $2,500 or less (the RepRap used in this study was $500 in parts).

In addition, to overcoming these limitations there are several other areas of future research. First, this system would be improved if it was applied to all sides of the printing object. For future research, this error detection system will be implemented and extended from the basic approach into 360 degree around FFF-based 3-D printing. It will improve the object detection capability as there is better understanding for the scene geometry and

therefore for object detection in the depth dimension. Furthermore, to reduce the cost for adding the error detection system to FFF-based 3-D printing, low-cost web cameras will be applied in this system. Using low cost optics will need to be vetted for its effects on the performance of the system and the algorithms presented here.

Table 3.4 Error measurements for complete printing each tested geometry (W: Width, H: Height) of two cameras (size error) and single camera (shape error)

Object		Skull		Cube		Twisted gear vase		Rectangle		Cylinder		Triangle	
Color		Pink		Black		Red		Red		Glow		Orange	
Axes		W	H	W	H	W	H	W	H	W	H	W	H
Size (mm)	STL model	55.13	59.90	30.00	30.00	43.82	38.03	10.00	50.00	48.45	49.00	50.23	43.50
Size error (%)		0.07	0.90	1.80	0.75	0.77	0.79	0.70	2.11	3.94	3.03	0.54	0.96
Calculation time (sec)		73.88		45.66		67.05		38.73		58.78		51.56	
Shape error (%)		2.98		1.34		2.20		0.98		2.58		2.06	
Calculation time (sec)		6.64		6.90		7.07		7.16		9.03		6.56	

3.6 Conclusions

This paper described a low-cost reliable real-time monitoring platform for FFF-based 3-D printing based on a single and two cameras system for a single side. The results showed that both algorithms with a single and double camera system were effective at detecting a clogged nozzle, loss of filament, or an incomplete project for a wide range of 3-D object geometries and filament colors. The error calculation was determined from the difference in shape between stlimage and cameraimage, or the different size between stlimage and the 3-D reconstruction. The error was reported when these errors exceeded 5%. The validity of this approach using experiments shows that the error detection system is capable of a

100 percent detection rate for failure detection. The combined method analyzed here has a better detection rate and a lower cost to previous methods. In addition, this method is generalizable to a wide range of FFF 3-D printer geometries, which enables further adoption of desktop 3-D printing for distributed manufacturing as wasted print time and filament are reduced.

3.7 References

1. Crump, S.S., Stratasys, Inc., 1992. Apparatus and method for creating three-dimensional objects. U.S. Patent 5,121,329.
2. Sells E, Smith Z, Bailard S, Bowyer A, Olliver V (2010) RepRap: The Replicating Rapid Prototyper: Maximizing Customizability by Breeding the Means of Production. In: Piller FT, Tseng MM (eds) Handbook of Research in Mass Customization and Personalization: Strategies and concepts, Vol.1. World Scientific, pp 568-580.
3. Jones R, Haufe P, Sells E, Iravani P, Olliver V, Palmer C, Bowyer A (2011) RepRap – the replicating rapid prototype. Robotica 29:177–191.doi:10.1017/S026357471000069X
4. Bowyer A (2014) 3D printing and humanity's first imperfect replicator. 3D Printing and Additive Manufacturing 1(1): 4-5. doi:10.1089/3dp.2013.0003
5. Gibb, A. and Abadie, S., 2014. Building open source hardware: DIY manufacturing for hackers and makers. Pearson Education.
6. Banzi, M. and Shiloh, M., 2014. Getting Started with Arduino: The Open Source Electronics Prototyping Platform. Maker Media, Inc..
7. Arduino https://www.arduino.cc/ Accessed 10 November 2016
8. Raymond, E. The cathedral and the bazaar. Knowledge, Technology & Policy 1999, 12(3), pp.23–49.
9. Rundle, G. A Revolution in the Making. Simon and Schuster, 2014.
10. Wohlers T. (2016) Wohlers Report 2016. Wohlers Associates, Inc; 2016 Apr 10.
11. Wittbrodt, B.T.; Glover, A.G.; Laureto, J.; Anzalone, G.C.; Oppliger, D.; Irwin, J.L.; Pearce, J.M. Life-cycle economic analysis of distributed manufacturing with open-source 3-D printers. Mechatronics 2013, 23(6), pp. 713-726.
12. Pearce, J.M. Morris Blair, C. , Laciak, K. J., Andrews, R., A. Nosrat and I. Zelenika-Zovko, "3-D Printing of Open Source Appropriate Technologies for Self-Directed Sustainable Development", Journal of Sustainable Development 3(4), pp. 17-29 (2010).
13. Fox, S., After the factory [Manufacturing renewal]. Engineering & Technology, 5(8), pp.59-61 (2010).

14. Pearce, J.M.(2015). Applications of Open Source 3-D Printing on Small Farms. Organic Farming 1(1), 19-35. DOI:10.12924/of2015.01010019
15. Kentzer, J., Koch, B., Thiim, M., Jones, R.W. and Villumsen, E., 2011, May. An open source hardware-based mechatronics project: The replicating rapid 3-D printer. In Mechatronics (ICOM), 2011 4th International Conference On (pp. 1-8). IEEE.
16. Irwin, J.L. Oppliger, D.E. Pearce, J.M. Anzalone, G. Evaluation of RepRap 3D Printer Workshops in K-12 STEM. 122nd ASEE 122nd ASEE Conf. Proceedings, paper ID#12036, 2015.
17. Gonzalez-Gomez, J., Valero-Gomez, A., Prieto-Moreno, A. and Abderrahim, M., 2012. A new open source 3d-printable mobile robotic platform for education. In Advances in autonomous mini robots (pp. 49-62). Springer Berlin Heidelberg.
18. Grujović, N., Radović, M., Kanjevac, V., Borota, J., Grujović, G. and Divac, D., 2011, September. 3D printing technology in education environment. In 34th International Conference on Production Engineering (pp. 29-30).
19. Schelly, C., Anzalone, G., Wijnen, B. and Pearce, J.M., 2015. Open-source 3-D printing technologies for education: Bringing additive manufacturing to the classroom. Journal of Visual Languages & Computing, 28, pp.226-237.
20. Campbell, I., Bourell, D. and Gibson, I., 2012. Additive manufacturing: rapid prototyping comes of age. Rapid Prototyping Journal, 18(4), pp.255-258.
21. Gibson, I., Rosen, D. and Stucker, B., 2014. Additive manufacturing technologies: 3D printing, rapid prototyping, and direct digital manufacturing. Springer.
22. O'Neill, P.F., Azouz, A.B., Vazquez, M., Liu, J., Marczak, S., Slouka, Z., Chang, H.C., Diamond, D. and Brabazon, D., 2014. Advances in three-dimensional rapid prototyping of microfluidic devices for biological applications. Biomicrofluidics, 8(5), p.052112.
23. Pearce, J.M., Anzalone, N.C. and Heldt, C.L., Open-source Wax RepRap 3-D Printer for Rapid Prototyping Paper-Based Microfluidics, Journal of Laboratory Automation 21(4) 510–516 (2016).
24. Rimock, M., 2015. An Introduction to the Intellectual Property Law Implications of 3D Printing. Canadian Journal of Law and Technology, 13(1).
25. Laplume, A., Anzalone, G.C. and Pearce, J.M. Open-source, self-replicating 3-D printer factory for small-business manufacturing. The International Journal of Advanced Manufacturing Technology. 85(1), pp 633-642 (2016). doi:10.1007/ s00170-015-7970-9
26. Tech, R.P., Ferdinand, J.P. and Dopfer, M., 2016. Open Source Hardware Startups and Their Communities. In The Decentralized and Networked Future of Value Creation (pp. 129-145). Springer International Publishing.
27. Troxler, P. and van Woensel, C., 2016. How Will Society Adopt 3D Printing?. In 3D Printing (pp. 183-212). TMC Asser Press.
28. Pearce, J. M. 2012. Building Research Equipment with Free, Open-Source Hardware. Science 337 (6100): 1303–1304. DOI: 10.1126/science.1228183

29. Pearce, J.M. Open-Source Lab: How to Build Your Own Hardware and Reduce Research Costs, Elsevier, 2014.
30. Baden, T., Chagas, A. M., Gage, G., Marzullo, T., Prieto-Godino, L. L., & Euler, T. (2015). Open Labware: 3-D Printing Your Own Lab Equipment. PLOS Biology, 13(3). DOI: 10.1371/journal.pbio.1002086
31. Coakley, M. and Hurt, D.E., 2016. 3D Printing in the Laboratory Maximize Time and Funds with Customized and Open-Source Labware. Journal of Laboratory Automation, p.2211068216649578.
32. Kłodowski, A., Eskelinen, H. and Semken, S., 2015. Leakage-proof nozzle design for RepRap community 3D printer. Robotica, 33(04), pp.721-746.
33. Mercuri, R. and Meredith, K., 2014, March. An educational venture into 3D Printing. In Integrated STEM Education Conference (ISEC), 2014 IEEE (pp. 1-6). IEEE.
34. Chonga, S., Chiub, H.L., Liaob, Y.C., Hungc, S.T. and Pand, G.T., 2015. Cradle to Cradle® Design for 3D Printing. CHEMICAL ENGINEERING, 45.
35. Moilanen, J. and Vadén, T., 2013. 3D printing community and emerging practices of peer production. First Monday, 18(8).
36. Frauenfelder, M., Make: Ultimate Guide to 3D Printing 2014: Maker Media. Inc., O'Reilly, Sepaspol CA, 2013.
37. Alastair J. (2016) presented 16 common 3D Printing Problems and Solutions publishing all3dpweb. https://all3dp.com/common-3d-printing-problems-and-their-solutions/ Accessed 10 November 2016
38. Kreiger, M. and Pearce, J.M., 2013. Environmental life cycle analysis of distributed three-dimensional printing and conventional manufacturing of polymer products. ACS Sustainable Chemistry & Engineering, 1(12), pp.1511-1519.
39. Vera, J., 2010. Promoting Tools that integrate LCA into the Product Design Process: a Case Study in Ontario.
40. Kreiger, M. and Pearce, J.M., 2013. Environmental impacts of distributed manufacturing from 3-D printing of polymer components and products. In MRS Proceedings (Vol. 1492, pp. 85-90). Cambridge University Press.
41. Kostakis, V., Roos, A. and Bauwens, M., 2016. Towards a political ecology of the digital economy: Socio-environmental implications of two competing value models. Environmental Innovation and Societal Transitions, 18, pp.82-100.
42. Bonvoisin, J., 2016. Implications of Open Source Design for Sustainability. In Sustainable Design and Manufacturing 2016 (pp. 49-59). Springer International Publishing.
43. Kleszczynski, S., Zur Jacobsmühlen, J., Sehrt, J.T. and Witt, G., 2012, August. Error detection in laser beam melting systems by high resolution imaging. In Proceedings of the Solid Freeform Fabrication Symposium.
44. zur Jacobsmühlen, J., Kleszczynski, S., Schneider, D. and Witt, G., "High resolution imaging for inspection of laser beam melting systems." 2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC). IEEE, 2013.

45. zur Jacobsmühlen, J., Kleszczynski, S., Witt, G. and Merhof, D., "Robustness analysis of imaging system for inspection of laser beam melting systems." Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA). IEEE, 2014.
46. Kleszczynski, S., zur Jacobsmühlen, J., Reinarz, B., Sehrt, J.T., Witt, G. and Merhof, D.,"Improving process stability of laser beam melting systems." Proceedings of the Frauenhofer Direct Digital Manufacturing Conference. 2014.
47. Concept Laser http://www.conceptlaserinc.com/ Accessed 10 November 2016
48. Hurd, Sam, Carmen Camp, and Jules White. "Quality Assurance in Additive Manufacturing Through Mobile Computing." International Conference on Mobile Computing, Applications, and Services. Springer International Publishing, 2015.
49. Faes, M., Abbeloos, W., Vogeler, F., Valkenaers, H., Coppens, K. and Ferraris, E., 2014, September. Process monitoring of extrusion based 3D printing via laser scanning. In PMI 2014 Conference Proceedings (Vol. 6, pp. 363-367).
50. Baumann, Felix, and Dieter Roller. "Vision based error detection for 3D printing processes." MATEC Web of Conferences. Vol. 59. EDP Sciences, 2016.
51. Ceruti, Alessandro, Alfredo Liverani, and Tiziano Bombardi. "Augmented vision and interactive monitoring in 3D printing process." International Journal on Interactive Design and Manufacturing (IJIDeM) (2016): 1-11.
52. Straub, J. "Initial Work on the Characterization of Additive Manufacturing (3D Printing) Using Software Image Analysis." Machines 3.2 (2015): 55-71.
53. Straub, J. "Characterization of internal geometry/overed surface defects with a visible light sensing system." SPIE Commercial+ Scientific Sensing and Imaging. International Society for Optics and Photonics, 2016.
54. Opencv http://opencv.org/ Accessed 10 November 2016
55. Python software foundation [US] https://www.python.org/ Accessed 10 November 2016
56. ARTOOLKIT https://artoolkit.org/ Accessed 10 November 2016
57. Bay, H., Ess, A., Tuytelaars, T. and Van Gool, L., "Speeded-up robust features (SURF)." Computer vision and image understanding 10.3 (2008): 346-359.
58. Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." Communications of the ACM 24.6 (1981): 381-395.
59. Microsoft https://msdn.microsoft.com/ Accessed 13 November 2016
60. RaspberryPi https://www.raspberrypi.org/ Accessed 10 November 2016
61. Hu, F., Li, L., Liu, Y. and Yan, D., “Enhancement of Agility in Small-Lot Production Environment Using 3D Printer, Industrial Robot and Machine Vision." doi: 0.5013/ IJSSST.a.17
62. Cummings I, Hillstrom E, Newton R, Flynn E, Wachtor A. "In-Process Ultrasonic Inspection of Additive Manufactured Parts." Topics in Modal Analysis & Testing, Volume 10. Springer International Publishing, 2016. 235-247.
63. Nuchitprasitchai, S., Roggemann, M.C.and Havens, T.C. "Algorithm for Reconstructing Three Dimensional Images from Overlapping Two-Dimensional

Intensity Measurements with Relaxed Camera Positioning Requirements to reconstruct 3D image." IJMER 6.9 (2016): 69-81.

64. Appropedia. Delta Build Overview:MOST publishing appropediaweb. http://www.appropedia.org/Delta_Build_Overview:MOST Accessed 13 June 2016
65. RepRap. http://reprap.org/ Accessed 13 June 2016
66. Johann . Rostock http://reprap.org/wiki/Rostock Accessed 5 November 2016
67. Nuchitprasitchai 3-D models https://osf.io/6rfky/ 2016-11-17
68. Pau M. (2015) presented stlTools publishing mathworksweb. https://www.mathworks.com/matlabcentral/fileexchange/51200-stltools Accessed 1 October 2016
69. Camera Calibration publishing mathworksweb. https://www.mathworks.com/help/releases/R2013b/vision/ug/find-camera-parameters-with-the-camera-calibrator.html Accessed 15 October 2016
70. Lowe, D.G., 1999. Object recognition from local scale-invariant features. In Computer vision, 1999. The proceedings of the seventh IEEE international conference on (Vol. 2, pp. 1150-1157).
71. Barker, B. Thrown Rod Halt Mod. Published on Appropedia.org http://www.appropedia.org/Thrown_Rod_Halt_Mod Accessed 5 November 2016
72. David G. (2016) presented adding a Raspberry Pi case and a camera to your LulzBot Mini publishing kupoos Web. http://www.kupoos.com/video/q7oqOPzCHYE/adding-a-raspberry-pi-case-and-a-camera-to-your-lulzbot-mini/ Accessed 20 November 2016
73. Real Time Optical Monitoring of Fused Filament 3-D Printing. Published on the Open Science Framework. https://osf.io/hwdzm/ Accessed 8 December 2016.
74. Mathworks, Pricing and Licensing https://www.mathworks.com/pricing-licensing/index.html?intendeduse=comm Accessed 8 December 2016.

Chapter 4: An Open Source Algorithm for Reconstructing 2-D Images of 3-D Objects being Fabricated for Low-cost, Reliable Real-Time Monitoring of FFF-Based 3-D Printing[3]

4.1 Abstract

Although the open source nature of self-replicating rapid prototoyper (RepRap) 3-D printers have enabled the democratization of additive manufacturing, these 3-D printers are still challenged to meet reliability standards. Relatively little work has investigated error detection in such prosumer desktop 3-D printing. In this study an open source low-cost reliable real-time optimal monitoring platform for 3-D printing is presented with a goal of reducing errors below 10% between reconstructed images and the 3-D printed object. This error detection system is implemented with low-cost web cameras from three different perspectives (providing 360 degrees of coverage) by extending an algorithm previously described for a single camera. The algorithm is now developed using open-source Python to reduce the cost and computation time. The results show that the algorithm was 100% effective at detecting a clogged nozzle, loss of filament, or an incomplete print for a wide range of 3-D object geometries. Error calculations were determined from the difference in shape between the rendering of the 3-D design and the camera image of the print and a 7% difference was found to be an accurate threshold for error detection. The validity of this approach using experiments shows that the error detection system is capable of a 100% rate for failure detection, which is a better detection rate at a lower cost than previous real-time monitoring methods. In addition, this method is generalizable to a wide range of fused

[3] This chapter has been completed as an article to submit. Citation: Nuchitprasitchai S, Roggemann M, & Pearce J (2017). An Open Source Algorithm for Reconstructing 2-D Images of 3-D Objects being Fabricated for Low-cost, Reliable Real-Time Monitoring of FFF-Based 3-D Printing

filament 3-D printer geometries, which enables further adoption of desktop 3-D printing for distributed manufacturing.

4.2 Introduction

With the development of the Arduino electronic prototyping platform [1-3] the self-replicating rapid prototoyper (RepRap) [4-6] was developed following open source hardware design principles [7] formulated in the software industry [8]. The free and open source nature of the RepRap 3-D printer quickly led to cost declines [9,10] and now the cost of open source RepRap style 3-D printers using fused filament fabrication (FFF) are under US$100 in parts. RepRaps can produce their own components [4-6] and can self-upgrade, which assists in rapid technical growth [11-12]. At the same time the open nature makes them accessible to many fields including appropriate technology and sustainable development [13-15], education [16-20], rapid prototyping [21-22], microfluidics [23,24], decentralized manufacturing [25-28], and bespoke scientific equipment [29-32]. However, these low-cost 3-D printers are still challenged to meet reliability standards due to common problems including: warping, elephant foot at the base of a print, bed adhesion, lower parts shrinking, skewed prints and shifted layers, layer misalignment, missing layers, cracks in tall objects, pillowing, stringing, under-extrusion, over-extrusion, gaps in the top layers, or lack of filament exiting the nozzle [33-38]. These failures cost money and printer operator time, and waste resources (polymers and energy), which detract from the environmental and sustainability benefits of distributed manufacturing with 3-D printing [39-43]. Over the last several years, many researchers have improved these issues using automatic error detection, however, the majority of this work has been on high-cost and high resolution laser-based 3-D printing systems [44-47].

Relatively little work has investigated error detection in prosumer desktop FFF-based 3-D printing. Some research in this area has used laser sensors to detect an error [48-50]. Faes et al. had nearly zero production failure in the z-direction to detect the deposited tracks and to determine the dimension of interest in a closed-loop feedback in an extrusion based 3-D printing with a modular 2-D laser triangulation scanner [48]. Volpato et al. could reduce the error in z-axis up to 50 % of the surface quality of the support base in the trademarked version of FFF (fused deposition modeling (FDM)) by using a Roland MDX-40 milling machine attached to a piezoelectric sensor [49]. Haixi et al. successfully applied the relationship between the machine conditions and the features of acoustic emission (AE) system to detect the extruder in different conditions (normal, semi-blocked, and blocked) for a non-intrusive condition monitoring of HYREL3-D [50]. Other works have used cameras or webcams to monitor 3-D printers. Hurd et al. successfully applied a Samsung Galaxy Tab 3 to remotely monitor printing errors [51], though it can detect only the top side of the printed object. Baumann, et al. [52] used a PlayStation eye cam to detect errors including detachment, missing material flow and deformed objects with OpenCV [53] and Python [54]. But this work can detect only the front side of the printed object and it could only detect failed printing up to 80% of the time. Straub successfully applied a visible light 3-D scanning system, five Raspberry Pi cameras, Raspberry Pi [55], and open source software approach with C# and Dot Net Framework [56] to detect incomplete prints [57]. However, this work cannot detect errors that occur in the horizontal axis. Other solutions to detect failure in RepRap 3-D printers have had video monitoring of printing [58-62], but the user has to watch the video of the print and end the print manually.

To monitor a 3-D printing error around FFF-based 3-D printing, an open source low-cost reliable real-time optimal monitoring platform for FFF-based 3-D printing is presented here with a goal of reducing errors between reconstructed images and the 3-D printed object below 10% while guaranteeing 100% print error detection. This error detection system is implemented with low-cost web cameras and extended from the basic approaches discussed above into 360 degrees around the printed object from three different perspectives by extending the algorithm previously described for the single camera setup [63]. The algorithm is developed using open-source Python and run on Raspberry Pi3 to reduce the cost and the computation time. For 3-D printing monitoring in three different perspectives, the single camera setup is tested with four different 3-D object geometries. The results are compared between the normal and the failure states as well as the results of previously published techniques. The limitations of this approached are detailed and future work is described and conclusions are drawn.

4.3 Method

For this study, optical experiments were set up around a delta-style RepRap as shown in Figure 4.1 running a single camera detection algorithm [64]. The single camera error detection from a given perspective uses the images from three cameras in different views to calculate three 2-D shape images. A Python algorithm was written for the experimental setup and is made available free and open source under an AGPLv3 license [65,66]. Due to the distance between the camera and the printer for the experiment setup, the field of view for all cameras can cover the printed area of 70 mm in width and 60 mm in height. The optical parameters of the camera system need to be known for using a camera

calibration technique [65] in order to calculate the intrinsic and extrinsic parameters for a specific camera setup. These parameters are used to correct for lens distortion and image rectification. A low-cost (<US$500 in parts) [16] open source delta-style polymer printing RepRap (MOST Delta) is used [67]. The MOST Delta is a RepRap [68] derived from the Rostock printer with a cylindrical build volume 270 mm in diameter and 250 mm high and overall dimensions of 375 mm diameter and 620 mm high [67,68]. Three LED light sources [69] are installed on the three sides of the printer. All light sources are connected to the circuit (Figure 4.2) with 4 volts from a DC power supply. Each camera is setup on the same side of LED light sources. All cameras are connected to a 7 port USB 3.0 hub with 12V/3A power adapter which is turn connected to Raspberry Pi3 [70]. The cameras used in this study are three Logitech C525 webcams, with an image size of 480-by-640 (height-by-width).

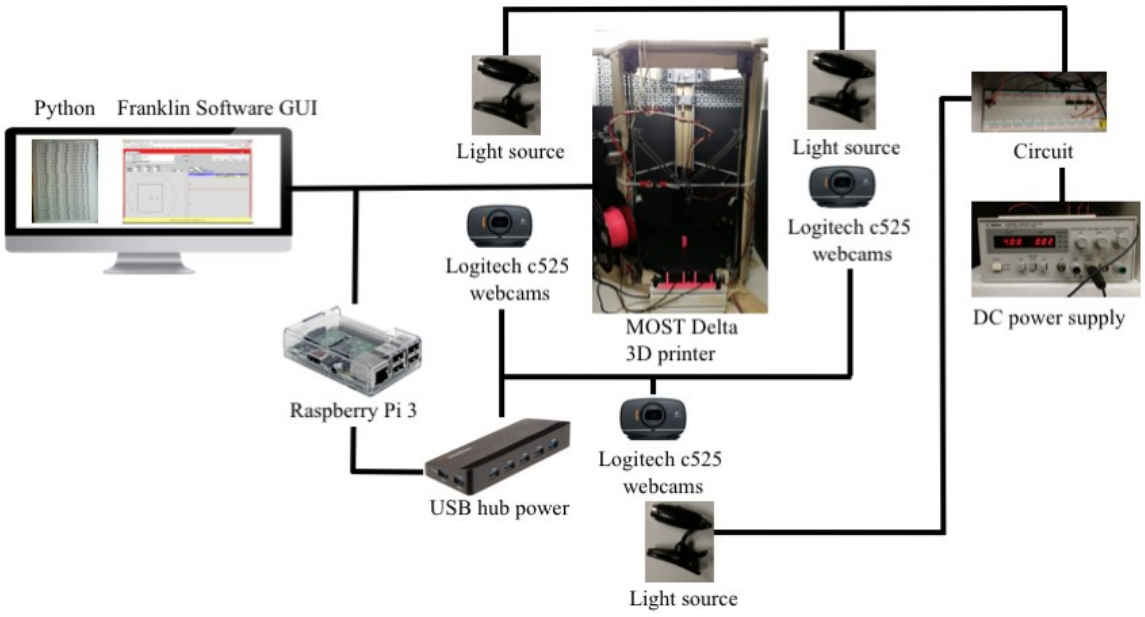


Figure 4.1 MOST Delta printer experiment setup

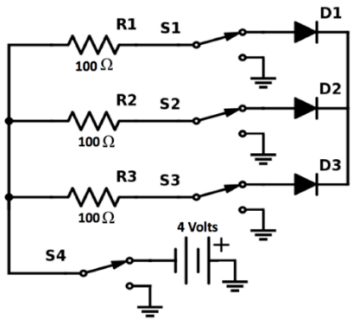


Figure 4.2 Light source circuit

There are three steps to prepare the error detection system before printing a 3-D model: 1) camera calibration, 2) preparing STereoLithography (STL) files and resultant images, and 3) setting up a pause and loop to move the extruder out of the field of view during imaging. STL [71] is a file format describing a 3-D model by using a series of connected triangles to create the surface of the model and it is usually generated by computer aided design (CAD) software. The first step is camera calibration. Sixteen chessboard images are taken from three different view of the cameras after the MOST Delta printer experiment is setup for camera calibration. There are three cameras named as camera0, camera1, and camera2. The sixteen chessboard images are taken in different perspectives, calculated, and saved as calibrationdata1, calibrationdata2, and calibrationdata3. The second step is preparing an stlimage by slicing stl files into every N layers where the error will be detected as shown in Figure 4.3. The layer height and the amount of slicing layers need to be assigned for slicing the stl file in three different views of the cameras. The layer height and the amount of total layers can be found in the gcode file. All data at every N layers from the stl file are plotted in the x, y, z axes to display the shape of the rendered 3-D model, which can be observed from different viewpoints. Thus, the shape of the stlimage is saved as PNG image type on the xz-plane. If the modulus after division between the total height and the height

of every N layers is not equal to zero, the last PNG files are named as the amount of total layers. For example, if the 3-D model in gcode has 129 total layers, layer height of 0.2 mm, and the 3-D model is sliced in every 30 layers, then the stl file is sliced at layer 30, 60, 90, 120, and 129 which result in heights of 6, 12,18, 24, and 25.8 mm, respectively. The first STL slicing files are saved as SCAD30_1.png, SCAD30_2.png, and SCAD30_3.png, the next slicing files are saved as SCAD60_1.png, SCAD60_2.png, and SCAD60_3.png, and so on. Slicing the stl files for four model found that three stl files can start slicing at every layers 10, 20, or 30, but the t55gear stl file can start slicing at every 30 layers. Therefore, this study uses six images every 30 layers. The last step in the process involves setting up a pause and a loop to move the extruder out of the images every N layers in order to eliminate visual noise in the object images. The extruder movement is paused and moved to the certain height out of the field of view of all cameras. The 3-D model is designed in OpenSCAD version2015.03-3 [72] and it is rendered and saved into an stl file. After the 3-D model stl file is opened in Cura version15.04.6 [73], the 3-D model is saved as a gcode file. The 3-D model gcode file is opened by any text editor program to add the extra code in every N layers as shown in Figure 4.4.

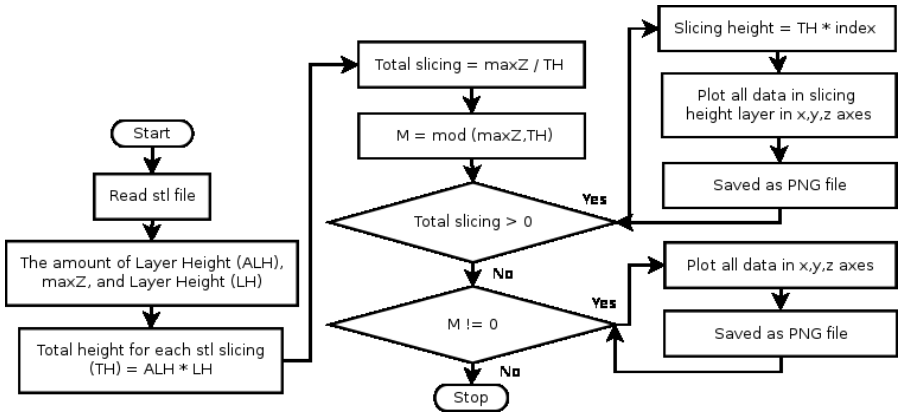


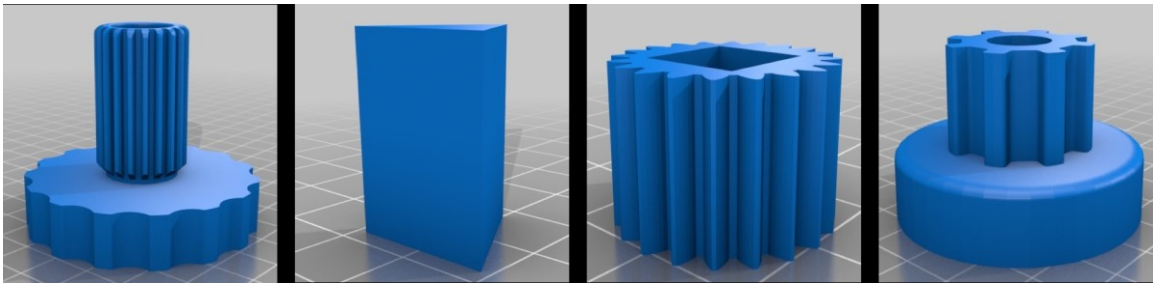
Figure 4.3 Slicing stl file flowchart

```
; MOVE Extruder for taking photo
;

G91                     ;relative positioning
G1 E-10 F2000           ;retract the filament 10 mm before lifting the nozzle, to release some of the pressure
G1 Z+100  X-20 Y-20     ;move Z 100 mm up and retract filament even more
G4 P30000               ; wait 30 secs for nozzle length to stabilize
G90                     ;absolute positioning
```

Figure 4.4 Gcode for pausing and moving the extruder to take the images

The 3-D printing models chosen (Figure 4.5) after the preparing stlimages step are: sun gear [74], prism, gear [75], and t55gear [76], which are freely available [77]. The printing parameters used are: layer height 0.2 mm, shell thickness 1 mm, enable retraction, bottom/top thickness 1mm, fill density 20%, print speed 60 mm/s, printing temperature 180oC, diameter filament 1.74 - 1.78 mm, flow filament 100%, and nozzle size 0.5 mm. The PLA filament used in this experiment is Hatchbox 3-D PLA (pink) with dimensional accuracy +/- 0.05 mm on 1 kg spools, 1.75 mm diameter.



a) sun gear b) prism c) gear d) t55gear

Figure 4.5 Rendering of STL models for testing

The error detection algorithm, written in Python, will alert users if the error is greater than 7% for single camera setup as shown in Figure 4.6. After the user orders printing a 3-D model through Franklin [78] with the amount of slicing layer number (N), the background images are taken before printing the 3-D model. The background images are taken from three cameras saved as bg1, bg2, and bg3, where the number 1, 2, and 3 mean the first, the second and the third cameras. At every N layers, the printer is paused to detect an error.

After the extruder is moved up 100 mm from the current height, the object images are taken and from three cameras saved as obj1, obj2, and obj3. In the input image preparation process, the object images have the background removed, rendered black between bg and obj images for each camera, and saved as new.png. It should be noted there can be a light reflection of the object on the substrate in the images that may cause an error. The new.png from the previous error detection will be used in the next error detection to create the new images named as newimg.png. For an example, if the current layer is the same as the amount of slicing layer number, the images after removing background are saved into two different file names as new and prev. If they are not equal, they are saved as newimg. The prev images is needed for the next step to improve the process of removing the background. If the current layer is greater than the slicing layer number, the prev image is read to combine the interested object area between the prev and the newimg images into two different file names as new and prev. Distortion is removed from the image by intrinsic parameters from camera calibration [79]. Next, a region of interest (ROI) is calculated from the image by converting the color image into a gray scale image, then converting it into a binary image. The object area in the binary image is converted to be white to be used as the ROI, otherwise it is converted to be black. The stlimages are read and resized to the same size as the cameraimages. The shape of the object in stlimage and cameraimage are defined by edge detection, then the object size ratio between these two images can be found for rescaling. After rescaling, edge detection is applied again to find min and max positions of the object in both images for rectification. After rectification, any errors in the process are detected by subtracting the shape between the stlimage and cameraimage. The error detection is calculated for all three images at the same time. If only one of them has the

difference of subtraction that is greater than 7%, there is an error; otherwise there is no error flagged and 3-D printing will continue. The two experiments tested by the single cameras setup are in the normal and the failure state. In the normal state the filament is in normal condition to completely print the 3-D object. In the failure state the printing is manually impeded by the experimenter to simulate a failure that would prevent printing the 3-D object.

After starting printing the 3-D model, all three background images were taken from three cameras in three different views. The filament was in normal condition to complete printing the 3-D object. After the extruder was paused and moved up for 100 mm at every 30 layers, the three object images from three cameras in three different perspectives was taken. The single camera error detection is calculated by absolute value of subtracting between the three 2-D shape images and the stlimages as explained in Equation (4.1).

$$Error\ (\%) = \left(\frac{\sum(\ |CameraImage - STLImage|)}{\sum(STLImage)}\right) * 100 \tag{4.1}$$

4.4 Results

The experimental procedures are tested in normal and failure states for the single camera setup with different object geometries (sun gear, prism, gear, and T55gear). In order to eliminate the background noise from the extruder, the images were taken after pausing printing and the extruder was moved from all three camera views. The full model image results for four different 3-D object geometries from three different perspectives are shown in Figure 4.7.

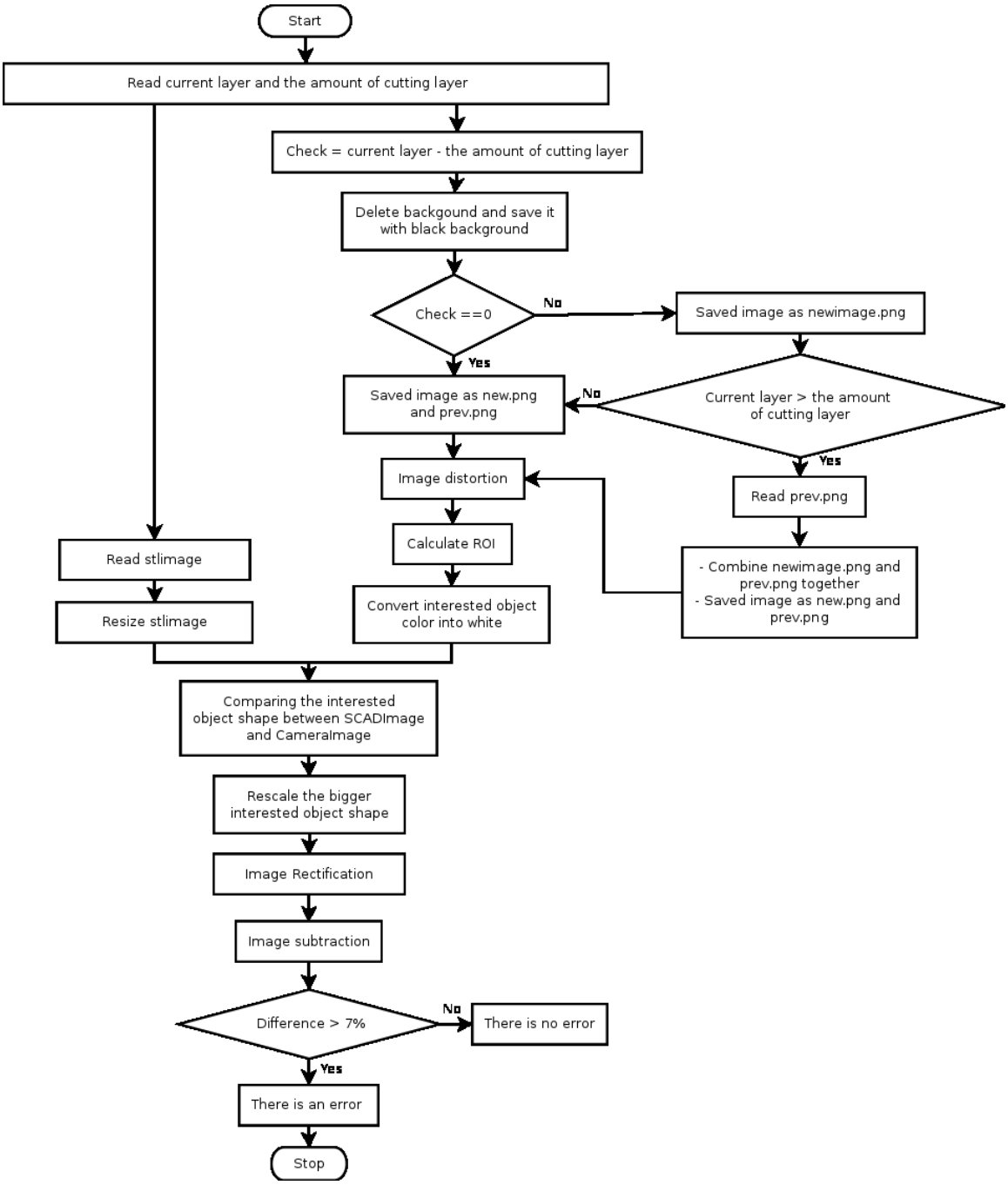
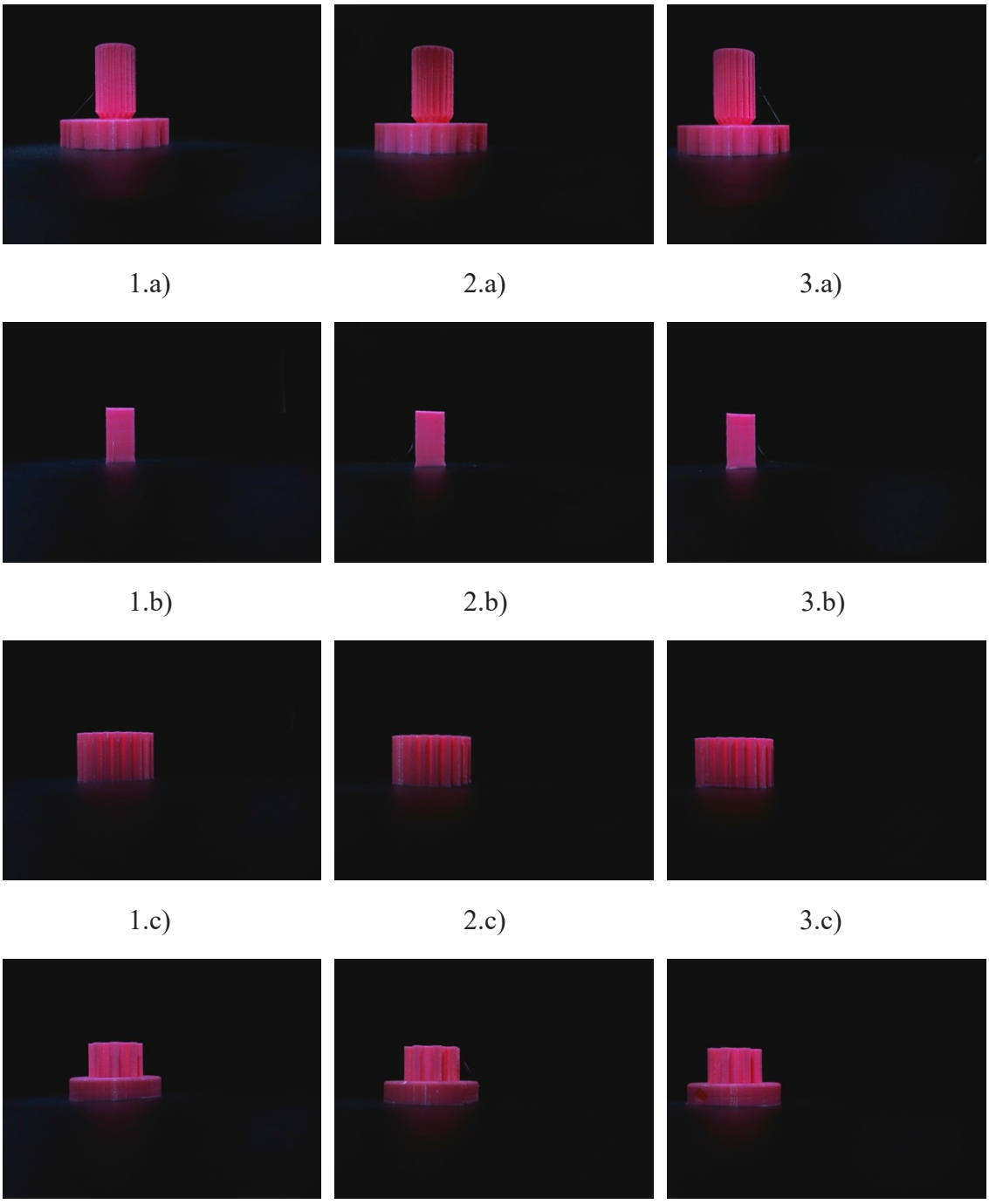


Figure 4.6 Single camera error detection system flowchart



1.a) 2.a) 3.a)

1.b) 2.b) 3.b)

1.c) 2.c) 3.c)

1.d) 2.d) 3.d)

Figure 4.7 Full model from 1st, 2nd and 3rd camera: a) sun gear, b) prism, c) gear, and d) t55gear

4.4.1 The Normal State of Filament Condition

The graph of the error detection percentage for the single camera setup for all four geometries is shown in Figure 4.8. Figure 4.8 shows that the shape errors are less than 7 % for each geometry. This meets the design goal of less than 10% error for shape reconstruction. This error is acceptable because the error of shape difference is less than 7%. The computation time for all images from three perspectives are fast as they are less than 11 second to detect an error as shown in Figure 4.9.

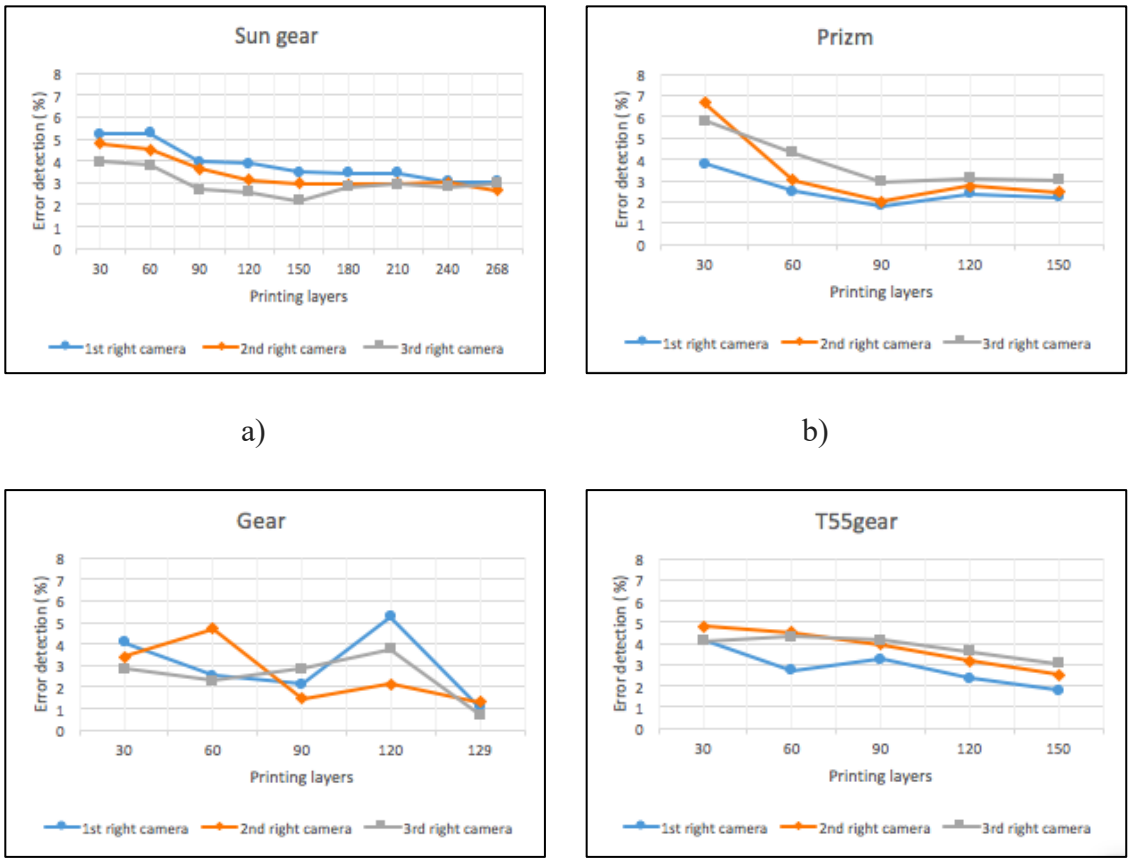


Figure 4.8 The error detection (%) of normal state: a) sun gear, b) prism, c) gear, and d) t55gear.

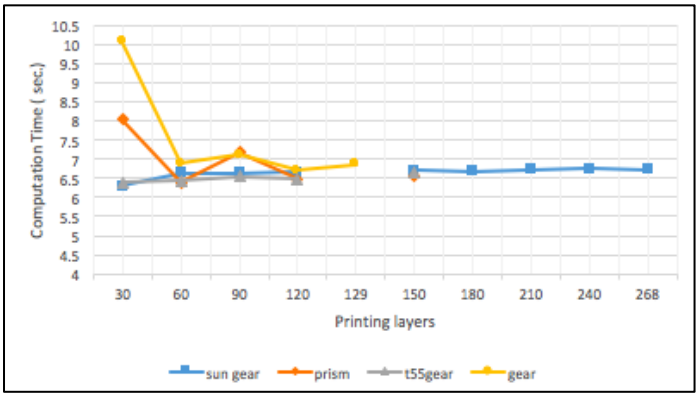


Figure 4.9 The computation time of normal state for four models

4.4.2 The Failure State of Filament Condition

The failure state has been tested every 30 layers with different geometries between the cameraimage and stlimage in different conditions. Figure 4.10 shows that the shape errors are greater than 7% for each geometry. When the nozzle is clogged, or an incomplete print is caused by filament running out that effect the 3-D printing shapes so they are smaller than the SCAD models. The computation time for all images from three perspectives are fast and less than 9 second to detect an error as shown in Figure 4.11.

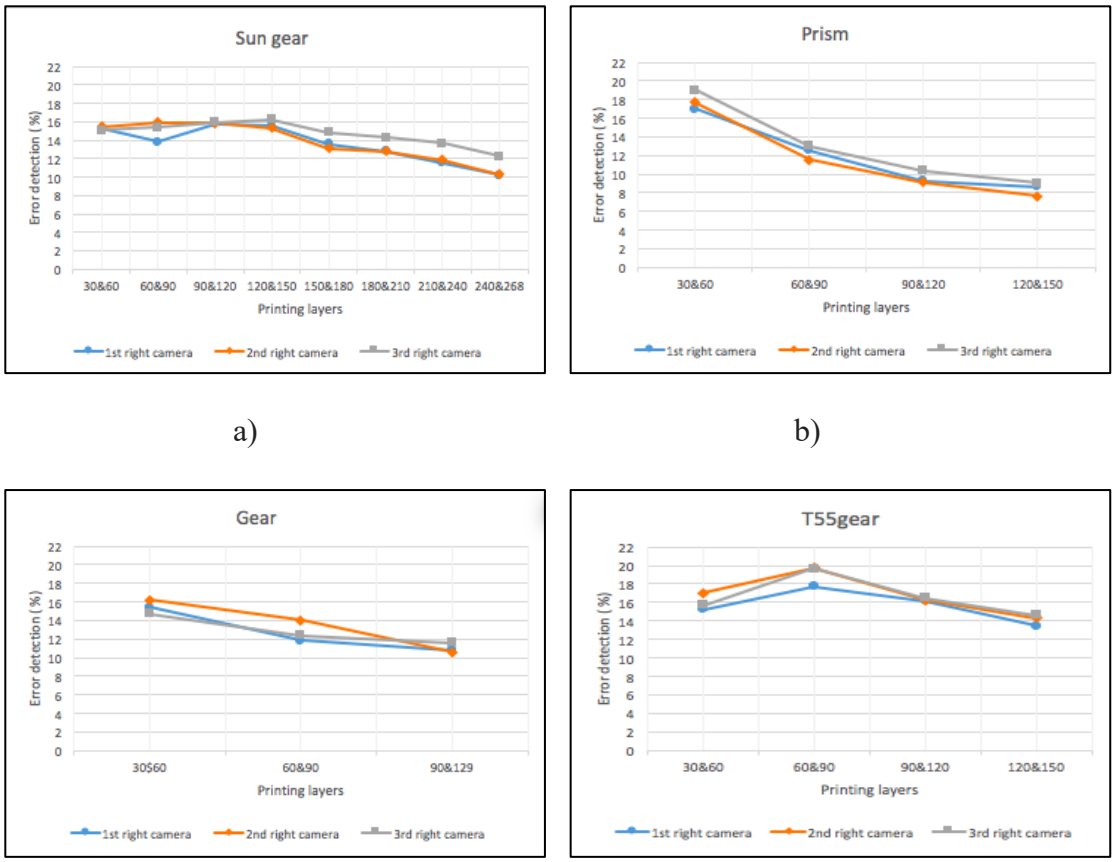


Figure 4.10 The error detection (%) of failure state: a) sun gear, b) prism, c) gear, and d) t55gear.

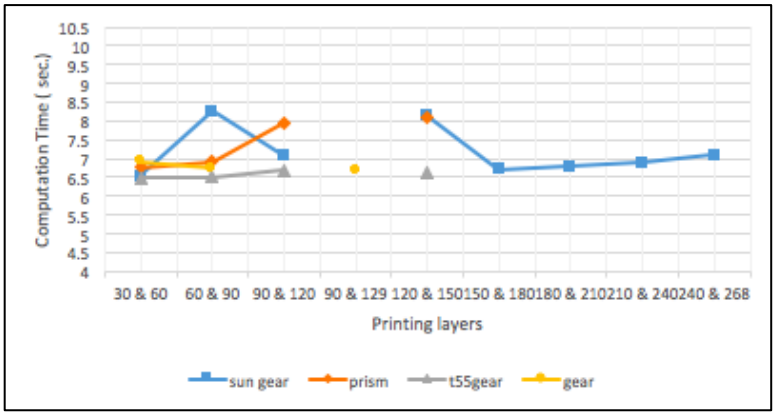


Figure 4.11 The computation time of failure state for four models

4.5 Discussion

The experimental results show that the three-camera setup in Python can be used to automatically detect a 3-D printer error such as clogged extruder, loss of filament, or an incomplete project for a wide range of 3-D object geometries. These errors can be significant a new user attempting RepRap printing can have a 20% failure rate [12]. Previous solutions depended on either continuous observation of the printer or proprietary software and expensive hardware. This work has overcome these limitations [65] by reducing the computation time for multiple cameras and reducing the cost of software to zero. The computation time here using Python is 3X faster and less expensive than the code [65] with the same algorithm run in the MATLAB environment, which costs $2,150 [80]. Although, this is not that expensive for research or in higher-end 3-D printer applications, it represents a barrier to deployment in the low-cost prosumer printers used for distributed manufacturing, which generally cost in total $2,500 or less (the RepRap used in this study was $500 in parts).

The single camera error detection works as designed. It should be noted, that a printed 3-D object usually has a small error when the 3-D model file is compared to the real 3-D printed object. These experiments show that the shape error detection can detect when the printing has failed because the 3-D printed objects are smaller than the SCAD models and the error percentages are greater than 7%. The use of web cameras can be less expensive than other methods that have more accurate error detection of a 3-D print such as a laser scanning or sensor [48], or science research-grade cameras [64-65]. Using the single camera method, the computation time for all three cameras for each model is faster than both subtraction (fastest is 10 sec. for 6.25 square cm) and the searching algorithm

developed by Hurd et al. (fastest is 12 sec. for 6.25 square cm) [48]. There are other methods to stop catastrophic failures. For example, Nuchitprasitchai developed a rod alarm system for delta-style RepRaps, which alerts users when electrical connections are broken if any of the linking rods lose connection with the end effector (hot end) [81]. The raspberry Pi and the raspberry Pi camera has also been installed on the delta-style RepRaps to remotely monitor the printer manually [82]. Barker also developed a thrown rod halt mod system for delta-style RepRaps, which stops a print when electrical connections are broken if any of the linking rods are thrown [83]. This new method presented here with 100 % detection can detect an error better than vision based error detection for 3-D printing processes when missing material flow (80% detection) [52].

Other solutions to detect the failure 3-D printing in the Reprap 3-D printer have had a video monitor of printing [58-62], but the user has to watch the video and stop the print manually. The error detection system here overcomes this issues by enabling the printer to automatically stopping printing without human oversight. However, the algorithm here still has limitations. First, slicing the stl model into every N layers cannot be done for some number of layers that user wants because Slic3r reported an error about removing a facet on a specific 3-D model. For example, the t55gear model used here could not be sliced into every 10 or 20 layers, which is why in this study it is sliced every 30 layers. Second, this method does not work for 3-D printing models that create too many shadows. In the background removal process, such models lose a lot of data from the bottom of the interested object in the image, which causes a faulty error detection. It should be noted before buying inexpensive web cameras for this application, the focal

length should be at least10 cm as it need to support the open-source environment discussed here.

In addition, to overcoming these limitations there are several other areas of future research. First, the slicing stl model process needs to be investigated to eliminate the error for removing a facet for an arbitrary number of layers. Second, the background removal algorithm can be more accurate to remove only noise in the images. Furthermore, to increase the quality of removing the background, new mathematic equations can be tested for their performance in this system.

4.6 Conclusions

This paper described an open-source low-cost reliable real-time monitoring platform for FFF-based 3-D printing based on a single camera system for three perspectives around 360 degrees. The results showed that the algorithms were effective at detecting a clogged nozzle, loss of filament, or an incomplete project for a wide range of 3-D object geometries. The error calculations were determined from the difference in shape between stlimage and cameraimage. The error was reported when these errors exceeded 7%. The validity of this approach using experiments shows that the error detection system is capable of a 100 % detection rate for failures. The method analyzed here has a better detection rate and a lower cost than previous methods. In addition, this method is generalizable to a wide range of FFF 3-D printer geometries, which enables further adoption of desktop 3-D printing for distributed manufacturing as wasted print time and filament are reduced.

4.7 References

1. Banzi, M. and Shiloh, M., 2014. Getting Started with Arduino: The Open Source Electronics Prototyping Platform. Maker Media, Inc.
2. Arduino https://www.arduino.cc/ (accessed 10.11.16)

3. Oxer, J. and Blemings, H., 2011. Practical Arduino: cool projects for open source hardware. Apress.
4. Sells E, Smith Z, Bailard S, Bowyer A, Olliver V (2010) RepRap: The Replicating Rapid Prototyper: Maximizing Customizability by Breeding the Means of Production. In: Piller FT, Tseng MM (eds) Handbook of Research in Mass Customization and Personalization: Strategies and concepts, Vol.1. World Scientific, pp 568-580.
5. Jones R, Haufe P, Sells E, Iravani P, Olliver V, Palmer C, Bowyer A (2011) RepRap – the replicating rapid prototype. Robotica 29:177–191. doi:10.1017/S026357471000069X
6. Bowyer A (2014) 3D printing and humanity's first imperfect replicator. 3D Printing and Additive Manufacturing 1(1): 4-5. doi:10.1089/3dp.2013.0003
7. Gibb, A. and Abadie, S., 2014. Building open source hardware: DIY manufacturing for hackers and makers. Pearson Education.
8. Raymond, E. The cathedral and the bazaar. Knowledge, Technology & Policy 1999, 12(3), pp.23–49.
9. Rundle, G. A Revolution in the Making. Simon and Schuster, 2014.
10. Wohlers T. (2016) Wohlers Report 2016. Wohlers Associates, Inc; 2016 Apr 10.
11. Wohlers, Terry. "3D Printing and Additive Manufacturing State of the Industry Annual Worldwide Progress Report." Wohlers Report (2014).
12. Wittbrodt, B.T.; Glover, A.G.; Laureto, J.; Anzalone, G.C.; Oppliger, D.; Irwin, J.L.; Pearce, J.M. Life-cycle economic analysis of distributed manufacturing with open-source 3-D printers. Mechatronics 2013, 23(6), pp. 713-726.
13. Pearce, J.M. Morris Blair, C. , Laciak, K. J., Andrews, R., A. Nosrat and I. Zelenika-Zovko, "3-D Printing of Open Source Appropriate Technologies for Self-Directed Sustainable Development", Journal of Sustainable Development 3(4), pp. 17-29 (2010).
14. Birtchnell, T. and Hoyle, W., 2014. 3D printing for development in the global south: The 3D4D challenge. Springer.
15. Pearce, J.M. (2015). Applications of Open Source 3-D Printing on Small Farms. Organic Farming 1(1), 19-35. DOI:10.12924/of2015.01010019
16. Kentzer, J., Koch, B., Thiim, M., Jones, R.W. and Villumsen, E., 2011, May. An open source hardware-based mechatronics project: The replicating rapid 3-D printer. In Mechatronics (ICOM), 2011 4th International Conference On (pp. 1-8). IEEE.
17. Irwin, J.L. Oppliger, D.E. Pearce, J.M. Anzalone, G. Evaluation of RepRap 3D Printer Workshops in K-12 STEM. 122nd ASEE 122nd ASEE Conf. Proceedings, paper ID#12036, 2015.
18. Gonzalez-Gomez, J., Valero-Gomez, A., Prieto-Moreno, A. and Abderrahim, M., 2012. A new open source 3d-printable mobile robotic platform for education. In Advances in autonomous mini robots (pp. 49-62). Springer Berlin Heidelberg.
19. Grujović, N., Radović, M., Kanjevac, V., Borota, J., Grujović, G. and Divac, D., 2011, September. 3D printing technology in education environment. In 34th International Conference on Production Engineering (pp. 29-30).

20. Schelly, C., Anzalone, G., Wijnen, B. and Pearce, J.M., 2015. Open-source 3-D printing technologies for education: Bringing additive manufacturing to the classroom. Journal of Visual Languages & Computing, 28, pp.226-237.
21. Campbell, I., Bourell, D. and Gibson, I., 2012. Additive manufacturing: rapid prototyping comes of age. Rapid Prototyping Journal, 18(4), pp.255-258.
22. Gibson, I., Rosen, D. and Stucker, B., 2014. Additive manufacturing technologies: 3D printing, rapid prototyping, and direct digital manufacturing. Springer.
23. O'Neill, P.F., Azouz, A.B., Vazquez, M., Liu, J., Marczak, S., Slouka, Z., Chang, H.C., Diamond, D. and Brabazon, D., 2014. Advances in three-dimensional rapid prototyping of microfluidic devices for biological applications. Biomicrofluidics, 8(5), p.052112.
24. Pearce, J.M., Anzalone, N.C. and Heldt, C.L., Open-source Wax RepRap 3-D Printer for Rapid Prototyping Paper-Based Microfluidics, Journal of Laboratory Automation 21(4) 510–516 (2016).
25. Knips, C., Bertling, J., Blömer, J. and Janssen, W., 2014. FabLabs, 3D-printing and degrowth–Democratisation and deceleration of production or a new consumptive boom producing more waste?. In Fourth International Conference on Degrowth for Ecological Sustainability and Social Equity.
26. Laplume, A., Anzalone, G.C. and Pearce, J.M. Open-source, self-replicating 3-D printer factory for small-business manufacturing. The International Journal of Advanced Manufacturing Technology. 85(1), pp 633-642 (2016). doi:10.1007/s00170-015-7970-9
27. Tech, R.P., Ferdinand, J.P. and Dopfer, M., 2016. Open Source Hardware Startups and Their Communities. In The Decentralized and Networked Future of Value Creation (pp. 129-145). Springer International Publishing.
28. Troxler, P. and van Woensel, C., 2016. How Will Society Adopt 3D Printing?. In 3D Printing (pp. 183-212). TMC Asser Press.
29. Pearce, J. M. 2012. Building Research Equipment with Free, Open-Source Hardware. Science 337 (6100): 1303–1304. DOI: 10.1126/science.1228183
30. Pearce, J.M. Open-Source Lab: How to Build Your Own Hardware and Reduce Research Costs, Elsevier, 2014.
31. Baden, T., Chagas, A. M., Gage, G., Marzullo, T., Prieto-Godino, L. L., & Euler, T. (2015). Open Labware: 3-D Printing Your Own Lab Equipment. PLOS Biology, 13(3). DOI: 10.1371/journal.pbio.1002086
32. Coakley, M. and Hurt, D.E., 2016. 3D Printing in the Laboratory Maximize Time and Funds with Customized and Open-Source Labware. Journal of Laboratory Automation, p.2211068216649578.
33. Kłodowski, A., Eskelinen, H. and Semken, S., 2015. Leakage-proof nozzle design for RepRap community 3D printer. Robotica, 33(04), pp.721-746.
34. Mercuri, R. and Meredith, K., 2014, March. An educational venture into 3D Printing. In Integrated STEM Education Conference (ISEC), 2014 IEEE (pp. 1-6). IEEE.
35. Chonga, S., Chiub, H.L., Liaob, Y.C., Hungc, S.T. and Pand, G.T., 2015. Cradle to Cradle® Design for 3D Printing. CHEMICAL ENGINEERING, 45.

36. Moilanen, J. and Vadén, T., 2013. 3D printing community and emerging practices of peer production. First Monday, 18(8).
37. Frauenfelder, M., Make: Ultimate Guide to 3D Printing 2014: Maker Media. Inc., O'Reilly, Sepaspol CA, 2013.
38. Alastair J. (2016) presented 16 common 3D Printing Problems and Solutions publishing all3dpweb. https://all3dp.com/common-3d-printing-problems-and-their-solutions/ (accessed 10.11.16)
39. Kreiger, M. and Pearce, J.M., 2013. Environmental life cycle analysis of distributed three-dimensional printing and conventional manufacturing of polymer products. ACS Sustainable Chemistry & Engineering, 1(12), pp.1511-1519.
40. Vera, J., 2010. Promoting Tools that integrate LCA into the Product Design Process: a Case Study in Ontario.
41. Kreiger, M. and Pearce, J.M., 2013. Environmental impacts of distributed manufacturing from 3-D printing of polymer components and products. In MRS Proceedings (Vol. 1492, pp. 85-90). Cambridge University Press.
42. Kostakis, V., Roos, A. and Bauwens, M., 2016. Towards a political ecology of the digital economy: Socio-environmental implications of two competing value models. Environmental Innovation and Societal Transitions, 18, pp.82-100.
43. Bonvoisin, J., 2016. Implications of Open Source Design for Sustainability. In Sustainable Design and Manufacturing 2016 (pp. 49-59). Springer International Publishing.
44. Kleszczynski, S., Zur Jacobsmühlen, J., Sehrt, J.T. and Witt, G., 2012, August. Error detection in laser beam melting systems by high resolution imaging. In Proceedings of the Solid Freeform Fabrication Symposium.
45. zur Jacobsmühlen, J., Kleszczynski, S., Schneider, D. and Witt, G., "High resolution imaging for inspection of laser beam melting systems." 2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC). IEEE, 2013.
46. zur Jacobsmühlen, J., Kleszczynski, S., Witt, G. and Merhof, D., "Robustness analysis of imaging system for inspection of laser beam melting systems." Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA). IEEE, 2014.
47. Kleszczynski, S., zur Jacobsmühlen, J., Reinarz, B., Sehrt, J.T., Witt, G. and Merhof, D.,"Improving process stability of laser beam melting systems." Proceedings of the Frauenhofer Direct Digital Manufacturing Conference. 2014.
48. Faes, M., Abbeloos, W., Vogeler, F., Valkenaers, H., Coppens, K. and Ferraris, E., 2014, September. Process monitoring of extrusion based 3D printing via laser scanning. In PMI 2014 Conference Proceedings (Vol. 6, pp. 363-367).
49. Volpato, N., Aguiomar Foggiatto, J. and Coradini Schwarz, D., 2014. The influence of support base on FDM accuracy in Z. Rapid Prototyping Journal, 20(3), pp.182-191.
50. Wu, H., Wang, Y. and Yu, Z., 2016. In situ monitoring of FDM machine condition via acoustic emission. The International Journal of Advanced Manufacturing Technology, 84(5-8), pp.1483-1495.

51. Hurd, S., Camp, C. and White, J., 2015, November. Quality Assurance in Additive Manufacturing Through Mobile Computing. In International Conference on Mobile Computing, Applications, and Services (pp. 203-220). Springer International Publishing.
52. Baumann, F. and Roller, D., 2016, January. Vision based error detection for 3D printing processes. In MATEC Web of Conferences (Vol. 59). EDP Sciences.
53. Opencv http://opencv.org/ (accessed 10.11.16)
54. Python software foundation [US] https://www.python.org/ (accessed 10.11.16)
55. RaspberryPi https://www.raspberrypi.org/ (accessed 10.11.16)
56. Microsoft https://msdn.microsoft.com/ (accessed 13.11.16)
57. Straub, J. "Initial Work on the Characterization of Additive Manufacturing (3D Printing) Using Software Image Analysis." Machines 3.2 (2015): 55-71.
58. David G. (2016) presented adding a Raspberry Pi case and a camera to your LulzBot Mini publishing kupoos Web. http://www.kupoos.com/video/q7oqOPzCHYE/adding-a-raspberry-pi-case-and-a-camera-to-your-lulzbot-mini/ (accessed 20.11.16)
59. Admin (2016) Free IP Camera Monitoringfor 3-D printerwith old webcam usb in 5 min publish-ing printer3d Web. http://www.printer3d.one/en/forums/topic/free-ip-camera-monitoring-for-3d-printer-with-old-webcam-usb-in-5min/ (accessed 18.03.17)
60. MusicTech (2016) Controlling and Monitoring your 3D printer with BeagleBone and Octoprint publishing element14 community web. https://www.element14.com/community/community/design-challenges/musictech/blog/2016/03/16/controlling-your-3d-printer-with-beaglebone-and-octoprint (accessed 18.03.17)
61. Jeremy S. (2014) Monitoring your 3D prints publishing 3duniverse web. https://3duniverse.org/2014/01/06/monitoring-your-3d-prints/ (accessed 18.03.17)
62. KenVersus (2015) Logitech C170 webcam mount for daVinci 3D printer publishing Thingiverse web. http://www.thingiverse.com/thing:747105 (accessed 18.03.17)
63. Camera Calibration publishing OpenCV-Python Tutorials web. http://opencv-python tutroals. readthedocs.io/en/latest/py_tutorials/py calib3d/py_calibration/py_calibration.html. (accessed 15.03.17)
64. Nuchitprasitchai, S., Roggemann, M.C.and Havens, T.C. "Algorithm for Reconstructing Three Dimensional Images from Overlapping Two Dimensional Intensity Measurements with Relaxed Camera Positioning Requirements to reconstruct 3D image." IJMER 6.9 (2016): 69-81.
65. Nuchitprasitchai, S., Roggemann, M.C.and Pearce, J.M. "Factors Effecting Real Time Optical Monitoring of Fused Filament 3-D Printing." (to be published).
66. Source code for single camera. Open Science Framework. https://osf.io/atgx8/ (accessed 4.6.2017)
67. Anzalone, G.C., Wijnen, B. and Pearce, J.M., 2015. Multi-material additive and subtractive prosumer digital fabrication with a free and open-source convertible delta RepRap 3-D printer. Rapid Prototyping Journal, 21(5), pp.506-519.

68. MOST Delta 3-D printer http://www.appropedia.org/Delta_Build_Overview:MOST (accessed 3.12.16)
69. LED Light Sources https://www.dollartree.com/ (accessed 3.12.16)
70. RaspberryPi: https://www.raspberrypi.org/ (accessed 3.12.16)
71. STL (file format): https://en.wikipedia.org/wiki/STL_(file_format) (accessed 05.04.17)
72. OpenSCAD http://www.openscad.org/ (accessed 3.12.16)
73. Cura https://ultimaker.com/en/products/cura-software (accessed 3.12.16)
74. Thing-O-Fun (2012) Exploded Planetary Gear Set publishing Thingiverse web. http://www.thingiverse.com/thing:18291 (accessed 3.12.16)
75. Jetty (2012) Paper Crimper publishing Thingiverse web. http://www.thingiverse.com/thing:17634 (accessed 3.12.16)
76. Droftarts (2012) Parametric pulley-lots of tooth profiles publishing Thigiverse web. http://www.thingiverse.com/thing:16627 (accessed 3.12.16)
77. Nuchitprasitchai 3-D models https://osf.io/utp6g/ (accessed 5.04.17)
78. Wijnen, B., Anzalone, G.C., Haselhuhn, A.S., Sanders, P.G. and Pearce, J.M., 2016. Free and open-source control software for 3-D motion and processing. Journal of Open Research Soft-ware, 4(1).
79. OpenCV camera calibration and 3-D reconstruction, http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#stereobm (accessed 3.12.16)
80. Mathworks, Pricing and Licensing https://www.mathworks.com/pricing-licensing/index.html?intendeduse=comm (accessed 8.12.16)
81. Nuchitprasitchai, S. Rod Alarm. Published on Appropedia.org http://www.appropedia.org/Rod_alarm (accessed 20.03.17)
82. Tjmahan Raspberry Pi Comtrol and Wireless Interface. Published on Appropedia.org http://www.appropedia.org/Raspberry_Pi_Control_and_Wireless_Interface (accessed 20.03.17)
83. Barker, B. Thrown Rod Halt Mod. Published on Appropedia.org http://www.appropedia.org/Thrown_Rod_Halt_Mod (accessed 20.03.17)

Chapter 5: 360 Degree Real-Time Monitoring of 3-D Printing Using Computer Analysis of Two Camera Views[4]

5.1 Abstract

Prosumer (producing consumer)-based desktop additive manufacturing has been enabled by the recent radical reduction in 3-D printer capital costs created by the open-source release of the self-replicating rapid (RepRap) prototype. Despite this success, these low-cost 3-D printers still suffer from a litany of printing challenges. There have been some efforts made to this end, which are either too expensive or not automated. A more promising method is to use computer vision and although there has been progress in this area the success rates are still too low for widespread use. To overcome these challenges an open source low-cost reliable real-time optimal monitoring platform for RepRap-based 3-D printing from double cameras is presented here. This error detection system is implemented with low-cost web cameras and extended from the basic approaches discussed above for 360 degrees around the printed object from three different perspectives by extending the algorithm using SIFT and RANSAC. The algorithm is developed in Python and run on a Raspberry Pi3 mini-computer to reduce the costs and computation time. For 3-D printing monitoring in three different perspectives, the systems are tested with four different 3-D object geometries (two experiments tested in the normal operating mode and two in failure states). This system is tested with two different techniques in the image pre-processing step: SIFT and RANSAC to rescale and rectify, and with non-rescale and rectification. The error percentage is calculated by the horizontal, and horizontal and

[4] This chapter has been completed as an article to submit. Citation: Nuchitprasitchai S, Roggemann M, & Pearce J (2017). 360 Degree Real-Time Monitoring of 3-D Printing Using Computer Analysis of Two Cameras Views.

vertical magnitude methods. The error calculations were determined from the horizontal and vertical magnitude of 3-D reconstruction image for the non-rescale and rectification technique successfully 100% detects the normal printing and failure state for all models, which is better than the single camera set up only. The computation time of the non-rescale and rectification technique is 2X faster than SIFT and RANSAC to rescale and rectification technique.

5.2 Introduction

Prosumer (producing consumer)-based additive manufacturing has been enabled by the recent radical reduction in 3-D printer capital costs (Wohlers, 2016) created by the open-source release of the self-replicating rapid (RepRap) prototyper (Sells et al., 2010; Jones, et al., 2011; Bowyer, 2014). The open-source hardware approach (Gibb and Abadie, 2014) has followed the traditional rapid development seen in free and open source software (Raymond, 1999) and the top-desktop 3-D printers are now routinely open source RepRaps derivatives (Make, 2017). The fast growth of the RepRap 3-D printers is a result of their ability to replicate (e.g. print their own parts) and self-upgrade its own parts (e.g. print a new cooling fan) as well as their ability to easily pay for themselves by fabricating consumer goods (Wittbrodt et al., 2013; Petersen and Pearce, 2017). In addition, open source desktop 3-D printers have been applied to create high value items in a wide range of fields including: rapid prototyping (Campbell, et al., 2012; Gibson, et al., 2014), distributed manufacturing (Kentzer, et al., 2011;;Pearce, 2015;), education (Irwin, et al., 2015; Gonzalez-Gomez, et al., 2012; Schelly, et al., 2015) , sustainable technology (Pearce, et al., 2010; Fox, 2010; Birtchnell and Hoyle, 2014; Pearce, 2015), scientific tools

(Pearce, 2012; Pearce, 2014; Baden, et al., 2015; Coakley and Hurt, 2016), microfluidics (O'Neill, et al., 2014; Pearce, et al., 2016).

Despite this success, these low-cost 3-D printers still suffer from a litany of printing challenges related to building up a part from thermoplastic one layer at time from a flat print bed including warping, elephant foot (thicker part touching the print bed), bed adhesion (prints peeling off of the bed during print), distortion due to shrinking, skewed prints/ shifted layers, layer misalignment, clogged nozzles, or snapped filament (Campbell, et al., 2012; O'Neil, et al., 2014; Rimock, 2015). These unintended results reduce the economic as well as the environmental advantage of distributed manufacturing with 3-D printing (Laplume, et al., 2016; Tech, et al., 2016; Troxler and Woensel, 2016; Pearce, 2012; 2014) in the aspect of environmental and sustainability. Many works had been done to automatically detect the errors while printing, but most of them are for the expensive laser-based 3-D printing (Kleszczynski, et al., 2012; Zur, et al., 2014, Kleszczynski, et al., 2014; Concept Laser, 2016). Therefore, there is an acute need for a low-cost real-time error detection system for prosumer-grade 3-D printers.

There have been some efforts made to this end. There were several works detecting an error based on the laser and piezoelectric sensors, which are not easily adapted to the low-cost market (Faes, et al., 2014; Volpato, et al., 2014; Wu, et al., 2016). A more promising method is to use computer vision, which has been shown to be highly effective at process monitoring for manufacturing (Atli, et al., 2006; Bradley, et al., 2001; Bradski, et al., 2008; Edinbarough, et al., 2005; Golnabi, et al., 2007; Ji, et al., 2002; Kerr, et al., 2006; Klancnik, et al., 2015; Lanzetta, et al., 2001; Li, et al., 2010; Pfeifer, et al., 2000; Wang, et al., 2007).

Some previous works used cameras to monitor the 3-D printing process (Hurd, et al., 2015; Baumann and Dieter, 2016; Straub, 2015). Hurd et al. installed Samsung Galaxy Tab 3 on the printer and monitored the printing via mobile phone (Hurd, 2015) but this can monitor only the top view of the printed part. Therefore, horizontal size can be determined. Baumann et al. used OpenCV (Opencv, 2016), Python (Python, 2016) and a PlayStation eye cam to detect detachment, missing material flow and deformed object in 3-D printing (Baumann, and Dieter, 2016), however, this work can detect only the shape of the printed part from only one side with success rate of 80%. Straub successfully applied a visible light 3-D scanning system, five Raspberry Pi cameras, Raspberry Pi (Raspberry, 2016), and open source software approach with C# and Dot Net Framework (Microsoft, 2016) to detect incomplete prints (Straub, 2015). Nonetheless, the work can only detect error in the shape aspect. Other solutions to detect the failure 3-D printing in the RepRap 3-D printer have had a video monitor of printing but the user must manually check the video and stop the printing if something goes wrong (Gewirtz, 2016; Printer3-D, 2016; Carmelito, 2016; Simon, 2014; KenVersus, 2015).

To monitor errors during FFF-based 3-D printing, an open source low-cost reliable real-time optimal monitoring platform for FFF-based 3-D printing from double cameras is presented here. This error detection system is implemented with low-cost web cameras and extended from the basic approaches discussed above for 360 degrees around the printed object from three different perspectives by extending the algorithm using the Scale Invariant Feature Transform (SIFT) (Lowe, 1999) and the RANdom SAmple Consensus (RANSAC) (Fischler and Robert, 1981) models previously described (Nuchitprasitchai, et al., 2016). The algorithm is developed under open-source Python and run on a Raspberry

Pi3 mini-computer to reduce the costs and computation time. For 3-D printing monitoring in three different perspectives, the systems is tested with four different 3-D object geometries (two experiments tested in the normal printing and two in the failure state). The normal printing state means that the filament can print correctly and complete printing the 3-D object. The failure state is the incomplete printing the 3-D object. This system is tested with two different techniques in the image pre-processing step: SIFT and RANSAC to rescale and rectify, and with non-rescale and rectification. The error percentage is calculated by the horizontal magnitude. Then the technique that can detect the error in the normal printing and the failure state correctly will be used in the second experiment were two different error detection methods are used: horizontal magnitude, and horizontal and vertical magnitudes. The results are discussed; conclusions are drawn and the limitations of these approaches are detailed.

5.3 Method

5.3.1 Experimental Equipment

For this work, optical experiments were setup around a delta-style (Anzalone, et al., 2015) RepRap as shown in Figure 5.1 running double cameras. This low-cost (<US$500 in parts) open source delta-style polymer printing RepRap (MOST Delta). The MOST Delta is a RepRap (Anzalone, et al., 2016) derived from the Rostock (Rostock, 2016) printer with a cylindrical build volume 270 mm in diameter and 250 mm high and overall dimensions of 375 mm diameter and 620 mm high. The double camera error detection use left and right images do three 3-D reconstruction (as seen in Figure 5.2). A Python algorithm was written for the experimental setup and is made available free and open source under an AGPLv3

license (Nuchitprasitchai, 2016). A different Python algorithm is used for each experimental setup, but the same type of webcam, 3-D printer, Raspberry Pi3, USB 3.0 hub with 12V/3A power adapter, three LED light sources, tested objects, black printing base, black background, and filament brand are used. Due to the distance between the camera and the printer for the experiment setup, the field of view for both cameras can cover the printed area of 70 mm in width and 60 mm in height. The relation of geometry between the 3-D printer and the camera system need to be known for using camera calibration technique (OpenCV, 2016) to calculate the intrinsic and extrinsic parameters for a specific camera setup. These parameters will be used to correct for lens distortion and image rectification. The three LED light sources (DollarTree, 2016) are installed on the three sides of the printer. All light sources are connected to the circuit with 4 volts from a DC power supply. The three pairs of cameras are setup on the same side of LED light sources. All cameras are connected to a 7 port USB 3.0 hub with 12V/3A power adapter which is connected to Raspberry Pi3. The cameras used in this study are six Logitech C525 webcams, with an image size of 480-by-640 (height-by-width), pixel size is 5.52-by-5.82 μm (height-by-width), and a focal length of 39.5 mm. The pixel size and the focal length calculation of the webcam below. The circuit of the light sources is shown in Figure 5.3.

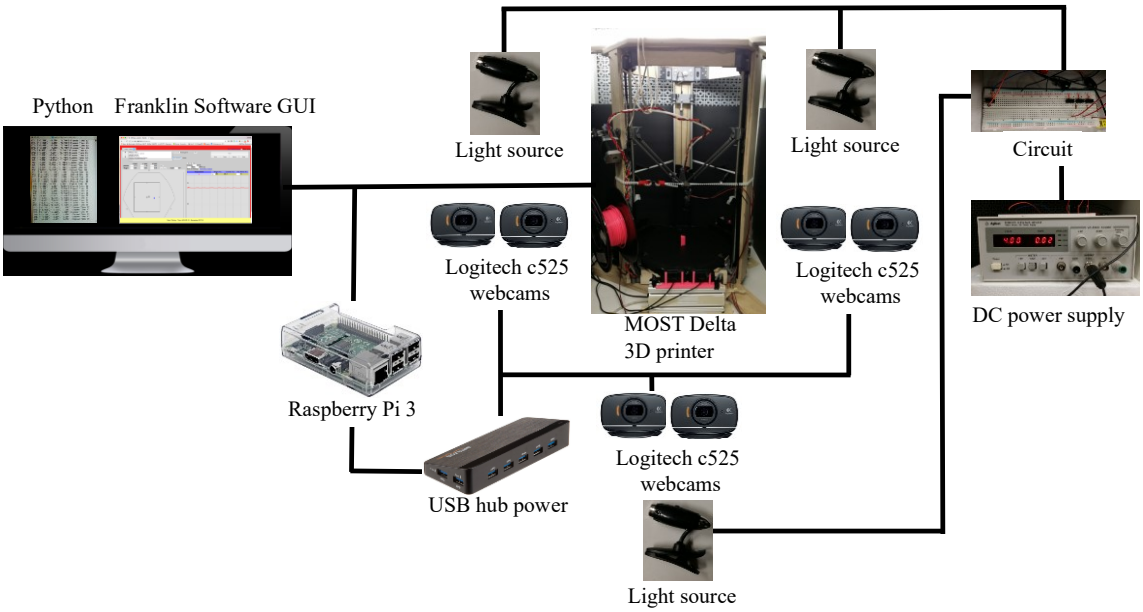


Figure 5.1 MOST Delta printer experiment setup

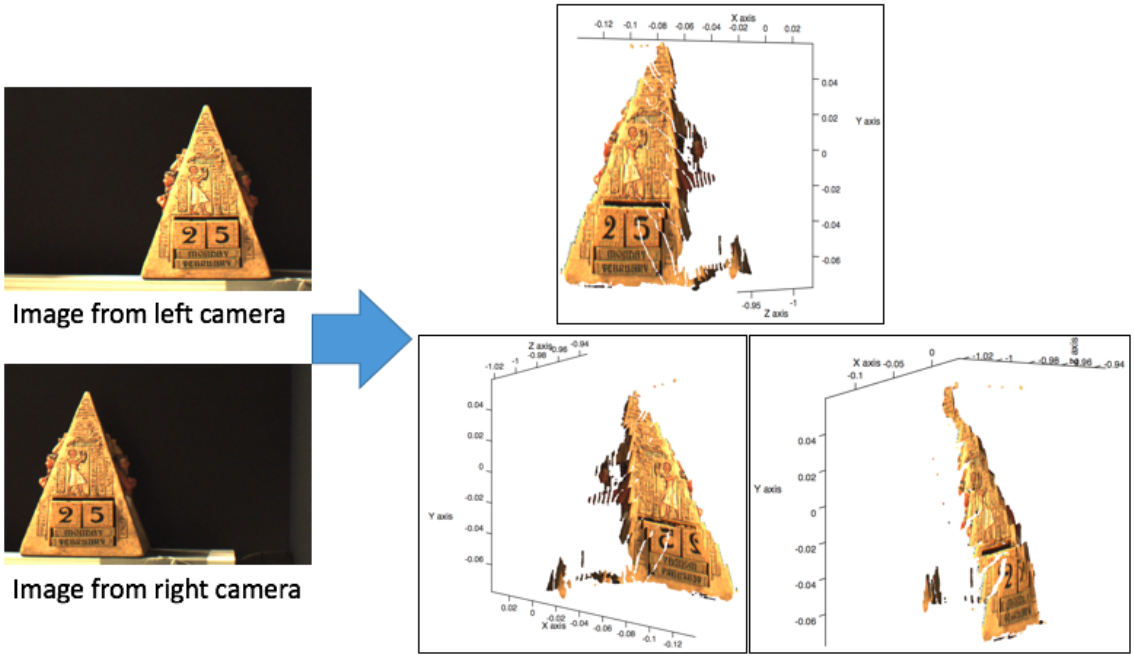


Figure 5.2 3-D reconstruction

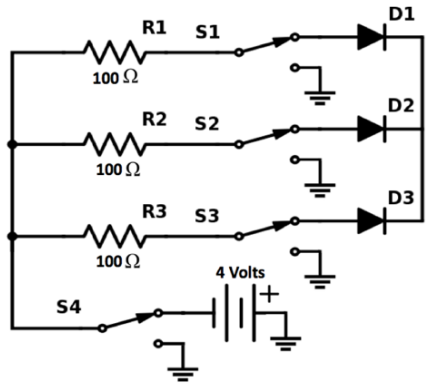


Figure 5.3 Light source circuit

5.3.2 Theory

5.3.2.1 Calculating Webcam Pixel Size and Focal Length

Unlike scientific cameras, inexpensive webcams do not normally ship with detailed technical specifications. The procedure below enables the extraction of pixel size and focal length from any inexpensive webcam. The Logitech C525 webcams used here do not come with information on the pixel size and focal length (on the package or the website), so the webcam was taking apart to calculate this information through the sensor size in the webcam as shown in Figure 5.4. The webcam sensor size is 2.52-by-3.73 mm (height-by-width), and the webcam diagonal is 4.50 mm. The width and the height of pixel size are calculated by

$$W_d = W_s / W_i \quad (\mu m) \tag{5.1}$$

Where $W_d$is a width of pixel size (μm), W_s is a width of sensor size (mm), and W_i is a width of images size (pixels).

$$H_p = H_s / H_i \quad (\mu m) \tag{5.2}$$

Where H_p is a height of pixel size (μm), H_s is a height of sensor size (mm), and H_i is a height of images size (pixels).

The checkerboard image shown in Figure 5.5 is taken to calculate the focal length in pixels. The checkerboard image was printed in 2-D for taking the image and the size of checkerboard square on a paper is 7-by-7 mm (Nuchitprasitchai, et al., 2016). The checkerboard image was taken where the distance between the image and the webcam was 230 mm, and the size of checkerboard square in the image was 20-by-20 pixels. The focal length in pixels is calculated by

$$F = (P * D)/W_c \qquad \text{(pixels)} \qquad (5.3)$$

Where F is the focal length (pixels), P is the size of checkerboard square in the image (pixels), D is the distance between the image and the webcam, and W_c is the size of checkerboard square on a paper (pixels).

$$f = (F * W_d)/W_i \qquad \text{(mm)} \qquad (5.4)$$

Where f is the focal length (mm), F is the focal length (pixels), W_d is a width of pixel size (μm), and W_i is a width of images size (pixels).

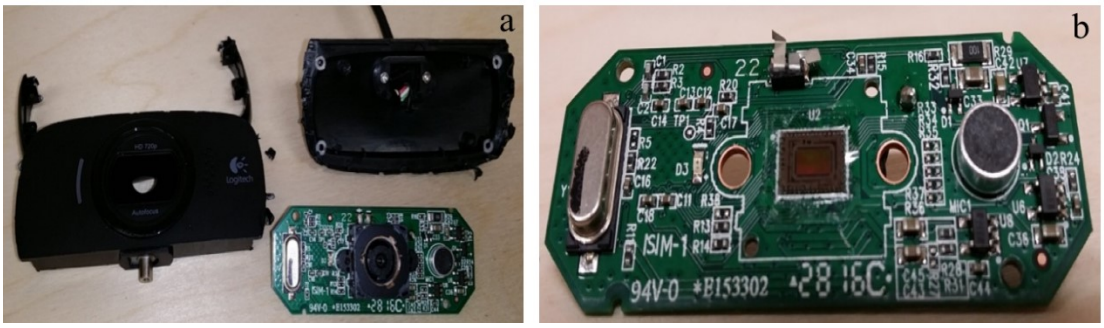


Figure 5.4 Logitech C525 webcam: a) webcam circuit board and body, and b) sensor of webcam

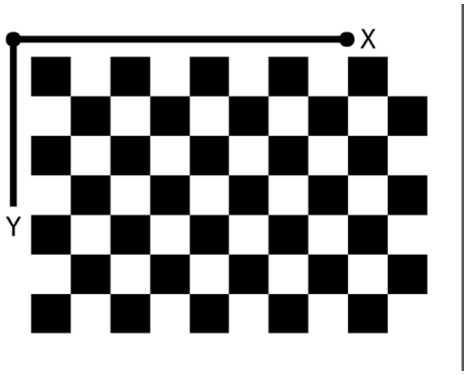


Figure 5.5 Example of the checkerboard image

5.3.2.2 Computer Vision Error Detection

There are three steps to prepare the error detection system before printing a 3-D model: 1) camera calibration, 2) preparing STereoLithography (STL) files and resultant images, and 3) setting up a pause and loop to move the extruder out of the view of the cameras for imaging. STL file is a file format describing 3-D model by using series of connected triangles to create the surface of the model and it is usually generated by computer aided design (CAD) software. The first step is camera calibration. Sixteen chessboard images are taken from three different views of the cameras after the 3-D printer experiment is setup

for camera calibration. There are six cameras named as camera0, camera1, camera2, camera3, camera4, and camera5. The camera0 and camera1 are setup as the first pair of cameras, camera2 and camera3 are setup as the second pair, and camera4 and camera5 are setup as the third pair. The camera0, camera 2, and the camera 4 are setup as the left cameras, and camera1, camera 3, and the camera 5 are setup as the right cameras. The calibration is calculated and saved as CalibrationData1, CalibrationData2, and CalibrationData3. The second step is preparing stlimage by slicing stl files into every N layers where the error will be detected as shown in Figure 5.6. The layer height and the amount of slicing layers need to be assigned for slicing stl file in three different views of the cameras. The layer height and the amount of total layers can be found in gcode file. All data at every N layers from stl file are plotted in x, y, z axes to display the shape of the rendered 3-D model, which can be observed from different viewpoints. Thus, the shape of the stlimage is saved as PNG image type on xz-plane. If a modulus after division between the total height and the height of every N layers is not equal to zero, the last PNG files are named as the amount of total layers. For example, if the 3-D model in gcode file has 129 total layers, layer height of 0.2 mm, and the 3-D model is slicing in every 30 layers, then the stl file is sliced at layer 30, 60, 90, 120, and 129 which result in heights of 6, 12, 18, 24, and 25.8 mm, respectively. The first stl slicing files are saved as SCAD30_1.png, SCAD30_2.png, and SCAD30_3.png, the next slicing files are saved as SCAD60_1.png, SCAD60_2.png, and SCAD60_3.png, and so on. After slicing stl files for four models, it was found that three stl files can start slicing every layer 10, 20, or 30, but t55gear stl file can start slicing at every 30 layers. Therefore, this study will be taking six images every 30 layers are printed. The last step in the process is involves setting up a pause and a loop to

move the extruder out of the images every N layers in order to eliminate visual noise in the object images, the extruder of 3-D printing will be paused and moved to the certain height. The 3-D model is designed in OpenSCAD version2015.03-3 (OpenSCAD, 2016) and it is rendered and saved into stl file. After the 3-D model stl file is opened in Cura version15.04.6 (Ultimaker, 2016), the 3-D model is saved as gcode file. The 3-D model gcode file is opened by any text editor program to add the extra code in every N layers as shown in Figure 5.7.

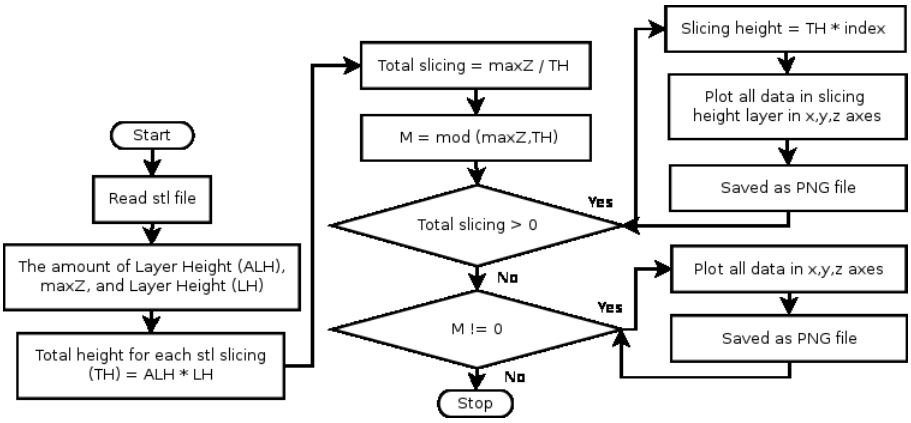


Figure 5.6 Slicing stl file flowchart

```
; MOVE Extruder for taking photo
;

G91                     ;relative positioning
G1 E-10 F2000           ;retract the filament 10 mm before lifting the nozzle, to release some of the pressure
G1 Z+100  X-20 Y-20     ;move Z 100 mm up and retract filament even more
G4 P30000               ; wait 30 secs for nozzle length to stabilize
G90                     ;absolute positioning
```

Figure 5.7 Python code for pausing and moving the extruder to take the images

The 3-D printing models chosen after the preparing stlimage step are sun gear (Thing-O-Fun, 2016), prism, gear (Jetty, 2016), and t55gear (Droftarts, 2016) are available (Nuchitprasitchai, 2017) as shown in Figure 5.8. The printing parameters used are: layer

height 0.2 mm, shell thickness 1 mm, unable retraction, bottom/top thickness 1mm, fill density 20%, print speed 60 mm/s, printing temperature 180oC, diameter filament 1.94 - 1.98 mm, flow filament 100%, and nozzle size 0.5 mm. The PLA filament used in this experiment is Hatchbox 3-D PLA with dimensional accuracy +/- 0.05 mm on 1 kg spools, 1.75 mm diameter with pink color.

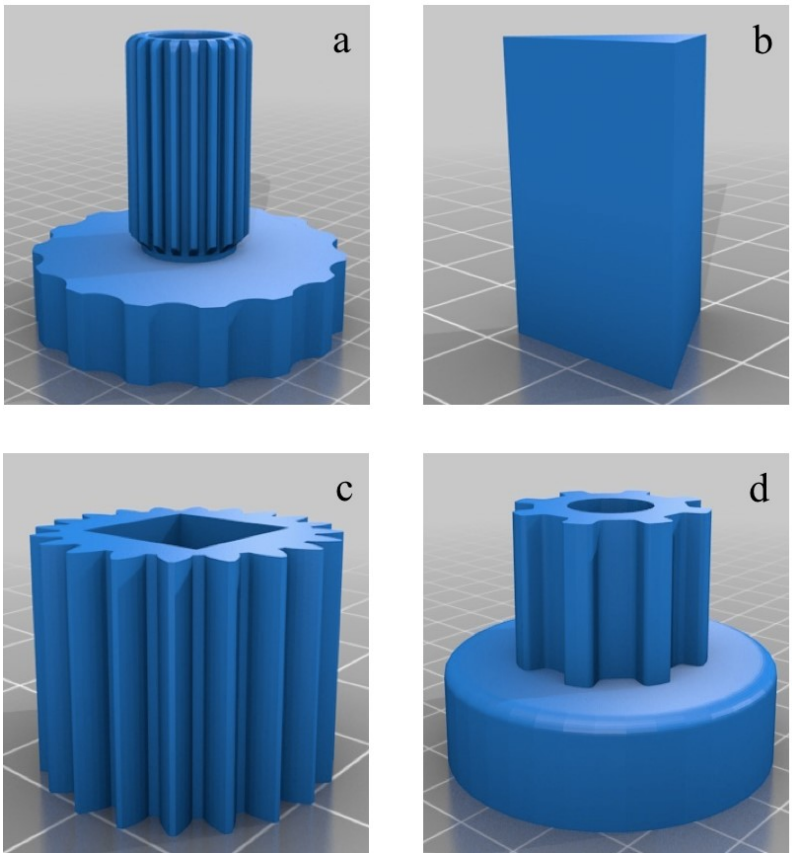


Figure 5.8 Rendering of STL models for testing: a) sun gear, b) prism, c) gear, and d) t55gear

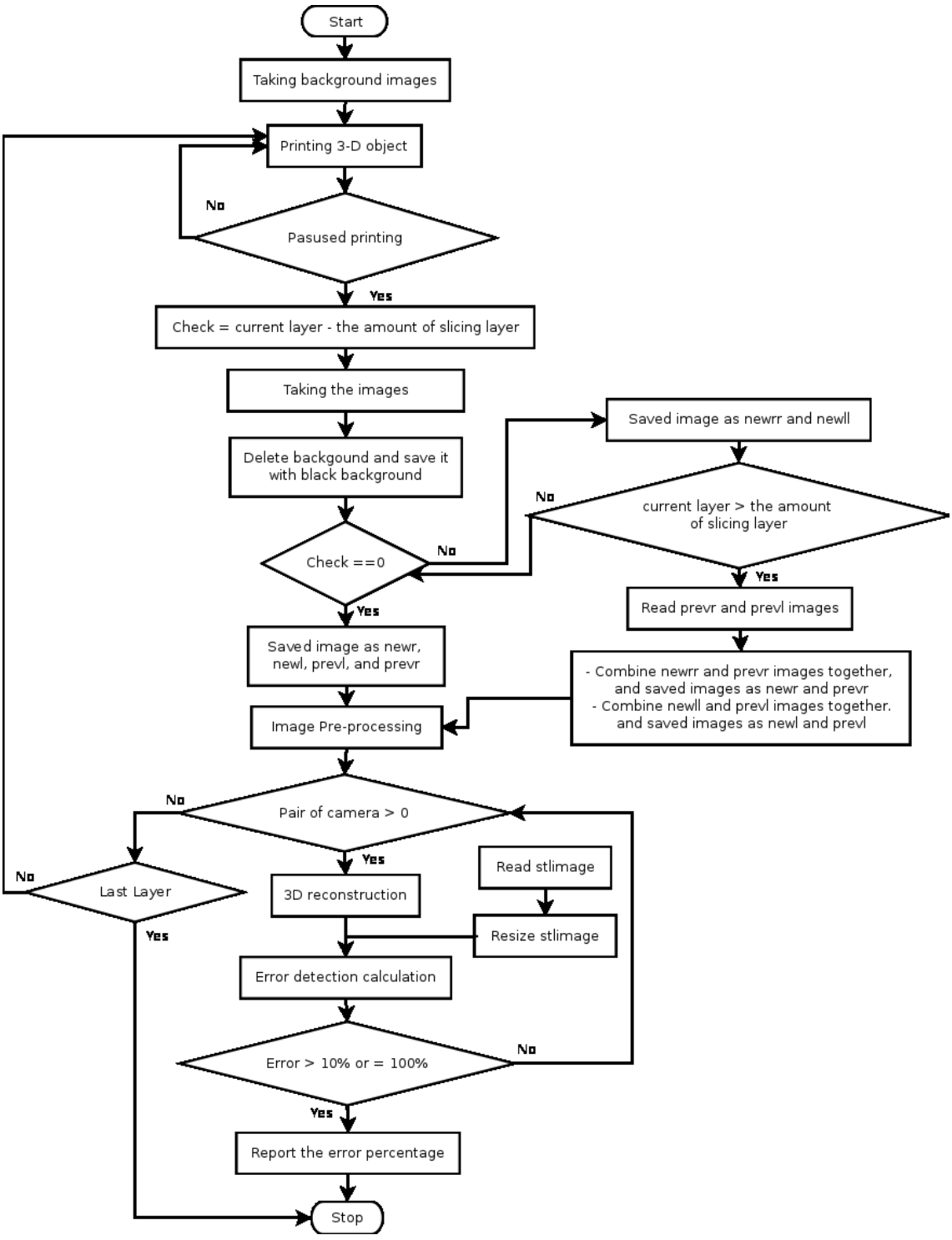


Figure 5.9 The error detection for double camera system flowchart

The double error detection algorithm, written in Python, will display the error percentage. If the error percentage is greater than 10% then the printing is failed as shown in Figure 5.9. After the user orders printing a 3-D model through Franklin (Wijnen, et al., 2016) with the amount of slicing layer number (N), the background images are taken before printing the 3-D model. The background images are taken from six cameras saved as bgr1, bgr2, bgr3, bgl1, bgl2, bgl3, where the bgr represents the images taken from the right cameras, and bgl images are taken from the left cameras and the number 1, 2, and 3 mean the first, the second and the third pair of cameras. At every N layers, the printer is paused to detect an error. After the extruder is moved to the certain height, the object images are taken. The object images are taken from six cameras saved as objr1, objr2, objr3, objl1, objl2, objl3.

The objr represents the object images taken from the right cameras, and objl are the object images taken from the left cameras. The number 1, 2, and 3 mean the first, the second and the third pair of cameras. In the removing background process, the object images need to be remove background, rendered black between bg and obj images for each pair of camera, and saved as newl.png and newr.png for each pair of camera. But there is a light reflection of the object in the images that may cause an error. The new.png from the previous error detection will be used in the next error detection to create the new images named as newll.png and newrr.png. For an example, if the current layer is the same as the amount of slicing layer number, the images after removing background are saved into two different file names as newr and prevr. If they are not equal, they are saved as newrr. The prevr images need for the next step to improve removing background. If the current layer is greater than the amount of slicing layer number, the prevr image is read to combine the interested object area between the prevr and the newrr images into two different file names

as newr and prevr. After input images are ready for 3-D reconstruction in image pre-processing step, the cameraimage is used to calculate the 3-D object points and the stlimage is rescaled to find the magnitude of the width. To reduce the computation time detecting an error, the error detection is calculated for each pair at a time started from first pair of images, second pair of images and third pair of images. Because the 3-D reconstruction calculation for each pair cost n second, so the total for three 3-D reconstruction cost O(N). The last step, the error detection is calculated. If there is an error, it will return the percentage of error and can be used as trigger to turn of the printer and alert the user.

5.3.3 Experiments

For this study, there are two experiments tested: image pre-processing and error detection. The image pre-processing step is run by two different techniques: SIFT and RANSAC to rescale and rectification, and with non-rescale and rectification. The error detection is tested by two different methods: horizontal magnitude, and horizontal and vertical magnitude. All cases are tested under normal printing and failure state. In the normal printing state means that the filament is in normal condition to complete printing the 3-D object. In the failure state is incomplete printing the 3-D object. The details for each experiment is explained later.

5.3.3.1 Image Pre-Processing

At every N layer that is equal to the amount of slicing layer numbers, the six object images are taken from three pair of cameras in different three perspectives. The background is removed and rendered black between bg and obj images for each camera such as (bgr1, objr1), (bgr2, objr2), (bgr3, objr3), (bgr4, objr4), (bgr5, objr5), and (bgr6, objr6). The new

images after removing background are named (newr1, prevr1), (newr2, prevr2), (newr3, prevr3), (newl1, prevl1), (newl2, prevl2), and (newl3, prevl3) when the current layer is the same as the amount of slicing layer number. If they are not equal, the images are saved as (newrr1, prevr1), (newrr2, prevr2), (newrr3, prevr3), (newll1, prevl1), (newll2, prevl2), and (newll3, prevl3). The prev images need for the next step to improve removing background. For an example, if the current layer is greater than the amount of slicing layer number, the prevr image is read to combine the interested object area between the prevr and the newrr images into two different file names as newr and prevr. Distortion is removed from all six images by intrinsic parameters from camera calibration. Next, a region of interest (ROI) is calculated from the image by converting the color image into a gray scale image, then converting it into binary image. The object area in the binary image is converted to be white used as the ROI, otherwise is converted to be black. After these steps, the images are ready for image pre-processing step tested by the SIFT and RANSAC to rescale and rectification, and with non- rescale and rectification. The 3-D points of the interested object is calculated. The algorithm for image rescaling, image rectification, and 3-D points calculation has been described previously (Nuchitprasitchai, 2016). The error percentage is calculated by using horizontal magnitude method. The error detection is calculated for each pair of cameras once at a time. It starts from the first, the second, and the third pair of the images, respectively. If the error detection is greater than 10%, this can be used as a trigger to pause the printer and notify the user. But if the error is less than 10%, then the next pair of the images is calculated to detect an error.

5.3.3.1.1 SIFT and RANSAC to Rescale and Rectification

The interested object location between the left and the right images may have different scale or size, or they locate in different rows or columns in the image. To resolve this problem, the SIFT and the RANSAC models are applied for image rescaling and image rectification. The 3-D points then are calculated.

5.3.3.1.2 With Non-Rescale and Rectification

Due to using SIFT and RANSAC in Python has error from wrong matching points or no matching points, and affected the rescale and rectification process which results in high error values. However, the images taken by the cameras are already in very similar scale and rectify. The six images are used to calculate the 3-D surface points.

5.3.3.2 Error Detection

After the image pre-processing experiments with two different techniques, the error percentage of non-rescale and rectification is more accurate, therefore this method is used for error detection experiment by horizontal magnitude, and horizontal and vertical magnitude methods as explained below. First pair of the images is processed, and if the error is greater than 10%, it can be used as a trigger the error and report to the user; otherwise the next pair of the images is calculated to detect an error until the last pair of the images.

5.3.3.2.1 Horizontal Magnitude

The error detection is obtained by subtracting the magnitude of the width of interested area at the current printing layers between the 3-D reconstruction and stlimage model.

5.3.3.2.2 Horizontal and Vertical Magnitude

The horizontal error magnitude is calculated as mentioned before. If only the width data available at the height of the current printing, then the vertical error magnitude is obtained by subtracting the magnitude of the height of interested area between the 3-D reconstruction and stlimage. If the width data is not available, then the percentage of error is 100.

5.3.4 Validation

The dimensions of the 3-D printed objects are measured with a digital caliper (+/-0.05mm). A 3-D reconstruction of the object is calculated from two images and the object size is calculated. Next, the size of both objects is compared to calculate size difference an error of the reconstruction. For validation of this approach four different test objects are printed including a) sun gear, b) prism, c) gear, and d) t55gear

5.4 Results

The experimental procedures were tested with different object geometries (sun gear, prism, gear, and t55gear). In order to eliminate the background noise from the extruder, the images were taken after pausing printing and the extruder was moved out from six camera views. The example of the full sun gear model image from three different perspectives are shown in Figure 5.10. The results of the two experiments reported as followed.

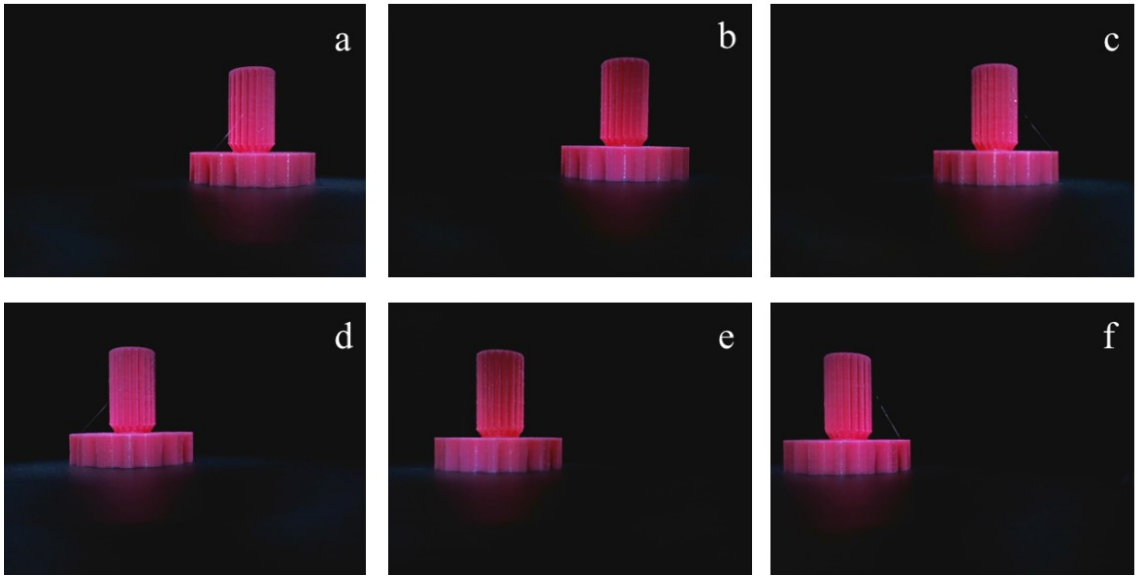


Figure 5.10 The example of full model of sun gear image results from the first, the second and the third pair of cameras respectively: a-c) the images from the left camera, and d-f) the images from the right camera.

5.4.1 Image Pre-Processing

After order printing the 3-D model, all six background images were taken from six cameras in three different views. For each technique, there are tested in normal printing and failure state. After the extruder was paused and moved up for 100 mm at every 30 layers, the six object images from six cameras in three different perspectives was taken. The error detection processed from six object and six background images in different technique for image pre-processing presented as followed.

5.4.1.1 SIFT and RANSAC to Rescale and Rectification

A) Normal Printing State

Figure 5.11 shows that most of the errors are greater than 10% for each geometry except the sun gear model at layers 60 to 240, the error is less than 10%. The printing layers at 30, 120, and 150 layers in the prism model had zero error percentage because the SIFT and

RANSAC did not have enough matching points to rescale. Therefore, they could not calculate 3-D object points. In sun gear, gear, and t55gear graph, there were some printing layers that the error percentage had the huge difference because the SIFT and RANSAC had the wrong matching and rescaling the wrong size. The computation time (as seen in Figure 5.12) depends on the size and the shape of the 3-D reconstruction. Most of the models had the same trend of the computation time that was increasing when the printing layers was increasing except the prism model because it could not reconstruct 3-D model. The sun gear model is the largest size, so the computation time for each pair of cameras took longer than other models (i.e. (~170 seconds per pair). It took about 510 seconds to detect an error for three pair of sun gear images.

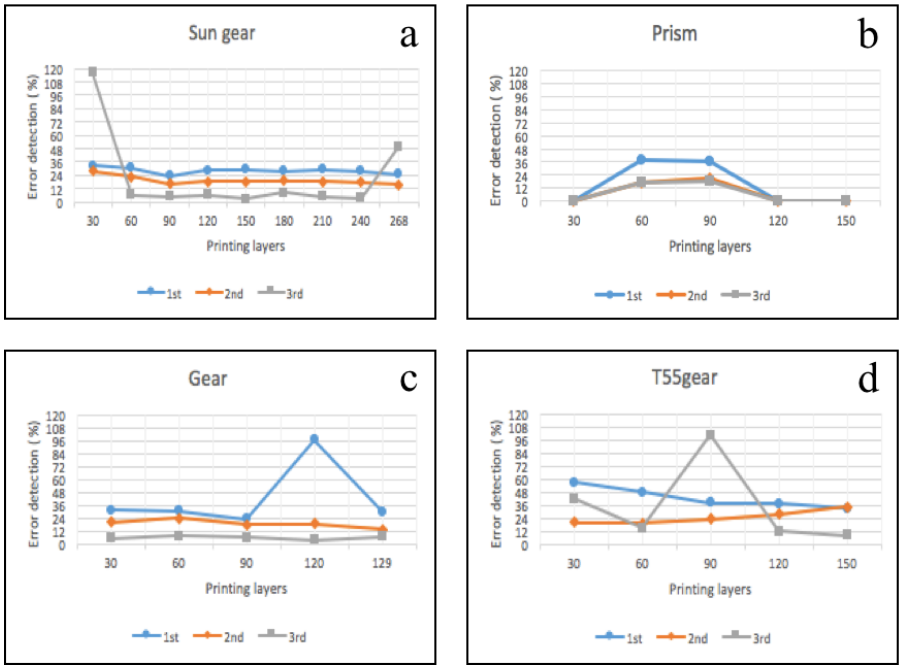


Figure 5.11 Image pre-processing - SIFT and RANSAC to rescale and rectification: the error detection of normal printing state for a) sun gear, b) prism, c) gear, and d) t55gear.

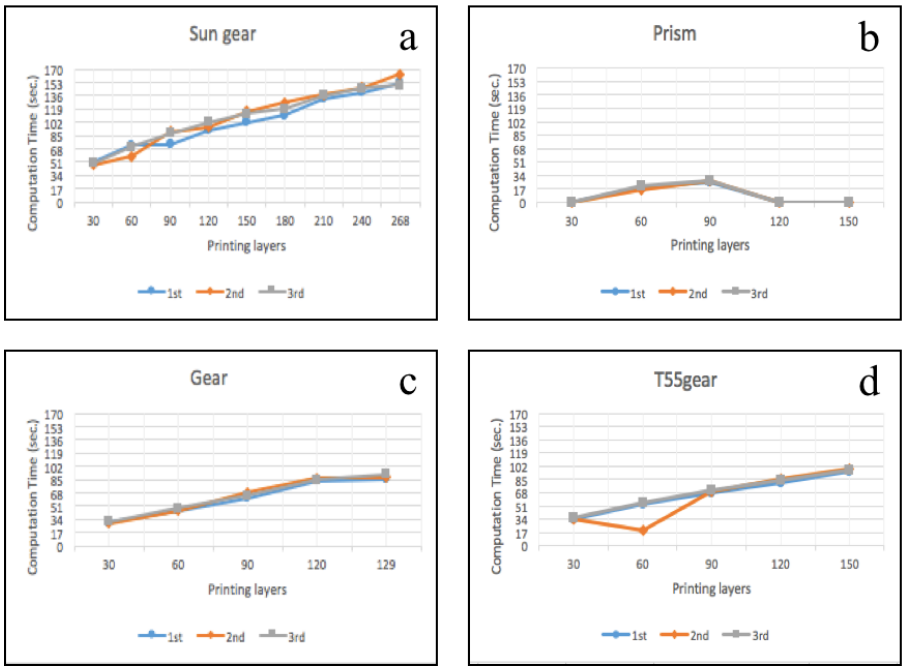


Figure 5.12 Image pre-processing - SIFT and RANSAC to rescale and rectification: the computational time of normal printing state for a) sun gear, b) Prism, c) gear, and d) t55gear.

B) Failure State

Figure 5.13 shows that most of the errors are greater than 10% for each geometry except the third pair of the sun gear model after 90 layers, and the third pair of images in the gear model for all cases that the errors are less than 10%. The computation time (as seen in Figure 5.14) had the same trend as the normal printing state.

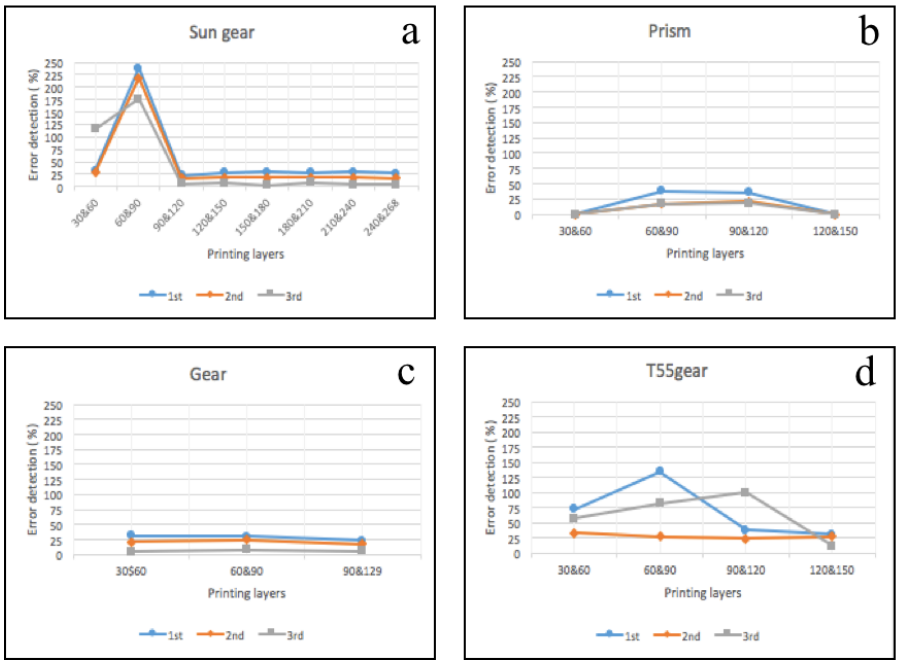


Figure 5.13 Image pre-processing - SIFT and RANSAC to rescale and rectification: the error detection of failure state for a) sun gear, b) Prism, c) gear, and d) t55gear.

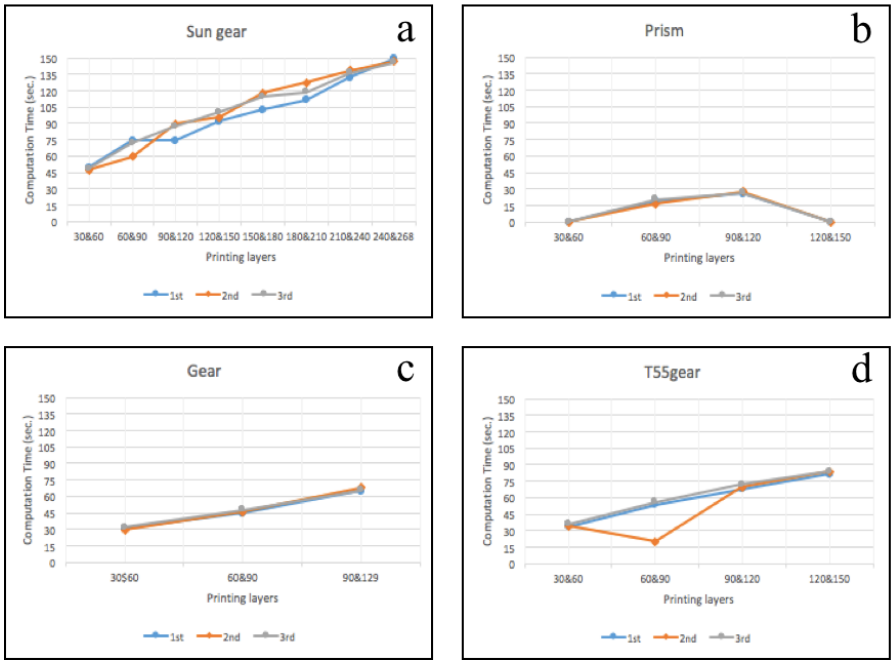


Figure 5.14 Image pre-processing - SIFT and RANSAC to rescale and rectification: the computational time of failure state for a) sun gear, b) Prism, c) gear, and d) t55gear.

5.4.1.2 Non-rescale and rectification

A) Normal Printing State

Figure 5.15 shows the error of all models are less than 10%. The computation time (as seen in Figure 5.16) depends on the size and the shape of the 3-D reconstruction. Most of the models had the same trend of the computation time that was increasing when the printing layers was increasing. The sun gear model is the largest size, so the computation time for each pair of camera took longer than other models, and it took around 100 seconds for each pair. It took about 300 seconds to detect an error for all three pair of sun gear images. On the other hand, the prism gear is the smallest size, so the total computation time for all three pair of images took only 60 seconds to calculate the errors.

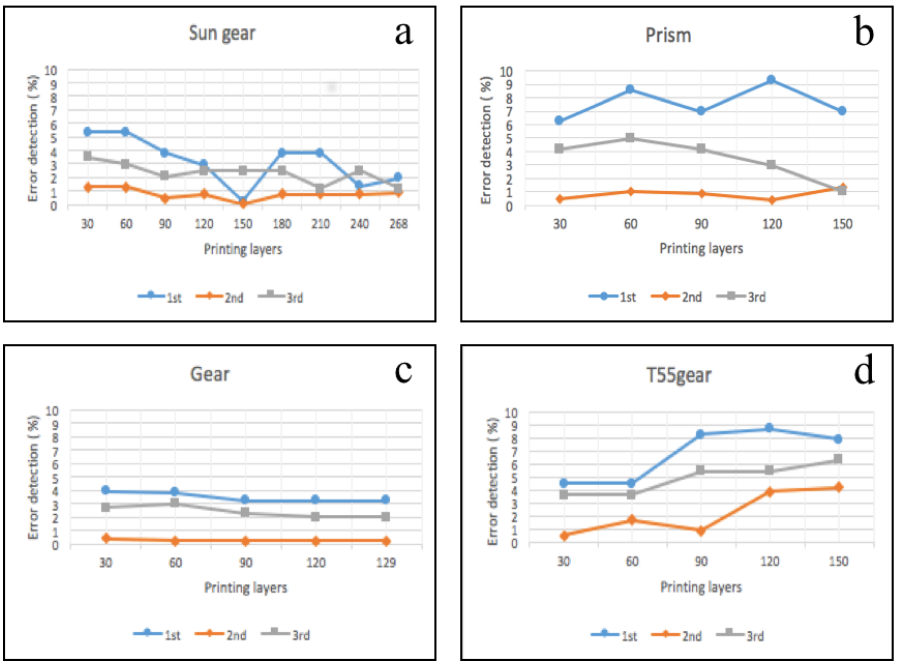


Figure 5.15 Image pre-processing – Non-rescale and rectification: the error detection of normal printing state for a) sun gear, b) Prism, c) gear, and d) t55gear.

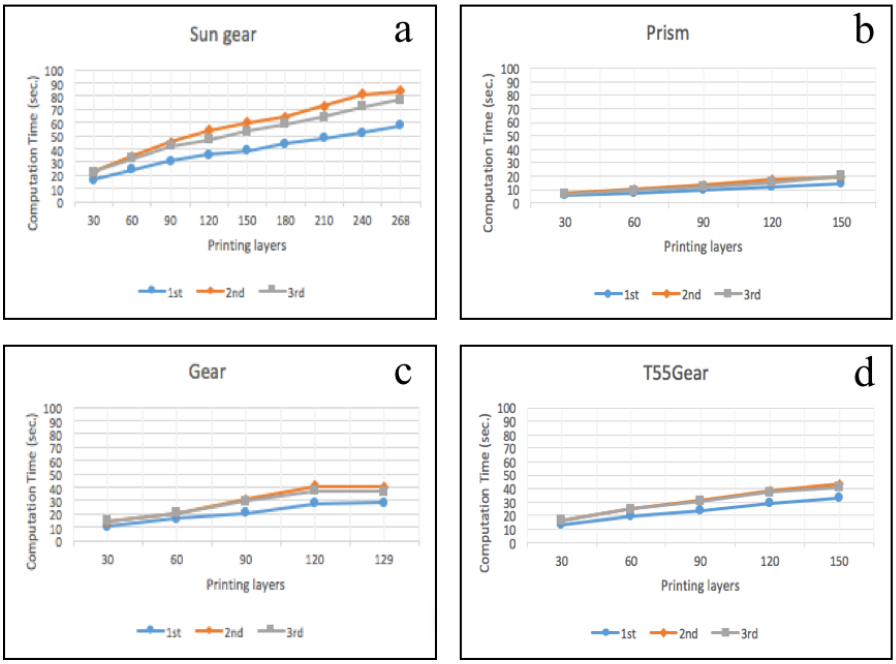


Figure 5.16 Image pre-processing – Non-rescale and rectification: the computation time of normal printing state for a) sun gear, b) prism, c) gear, and d) t55gear.

B) Failure State

Figure 5.17 shows that most of errors are greater than 10% except some layers of the sun gear model in the third pair of the images are less than 10%. The computation time (as seen in Figure 5.18) trends are similar to the case A in the single camera setup.

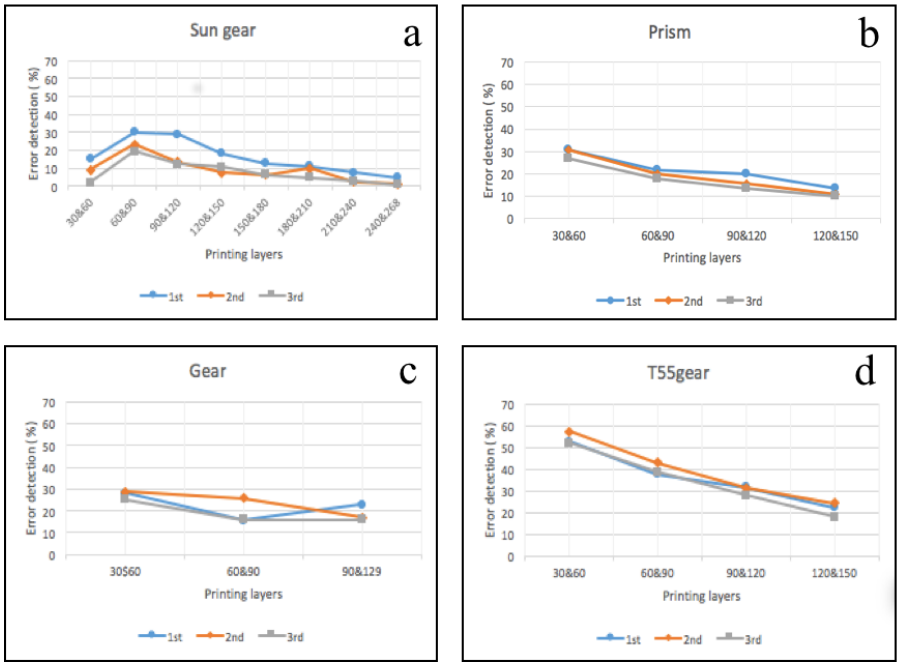


Figure 5.17 Image pre-processing – Non-rescale and rectification: the error detection of failure state for a) sun gear, b) prism, c) gear, and d) t55gear.

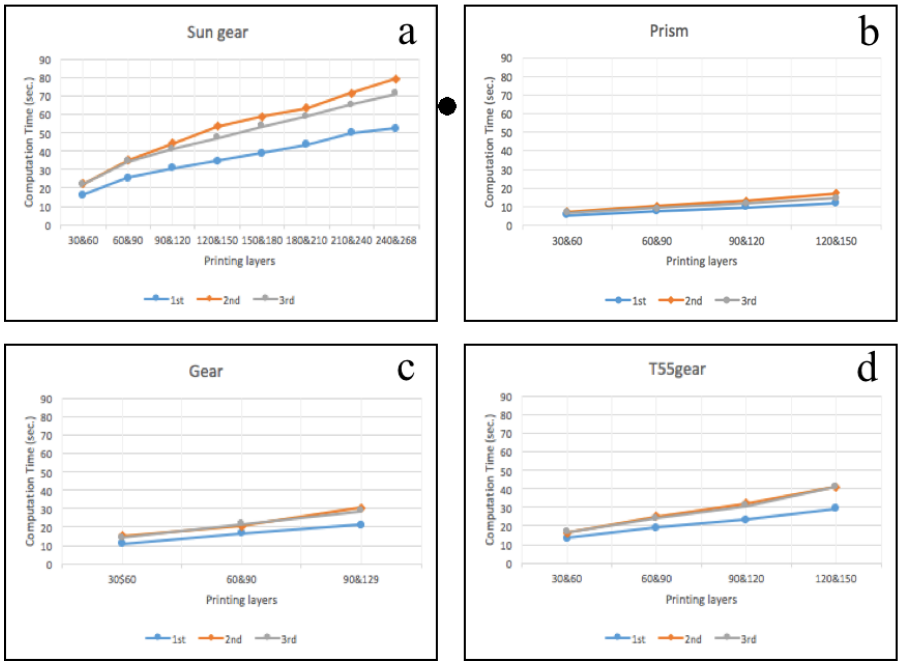


Figure 5.18 Image pre-processing – Non-rescale and rectification: the computation time of failure state for a) sun gear, b) prism, c) gear, and d) t55gear.

5.4.2 Error Detection

From image pre-processing experiment shows that the non-rescale and rectification technique can detect an error more accurately than the SITF and RANSAC to rescale and rectification method. The error detection method needs to be improved here and tested with horizontal magnitude, and horizontal and vertical magnitude.

5.4.2.1 Horizontal Magnitude

This results are the same as the image pre-processing experiment for non- rescale and rectification technique for both normal and failure state.

5.4.2.2 Horizontal and Vertical Magnitude

A) Normal Printing State

Figure 5.19 shows that all errors are less than 10% for each geometry. The computation time (as seen in Figure 5.20) depends on the size and the shape of the 3-D reconstruction. The computation time trends are similar as the horizontal magnitude method.

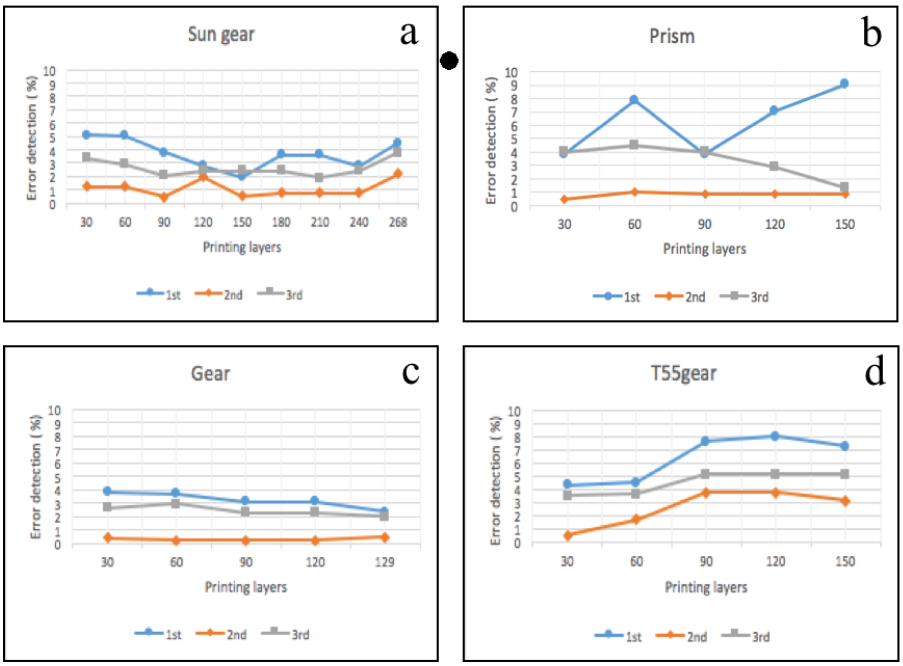


Figure 5.19 Error detection – Horizontal magnitude: the error detection of normal printing state for a) sun gear, b) Prism, c) gear, and d) t55gear.

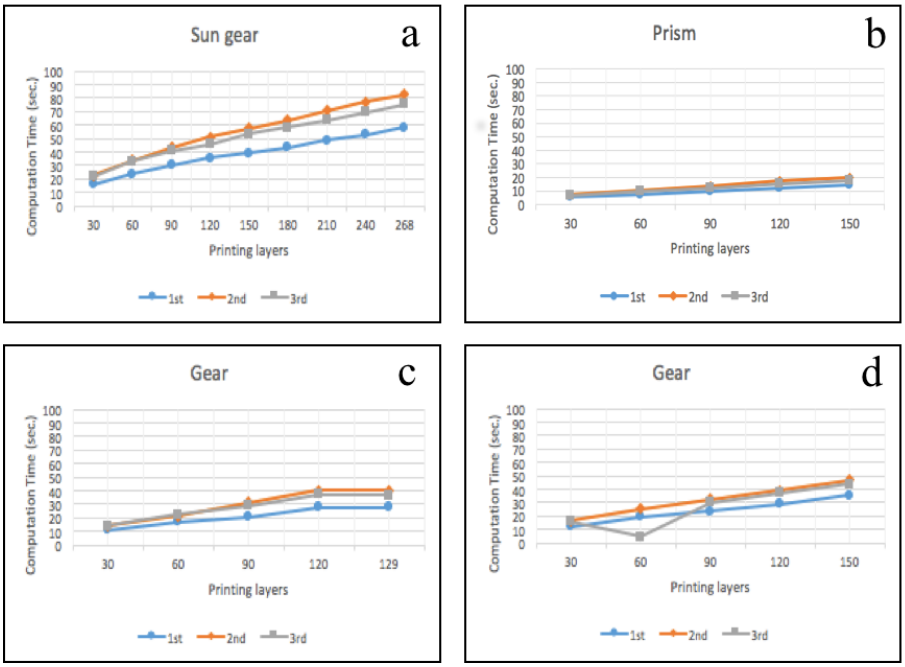


Figure 5.20 Error detection – Horizontal magnitude: the computation time of normal printing state for a) sun gear, b) Prism, c) gear, and d) t55gear.

B) Failure State

All cases correctly are 100% error. The computation time as shown in Figure 5.21 depends on the size and the shape of the 3-D reconstruction similar as the failure state of the non-rescale and rectification in the image pre-processing experiment in Figure 5.18.

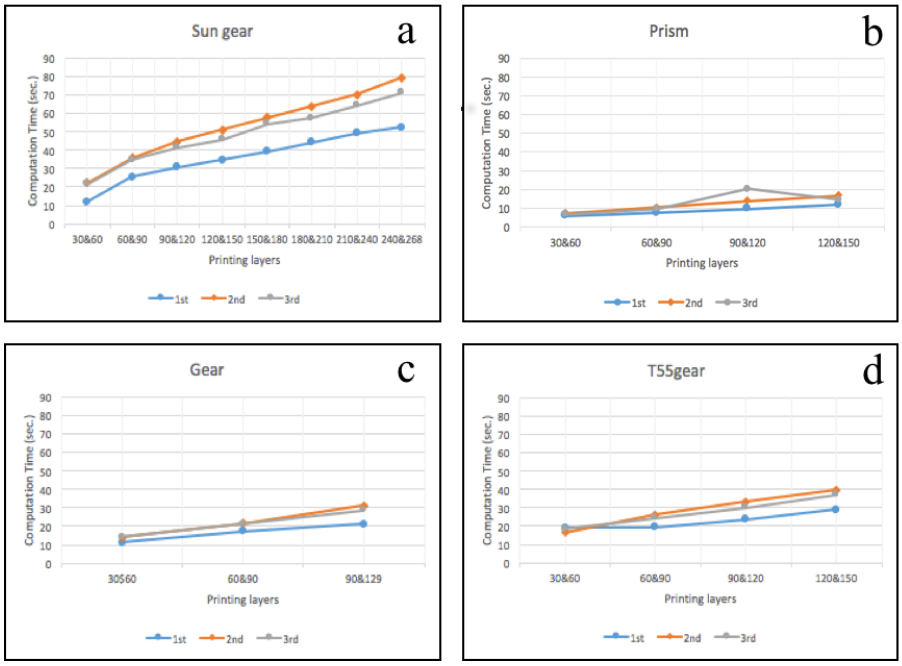


Figure 5.21 Error detection – Horizontal and vertical magnitude: the computation time of failure state for a) sun gear, b) Prism, c) gear, and d) t55gear.

The summary of the image pre-processing experiment for SIFT and RANSAC to rescale and rectification, and non-rescale and rectification method for both normal printing and failure state are shown in Figure 5.22 to 5.25. In the normal printing state, the non- rescale and rectification method is better than SIFT and RANSAC to rescale and rectification method for both the percentage of error and computation time. It can detect an error more accurate than SIFT and RANSAC to rescale and rectification method for all models as shown in Figure 5.22. But both methods are fail to detect the failure state as shown in

Figure 5.24. The computation time for both normal state and failure state of non-rescale and rectification method is 2X faster than SIFT and RANSAC to rescale and rectification method for all models as shown in Figure 5.23 and 5.25.

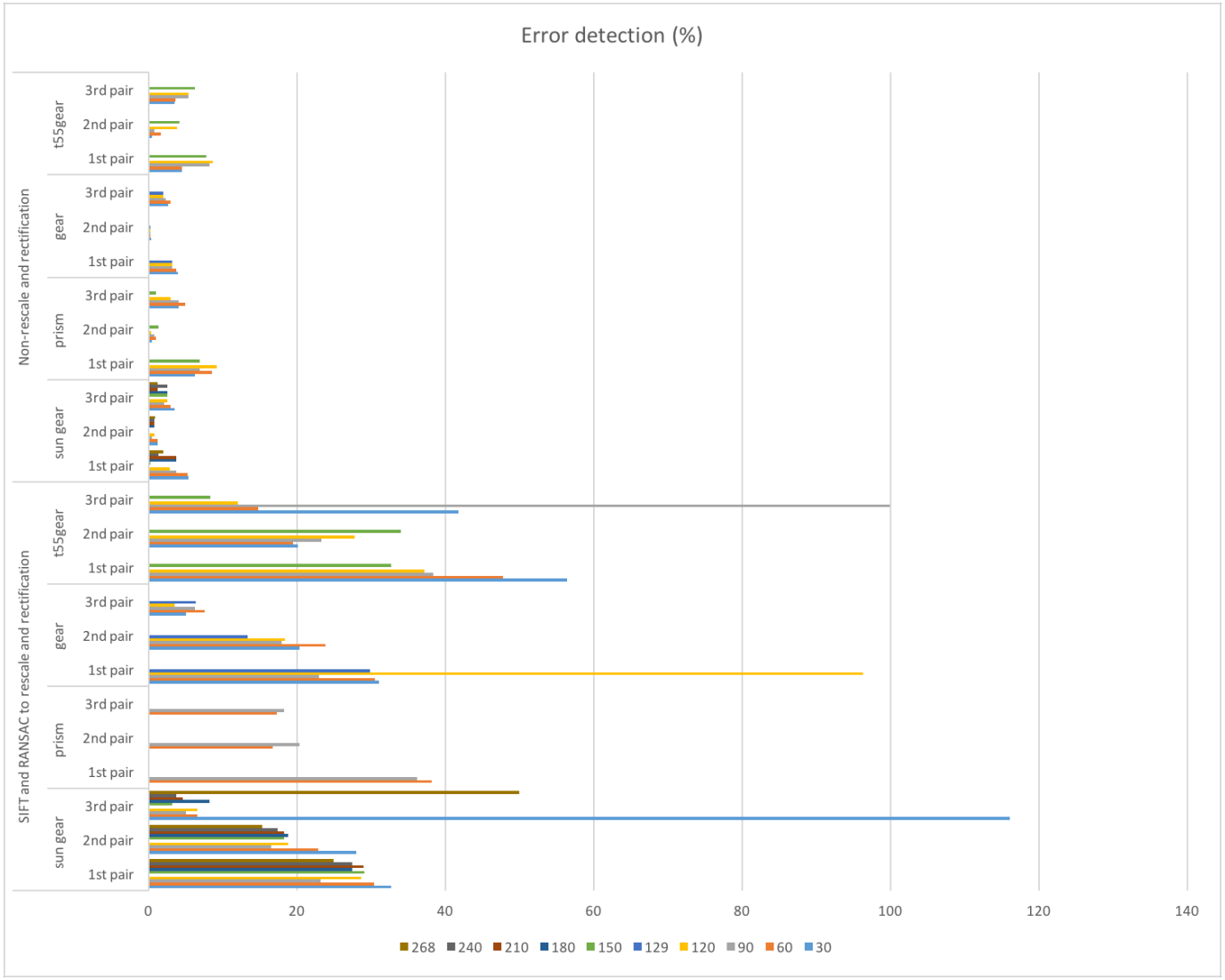


Figure 5.22 Summary of image pre-processing: the error detection of normal printing state for a) sun gear, b) prism, c) gear, and d) t55gear.

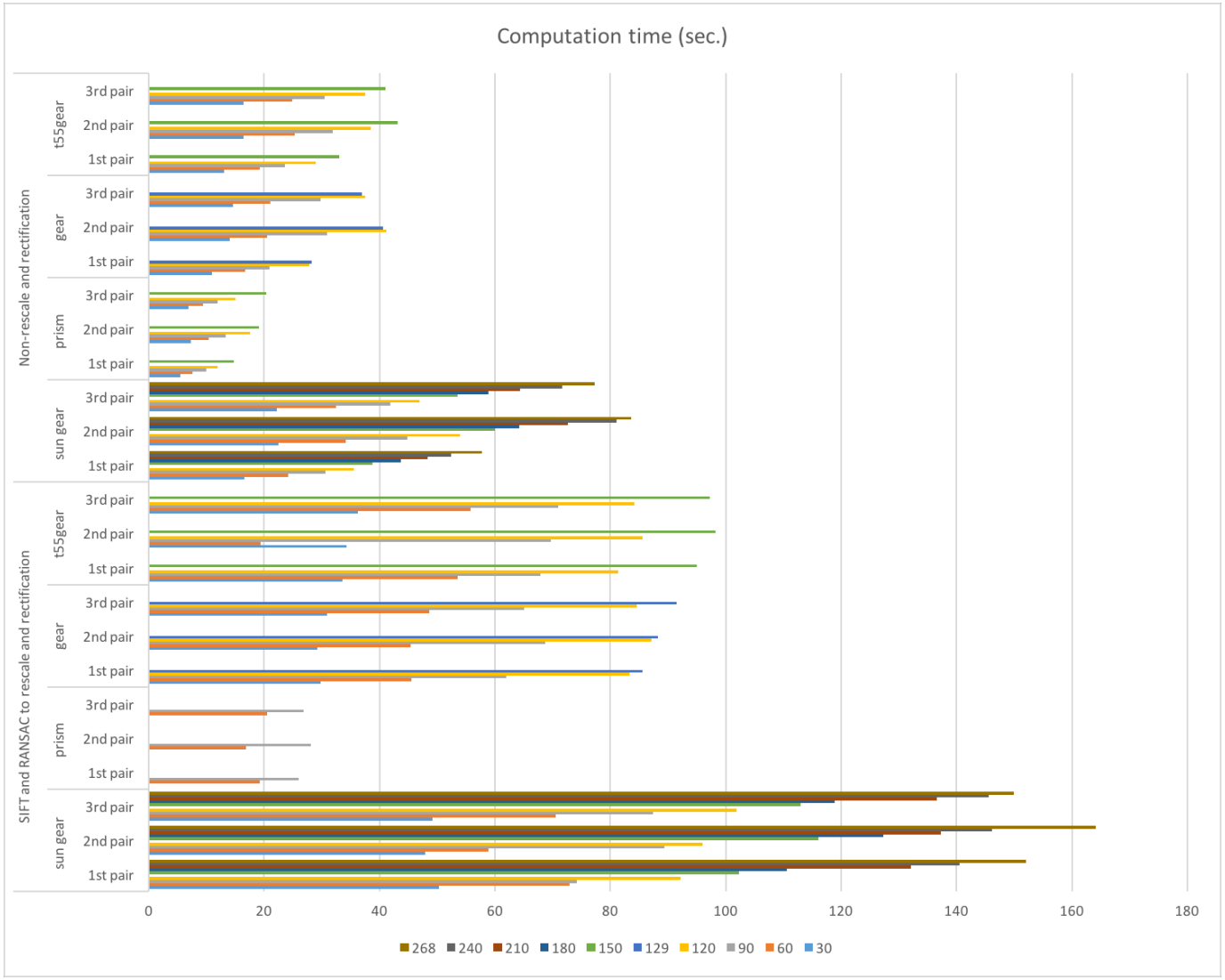


Figure 5.23 Summary of image pre-processing: the computation time of normal printing state for a) sun gear, b) prism, c) gear, and d) t55gear.

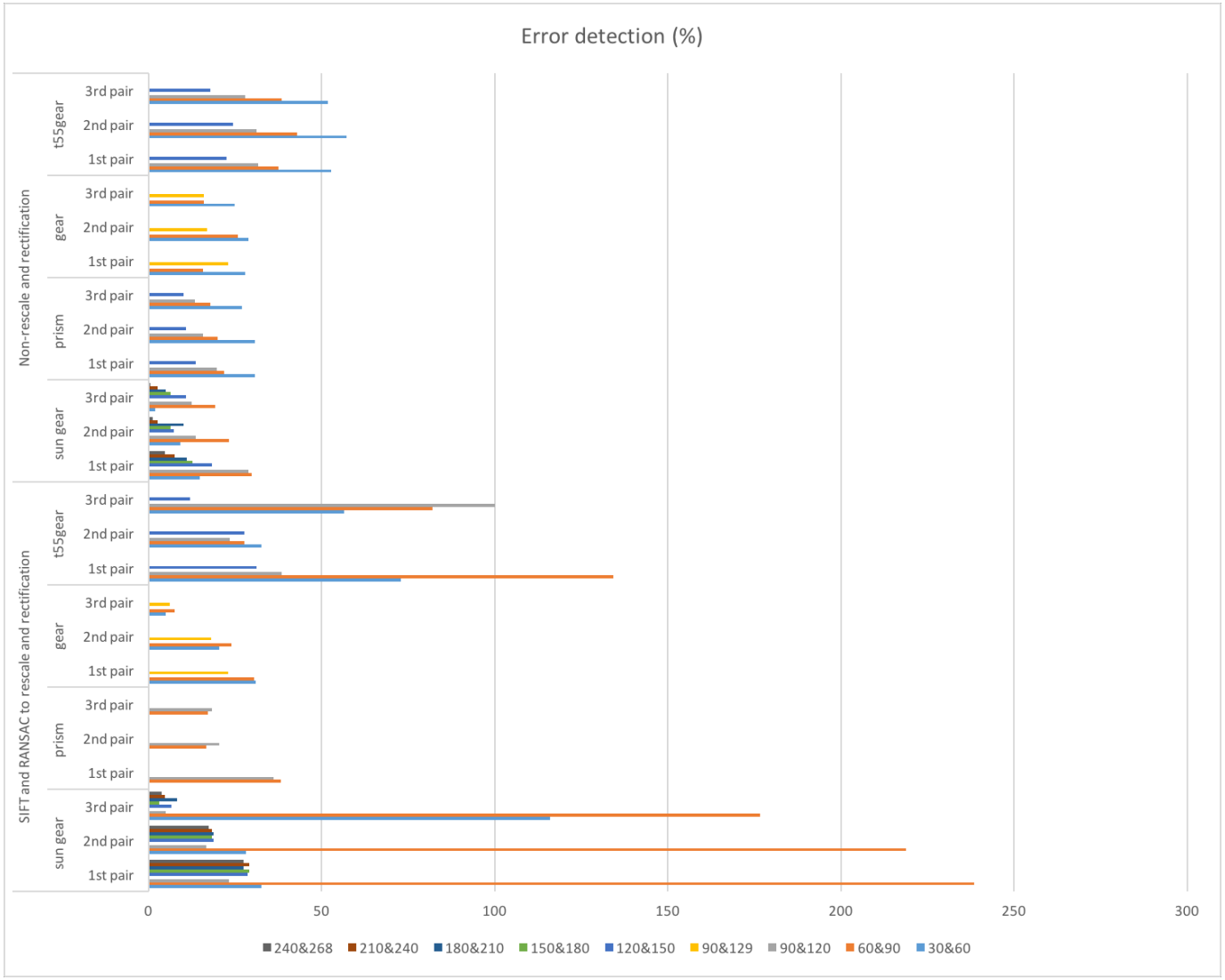


Figure 5.24 Summary of image pre-processing: the error detection of failure state for a) sun gear, b) prism, c) gear, and d) t55gear.

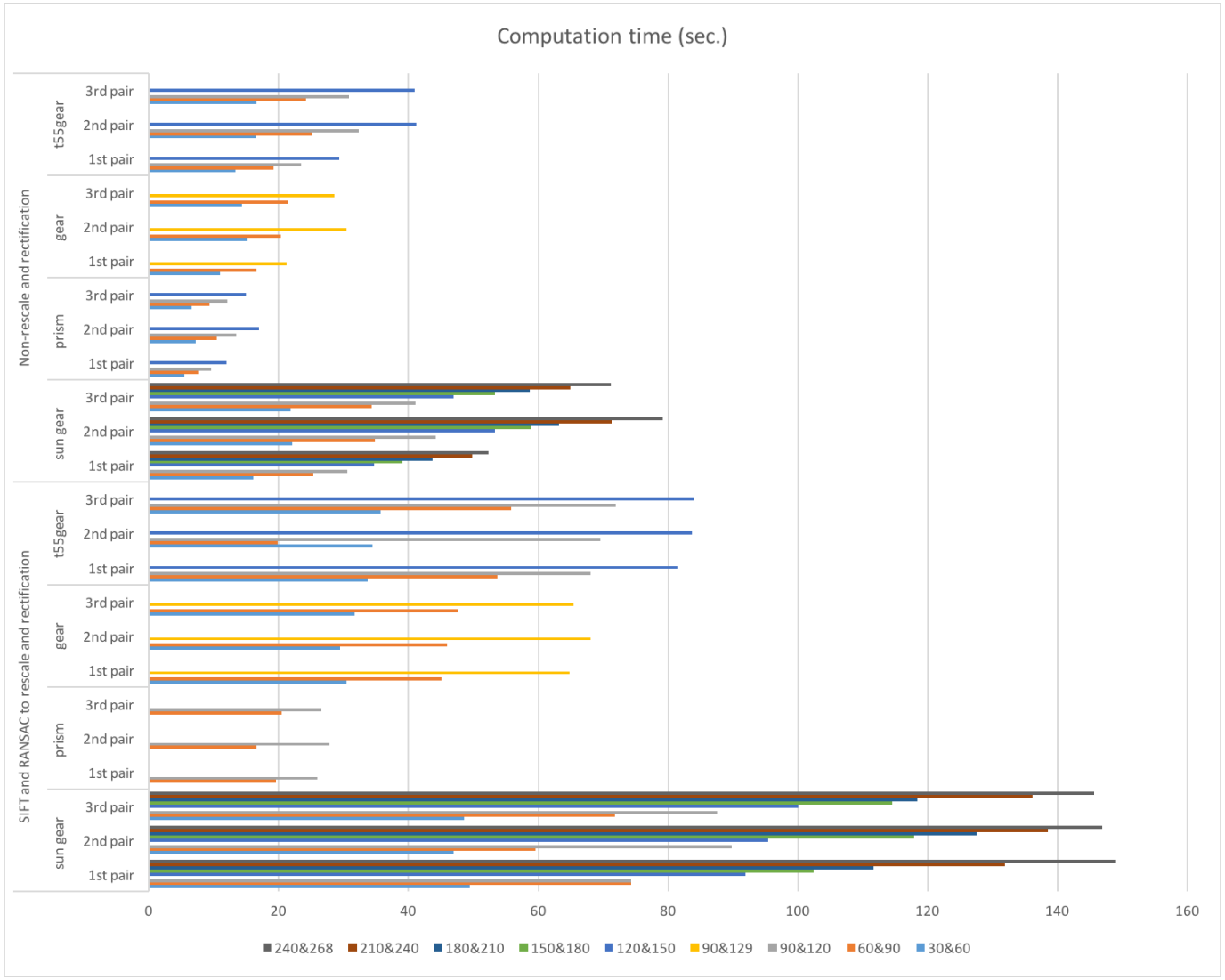


Figure 5.25 Summary of image pre-processing: the computation time of failure state for a) sun gear, b) prism, c) gear, and d) t55gear.

The summary of the error detection experiment for horizontal magnitude, and horizontal and vertical magnitude for both normal printing and failure state are shown in Figure 5.26 to 5.29. In normal printing state, both horizontal magnitude, and horizontal and vertical magnitude can detect error correctly under 10% as shown in Figure 5.26. But in the failure state, the horizontal and vertical magnitude can detect the failure more accurate than the horizontal magnitude for all models by reporting 100% error as shown in Figure 5.28. Also the computation time are the same in both normal printing and failure state as shown in Figure 5.27 and 5.29.

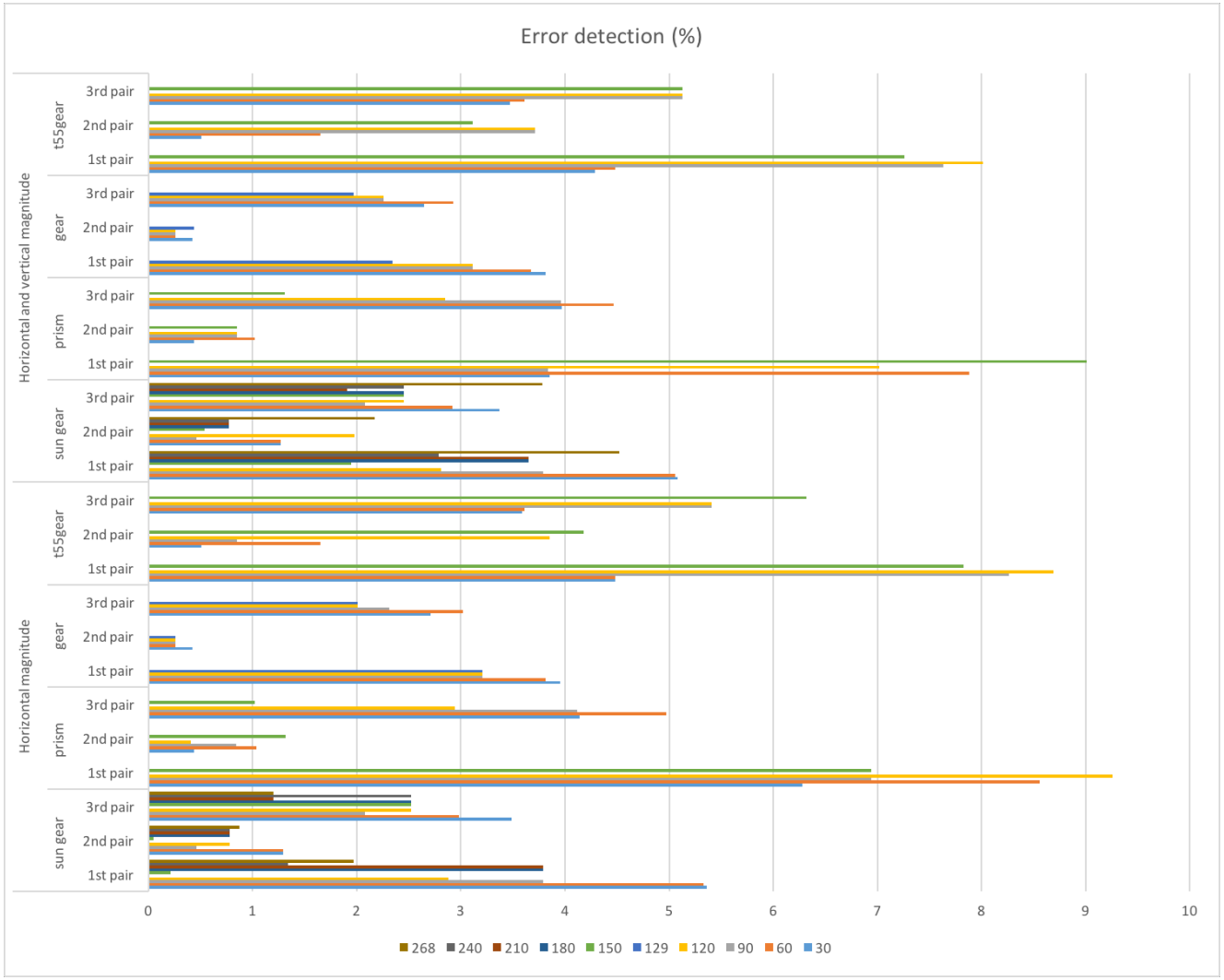


Figure 5.26 Summary of error detection: the error detection of normal printing state for a) sun gear, b) prism, c) gear, and d) t55gear.

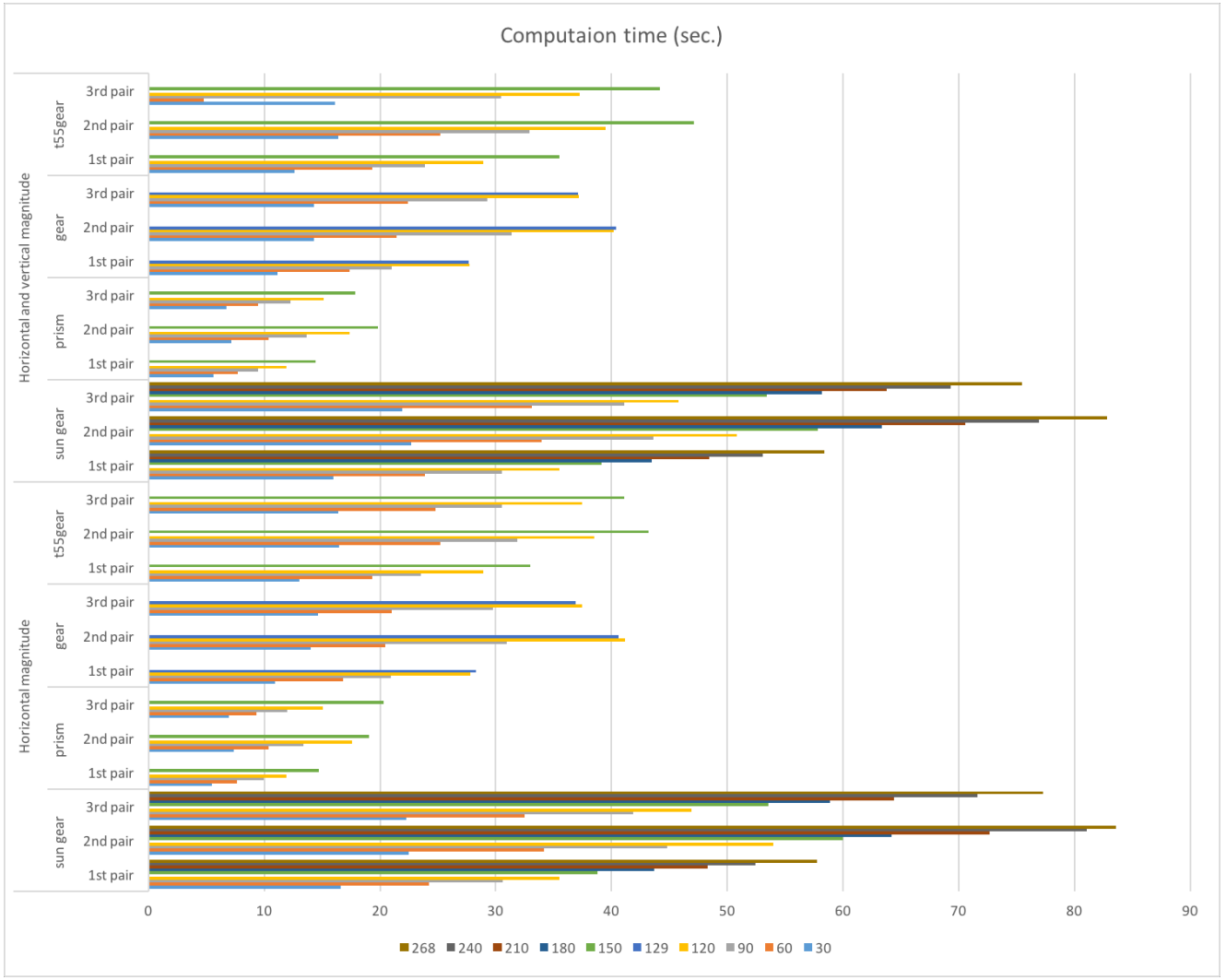


Figure 5.27 Summary of error detection: the computation time of normal printing state for a) sun gear, b) prism, c) gear, and d) t55gear.

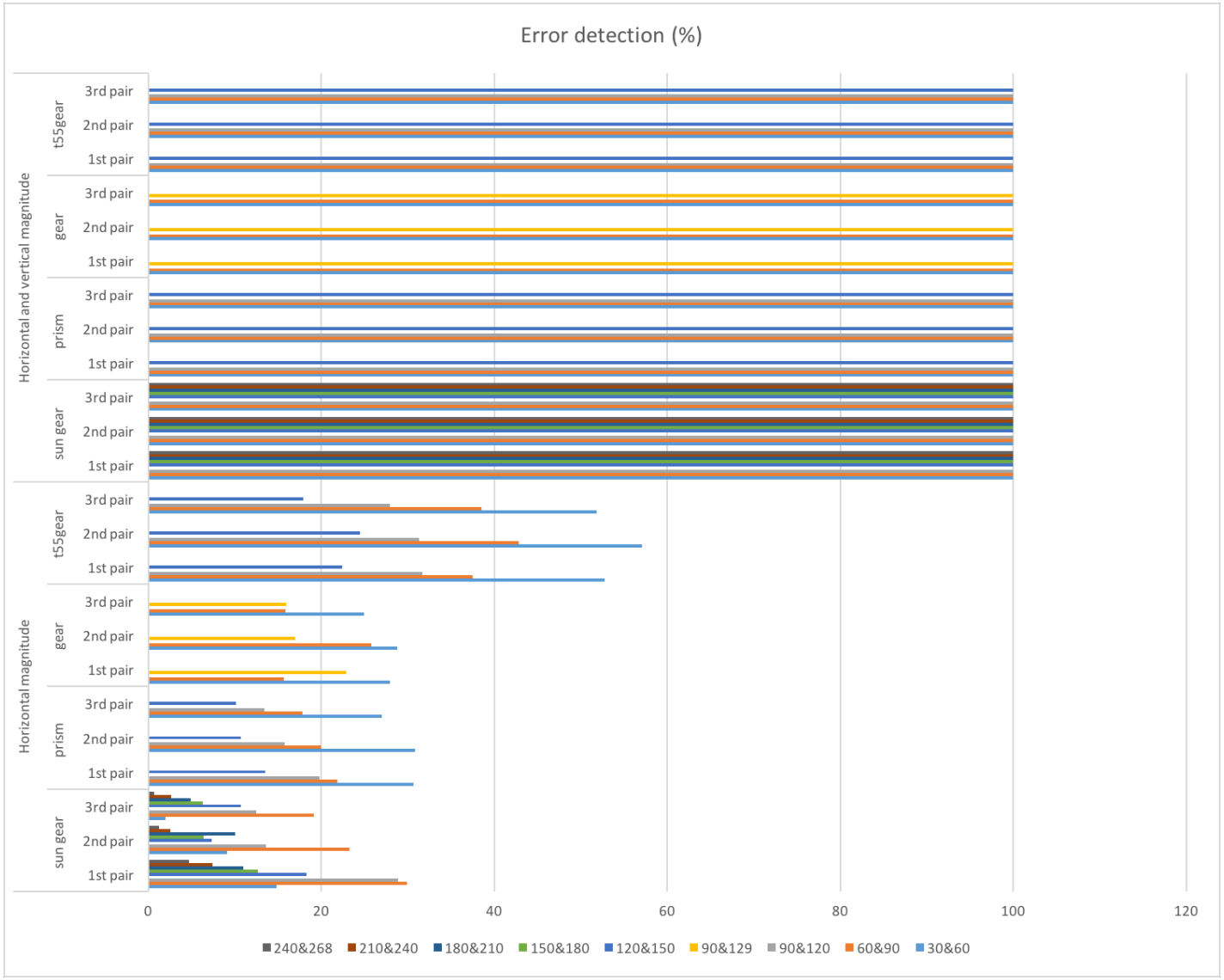


Figure 5.28 Summary of error detection: the error detection of failure state for a) sun gear, b) prism, c) gear, and d) t55gear.

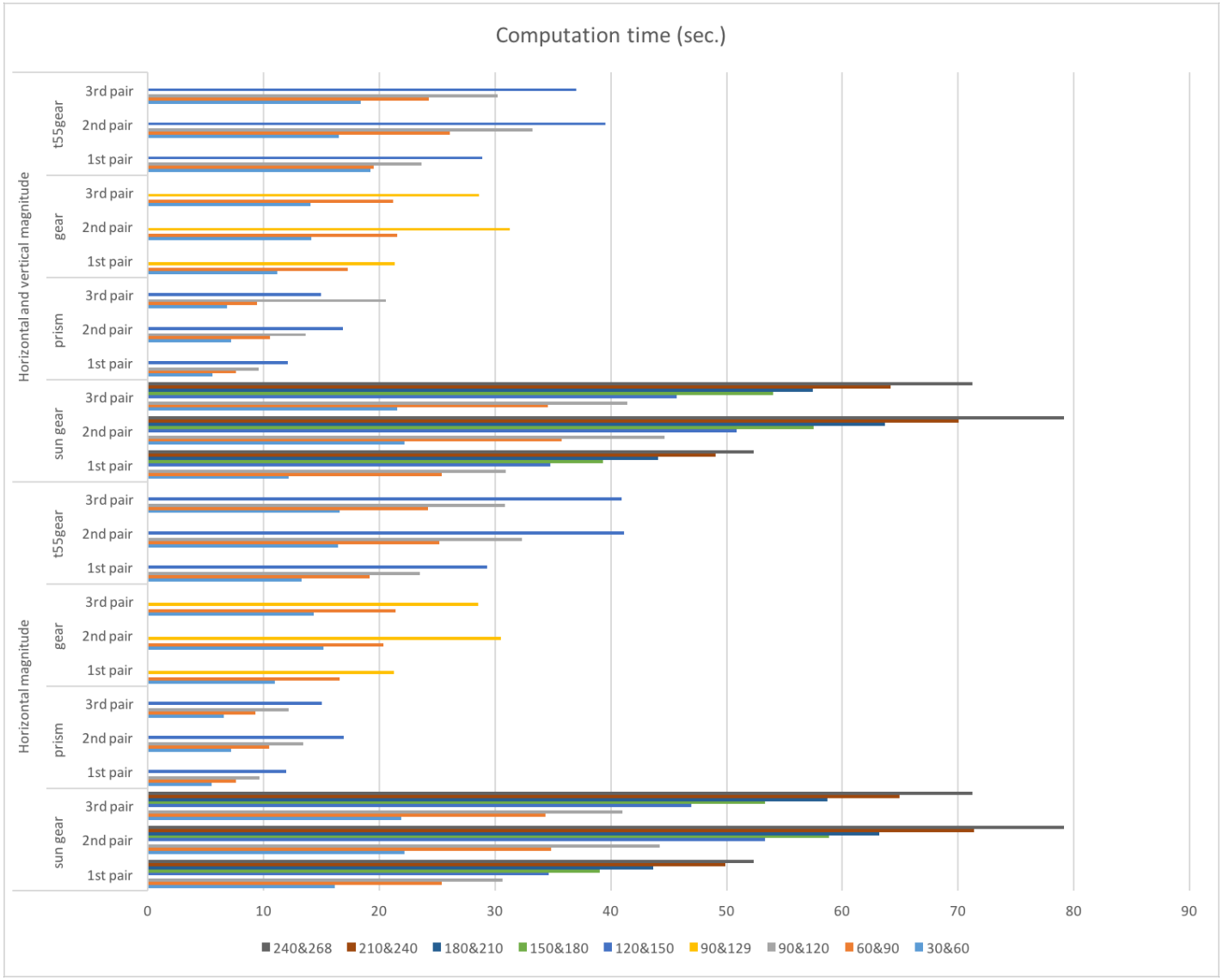


Figure 5.29 Summary of error detection: the computation time of failure state for a) sun gear, b) prism, c) gear, and d) t55gear.

5.5 Discussion

The experimental results show that the three double camera set up in Python can be used to automatically detect a 3-D printer error such as clogged extruder, loss of filament, or an incomplete project for a wide range of 3-D object geometries. These errors can be significant as new user RepRap printing has been shown to have a 20% failure rate (Wittbrodt, et al., 2013). Previous solutions depended on proprietary software and expensive hardware. This work has overcome the limitations (Nuchitprasitchai, et al., 2016; Nuchitprasitchai, et al., 2017) by reducing the computation time for multiple cameras and reducing the cost of software. The computation time here for the similar area size of ROI using Python is around 2X faster and less expensive than the code (Nuchitprasitchai, et al.., 2016; Nuchitprasitchai, et al., 2017) with the same algorithm run in the MATLAB environment which costs $2,150 (Mathworks, 2016). This is not that expensive for research or highend 3-D printer applications, but represents a barrier to deployment in the low-cost prosumer printers used for distributed manufacturing, which generally cost in total $2,500 or less (the RepRap used in this study was $500 in parts).

The double error detection works as designed. It should be noted, that a printed 3-D object usually has a small error when compared between the 3-D model file and the real 3-D printed object. The image pre-processing with horizontal magnitude error detection experiment shows that the algorithm with non-rescale and rectification can detect when the printing has failed more accurately than the one using the SIFT and RANSAC to rescale and rectification. But the error detection using horizontal magnitude results in sun gear model are not correct in some layers such as layers between 210 and 240, or between 240

and 268 in the first pair of cameras are less than 10% in failure state that should be greater than 10%. Therefore, the non-rescale and rectification algorithm was used in the error detection experiment with two different methods: horizontal magnitude, and horizontal and vertical magnitude. The horizontal and vertical magnitude method showed that the 3-D reconstruction error detection can detect 100% error when the printing has failed because the 3-D printed objects are smaller than the STL models because there are no data at the current height of the printing. The use of web cameras can be less expensive than other methods which are more accurate error detection of a 3-D print such as a laser scanning or sensor (Faes, et al., 2014), or scientific research cameras that cost about US$300 (Nuchitprasitchai, et al., 2016; Nuchitprasitchai, et al., 2017). There are other methods to stop catastrophic failures. For example, there is a thrown rod alarm system for delta-style RepRaps, which alerts user when electrical connections are broken if any of the linking rods lose connection with the end effector (hot end) (Nuchitprasitchai, 2017) and Barker developed a similar thrown rod halt mod, which stops a print when electrical connections are broken if any of the linking rods are thrown (Barker, 2017). This type of warning system only addresses one failure mode while the work described here stops printing for any failure mode. Others demand user oversight as (Mahan, 2016; Gewirtz, 2016; Printer 3-D, 2016; Carmelito, 2016; Simon, 2014; KenVersus, 2015), while the system described here is automatic This double cameras error detection algorithm (100% detection) can also detect the error better than vision-based error detection for 3-D printing processes when missing material flow (80% detection) (Baumann and Dieter, 2016). However, the algorithm here still has limitations. First, slicing the STL model into every N layers cannot be done for some number layers that the user may want because Slic3r reports an error for

removing a facet. For example, the t55gear model used here could not be sliced every 10 or 20 layers, which is why here tested in every 30 layers. Second for 3-D printing models that create too many shadows in the model after taking the images can also not be monitored in this way. In the removing background process, such models lose a lot of data of the bottom of the object in the image caused a false error detection. Thus, the geometries that this process works for is limited. Finally, for users setting up the systems for themselves web cameras must be selected with a focal length of 10 cm or longer and must be supported by the open source environment.

From the previous work (Nuchitprasitchai, 2017), the images from the single camera set up can be processed to detect the shape error in a low-cost 3-D printing, and the detection rate for both normal printing and failure state are 100% correctly. The computation time of the single camera set up is fast, less than 10 seconds for all three cameras. Also, this work represented reconstructing 3-D images of 3-D objects from 2-D images that successfully used to detect the size error of failure printing by six cameras. The computation time of the double camera set up depends on the size of the 3-D model. In this experiment, the average of the computation time is 45 seconds for each pair of cameras. Therefore, the single and double camera setup in an open source algorithm have been used together for more efficiency in reliable monitoring error of FFF-based 3-D printing in shape and size.

In addition, to overcoming these limitations there are several other areas of future research. First, the slicing STL model process need to be investigated to eliminate the error for removing a facet. Second, removing the background algorithm need to be more accurate to remove only noise in the. Furthermore, to increase the quality of removing the background,

the new mathematical equations need to be tested for the performance of the system. Third, the computation time of this system would be improved if the 3-D reconstruction process is calculated only on the new area of the 3-D printed part. For example, the STL model is sliced every 30 layers. The first 3-D reconstruction is for layer 1 to 30, then the next 3-D reconstruction should be only for layer 31 to 60. This will reduce the area of pixels need to be calculated. Last, this system may be tested with other block matching algorithms to see if another algorithm is faster and more accurate such as correlation coefficient, normalized correlation coefficient, cross correlation, normalized cross correlation, squared difference, or normalized squared difference (Abidrahmank, 2013). Last, Franklin need to be modified to include this algorithm in order to alert user and pause the printing when an error occurs.

5.6 Conclusions

This paper described an open-source low-cost reliable real-time monitoring platform for FFF-based 3-D printing based on a double cameras system for three perspectives around 360 degrees. The results showed that the algorithm using non-rescale and rectification with detecting an error at the current height of the printing was effective at detecting a clogged nozzle, loss of filament, or an incomplete project for a wide range of 3-D object geometries. The error calculations were determined from the data in the 3-D reconstruction points at the current height of the printing. The error results can be used to inform user and as the feedback control for the printer. The validity of this approach using experiment shows that the error detection system is capable of a 100 percent detection rate for failure detection.

5.7 References

1. Abidrahmank, 2014. OpenCV2-Python-Tutorials [WWW Document]. URL https://github.com/abidrahmank/OpenCV2-Python-Tutorials/ (accessed 3.30.17).
2. Anzalone, G.C., Wijnen, B., Pearce, J.M., 2015. Multi-material additive and subtractive prosumer digi-tal fabrication with a free and open-source convertible delta RepRap 3-D printer. In Rapid Proto-typing Journal. 21, 506–519. doi:10.1108/RPJ-09-2014-0113
3. Anzalone, G., Wijnen, B., Pearce, J.M., 2016. Delta Build Overview:MOST - Appropedia: The sustain-ability wiki [WWW Document]. URL http://www.appropedia.org/Delta_Build_Overview:MOST (accessed 6.13.16).
4. Atli, A.V., Urhan, O., Ertürk, S., Sönmez, M., 2006. A computer vision-based fast approach to drilling tool condition monitoring.In Proceedings of the Institution of Mechanical Engineers, Part B: Jour-nal of Engineering Manufacture. 220, 1409–1415. doi:10.1243/0954
5. Baden, T., Chagas, A.M., Gage, G.J., Marzullo, T.C., Prieto-Godino, L.L., Euler, T., 2015. Correction: open labware: 3-d printing your own lab equipment. In PLOS Biology. 13, e1002175. doi:10.1371/journal.pbio.1002175
6. Barker, B., 2014. Thrown Rod Halt Mod - Appropedia: The sustainability wiki [WWW Document]. URL http://www.appropedia.org/Thrown_Rod_Halt_Mod (accessed 3.20.17).
7. Baumann, F., Roller, D., 2016. Vision based error detection for 3D printing processes. In EDP Science 59
8. Birtchnell, T., Hoyle, W., 2014. 3D printing for development in the global south: The 3D4D challenge. In Palgrave Macmillan. Doi:http://dx.doi.org/10.1057/9781137365668.
9. Bowyer, A., 2014. 3D Printing and Humanity's First Imperfect Replicator. 3D Print. In Additive Man-ufacturing. 1, 4–5. doi:10.1089/3dp.2013.0003
10. Bradley, C., Wong, Y.S., 2001. Surface texture indicators of tool wear - a machine vision approach. In The international Journal of Advanced Manufacturing Technology. 17, 435–443. doi:10.1007/s001700170161
11. Bradski, G., Kaehler, A., 2008. Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc.
12. Campbell, I., Bourell, D., Gibson, I., 2012. Additive manufacturing: rapid prototyping comes of age. In Rapid Prototyping Journal. 18, 255–258. doi:10.1108/13552541211231563
13. Gibson, I., Rosen, D. and Stucker, B., 2014. Additive manufacturing technologies: 3D printing, rapid prototyping, and direct digital manufacturing. In Springer.

14. Carmelito, 2016. Controlling and Monitoring your 3D printer with... | element14 | MusicTech [WWW Document]. URL https://www.element14.com/community/community/design-challenges/musictech/blog/2016/03/16/controlling-your-3d-printer-with-beaglebone-and-octoprint (accessed 3.18.17).
15. Coakley, M., Hurt, D.E., 2016. 3D Printing in the Laboratory. In Journal Laboratory Automation. 21, 489–495. doi:10.1177/2211068216649578
16. Concept Laser, 2016. Metal Additive Manufacturing Machines [WWW Document]. URL http://www.conceptlaserinc.com/ (accessed 11.10.16).
17. Dollar Tree, Inc., 2016. Floral Supplies, Party Supplies, Cleaning Supplies [WWW Document]. URL https://www.dollartree.com/ (accessed 12.3.16).
18. Droffarts, 2012. Parametric pulley - lots of tooth profiles by droftarts - Thingiverse [WWW Document]. URL http://www.thingiverse.com/thing:16627 (accessed 3.12.17).
19. Edinbarough, I., Balderas, R., Bose, S., 2005. A vision and robot based on-line inspection monitoring system for electronic manufacturing. In Computer in Industry. 56, 986–996. doi:10.1016/j.compind.2005.05.022
20. Faes, M., Abbeloos, W., Vogeler, F., Valkenaers, H., Coppens, K., Goedemé, T., Ferraris, E., 2014. In Process Monitoring of Extrusion Based 3D Printing via Laser Scanning. 6, 363-367. doi:10.13140/2.1.5175.0081
21. Fischler, M.A., Bolles, R.C., 1981. Random sample consensus: a paradigm for model fitting with appli-cations to image analysis and automated cartography. In Communications of the ACM. 24(6), 381–395. doi:10.1145/358669.358692
22. Fox, S., 2010. After the factory [Manufacturing renewal]. In Engineering & Technology, 5(8), pp.59-61.
23. Gewirtz, D., 2016. Adding a Raspberry Pi case and a camera to your LulzBot Mini - Watch Video Online - Watch Latest Ultra HD 4K Videos Online [WWW Document]. URL http://www.zdnet.com/article/3d-printing-hands-on-adding-a-case-and-a-camera-to-the-raspberry-pi-and-lulzbot-mini/ (accessed 11.30.16)
24. Gibb, A., Abadie, S., 2014. Building open source hardware: DIY manufacturing for hackers and mak-ers. In Pearson Education.
25. Gibson, I., Rosen, D., Stucker, B., 2014. Additive manufacturing technologies: 3D printing, rapid pro-totyping, and direct digital manufacturing. In Springer-Verlag New York, New York.
26. Golnabi, H., Asadpour, A., 2007. Design and application of industrial machine vision systems. In Ro-botics and Computer-Integrated Manufacturing. 23, 630–637. doi:10.1016/j.rcim.2007.02.005
27. Gonzalez-Gomez, J., Valero-Gomez, A., Prieto-Moreno, A., Abderrahim, M., 2012. A New Open Source 3D-Printable Mobile Robotic Platform for Education.

In Advances in Autonomous Mini Robots. Springer Berlin Heidelberg, Berlin, Heidelberg. 49–62. doi:10.1007/978-3-642-27482-4_8
28. Hurd, S., Camp, C., White, J., 2015. Quality Assurance in Additive Manufacturing Through Mobile Computing. In Springer, Cham, 203–220. doi:10.1007/978-3-319-29003-4_12
29. Irwin, J.L., Oppliger, D.E., Pearce, J.M., Anzalone, G., 2015. Evaluation of RepRap 3D Printer Work-shops in K-12 STEM. In 122nd ASEE 122nd ASEE Conference Proceedings, Pap. ID 12036.
30. Jetty, 2012. Paper Crimper by jetty - Thingiverse [WWW Document]. URL http://www.thingiverse.com/thing:17634 (accessed 12.3.16).
31. Ji, S., Zhang, X., Zhang, L., WAN, Y., YUAN, J., ZHANG, L., 2002. Application of computer vision in tool condition monitoring. In Journal-Zhejiang University Technology. 30, 143–148.
32. Jones, R., Haufe, P., Sells, E., Iravani, P., Olliver, V., Palmer, C., Bowyer, A., 2011. RepRap – the rep-licating rapid prototyper. In Robotica. 29, 177–191. doi:10.1017/S026357471000069X
33. Kentzer, J., Koch, B., Thiim, M., Jones, R.W. and Villumsen, E., 2011. An open source hardware-based mechatronics project: The replicating rapid 3-D printer. In 4th International Conference on Mecha-tronics (ICOM). IEEE. 1–8. doi:10.1109/ICOM.2011.5937174
34. KenVersus, 2015. Logitech C170 webcam mount for daVinci 3D Printer by KenVersus - Thingiverse [WWW Document]. URL http://www.thingiverse.com/thing:747105 (accessed 3.18.17).
35. Kerr, D., Pengilley, J., Garwood, R., 2006. Assessment and visualisation of machine tool wear using computer vision. In International Journal Advanced Manufacturing Technology. 28, 781–791. doi:10.1007/s00170-004-2420-0
36. Klancnik, S.;, Ficko, J.&, Pahole, I., 2015. Computer Vision-Based Approach to End Mill Tool Moni-toring. In International Journal Simulation Modeling. 14(4), 571–583. doi:10.2507/IJSIMM14(4)1.301
37. Kleszczynski, S., zur Jacobsmühlen, J., Sehrt, J.T., Witt, G., 2012. Error detection in laser beam melting systems by high resolution imaging, In Proceedings of the Solid Freeform Fabrication Symposium.
38. Kleszczynski, S., zur Jacobsmühlen, J., Reinarz, B., Sehrt, J.T., Witt, G., Merhof, D., 2014. Improving process stability of laser beam melting systems, In Proceedings of the Frauenhofer Direct Digital Manufacturing Conference.
39. Lanzetta, M., 2001. A new flexible high-resolution vision sensor for tool condition monitoring. In Jour-nal Materials Processing Technology. 119, 73–82. doi:10.1016/S0924-0136(01)00878-0
40. Laplume, A., Anzalone, G.C., Pearce, J.M., 2016. Open-source, self-replicating 3-D printer factory for small-business manufacturing. In International Journal

Advanced Manufacturing Technology. 85, 633–642. doi:10.1007/s00170-015-7970-9
41. Li, Y., Li, Y.F., Wang, Q.L., Xu, D., Tan, M., 2010. Measurement and defect detection of the weld bead based on online vision inspection. In IEEE Transactions on Instrumentation Measurement. 59, 1841–1849. doi:10.1109/TIM.2009.202822
42. Lowe, D.G., 1999. Object recognition from local scale-invariant features, In Proceedings of the Sev-enth IEEE International Conference on Computer Vision. IEEE. 2, 1150–1157. doi:10.1109/ICCV.1999.790410
43. Mahan, T., 2016. Raspberry Pi Control and Wireless Interface - Appropedia: The sustainability wiki [WWW Document]. URL http://www.appropedia.org/Raspberry_Pi_Control_and_Wireless_Interface (accessed 3.20.17).
44. Make, 2017. 3D Printer Shootout News, Reviews and More | Make: DIY Projects and Ideas for Makers [WWW Document]. URL http://makezine.com/tag/3d-printer-shootout/ (accessed 4.11.17).
45. MathWorks, 2016. Pricing and Licensing - MATLAB & Simulink [WWW Document]. URL https://www.mathworks.com/pricing-licensing.html?intendeduse=comm (accessed 12.8.16).
46. Microsoft, 2016. Learn to Develop with Microsoft Developer Network | MSDN [WWW Document]. URL https://msdn.microsoft.com/en-us/default.aspx (accessed 11.13.16).
47. Nuchitprasitchai, S., Roggemann, M., Pearce, J. An Open Source Algorithm for Reconstruction 3-D images for Low-cost, Reliable Real-time Monitoring of FFF-based 3-D Printing. (to be submitted).
48. Nuchitprasitchai, S., Roggemann, M., Pearce, J. Factors Effecting Real Time Optical Monitoring of Fused Filament 3-D Printing. (to be published).
49. Nuchitprasitchai, S., 2016. Rod alarm - Appropedia: The sustainability wiki [WWW Document]. URL http://www.appropedia.org/Rod_alarm (accessed 3.20.17).
50. Nuchitprasitchai, S., 2017. 3-D models [WWW Document]. URL https://osf.io/utp6g/ (accessed 4.5.17).
51. O'Neill, P.F., Ben Azouz, A., Vázquez, M., Liu, J., Marczak, S., Slouka, Z., Chang, H.C., Diamond, D., Brabazon, D., 2014. Advances in three-dimensional rapid prototyping of microfluidic devices for biological applications. In Biomicrofluidics 8, 52112. doi:10.1063/1.4898632
52. OpenCV, 2016. OpenCV library [WWW Document]. URL http://opencv.org/ (accessed 11.10.16).
53. OpenCV, 2016. Camera Calibration and 3D Reconstruction — OpenCV 2.4.13.2 documentation [WWW Document]. URL

http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#stereobm (accessed 12.3.16).
54. OpenSCAD, 2016. OpenSCAD - The Programmers Solid 3D CAD Modeller [WWW Document]. URL http://www.openscad.org/ (accessed 12.3.16).
55. Pearce, J.M., Anzalone, N.C., Heldt, C.L., 2016. Open-Source Wax RepRap 3-D Printer for Rapid Pro-totyping Paper-Based Microfluidics. In Journal Laboratory Automation. 21, 510–516. doi:10.1177/2211068215624408
56. Pearce, J.M., 2014. Open-source lab: How to build your own hardware and reduce research costs. In Elsevier.
57. Pearce, J.M., 2015. Applications of open source 3-D printing on small farms. In Organic Farming. 1(1), 19–35.
58. Pearce, J.M., 2012. Building Research Equipment with Free, Open-Source Hardware. In Science. 337 (6100), 1303-1304.
59. Pearce, J.M., Blair, C.M., Laciak, K.J., Andrews, R., Nosrat, A., Zelenika-Zovko, I., 2010. 3-D printing of open source appropriate technologies for self-directed sustainable development. In Journal of Sustainable Development. 3(4), 17-29.
60. Petersen, E.E., Pearce, J., 2017. Emergence of Home Manufacturing in the Developed World: Return on Investment for Open-Source 3-D Printers. In Technologies. 5, 7. Doi:10.3390/technology5010007
61. Pfeifer, T., Wiegers, L., 2000. Reliable tool wear monitoring by optimized image and illumination con-trol in machine vision. In Measurement. 28, 209–218. doi:10.1016/S0263-2241(00)00014-2
62. Point Grey, 2017. Point Grey Firefly MV 0.3 MP Color USB 2.0 Research Camera [WWW Document]. URL http://www.trossenrobotics.com/fireflyMV (accessed 4.8.17).
63. Printer3D, 2017. Free IP Camera Monitoring for 3D printer with old webcam usb in 5min - 3D Printers English French & FAQ Wanhao Duplicator D6 Monoprice Maker Ultimate & D4, D5, Duplicator 7 [WWW Document]. URL http://www.printer3d.one/en/forums/topic/free-ip-camera-monitoring-for-3d-printer-with-old-webcam-usb-in-5min/ (accessed 3.18.17).
64. Python, 2016. Welcome to Python.org [WWW Document]. URL https://www.python.org/ (accessed 11.10.16).
65. Raspberry Pi, 2016. Teach, Learn, and Make with Raspberry Pi [WWW Document]. URL https://www.raspberrypi.org/ (accessed 12.3.16).
66. Raymond, E., 1999. The cathedral and the bazaar. Knowledge, Technol. Policy 12, 23–49. doi:10.1007/s12130-999-1026-0
67. Rimock, M., 2015. An Introduction to the Intellectual Property Law Implications of 3D Printing. In Canadian Journal Law and Technology. 13(1).
68. Rostock, 2016. RepRapWiki [WWW Document]. URL http://reprap.org/wiki/Rostock (accessed 11.5.16).

69. Schelly, C., Anzalone, G., Wijnen, B., Pearce, J.M., 2015. Open-source 3-D printing technologies for education: Bringing additive manufacturing to the classroom. In Journal of Visual Languages and Computing. 28, 226–237. doi:10.1016/j.jvlc.2015.01.004
70. Sells, E., Smith, Z., Bailard, S., Bowyer, A., Olliver, V., 2010. RepRap: The Replicating Rapid Proto-typer: Maximizing Customizability by Breeding the Means of Production. In Piller FT, Tseng MM (eds) Handbook of Research in Mass Customization and Personalization: Strategies and concepts. In World Scientific. 1,568-580.
71. Simon, J., 2017. Monitoring Your 3D Prints | 3D Universe [WWW Document]. URL https://3duniverse.org/2014/01/06/monitoring-your-3d-prints/ (accessed 3.18.17).
72. STL (file format), 2017. [WWW Document]. URL https://en.wikipedia.org/wiki/STL_%28file_format%29 (accessed 4.5.17).
73. Straub, J., 2015. Initial work on the characterization of additive manufacturing (3D printing) using software image analysis. In Machines. 3, 55–71.
74. Tech, R.P.G., Ferdinand, J.-P., Dopfer, M., 2016. Open Source Hardware Startups and Their Commu-nities. In Springer International Publishing. 129–145. doi:10.1007/978-3-319-31686-4_7
75. Thing-O-Fun, 2012. Exploded Planetary Gear Set by Thing-O-Fun - Thingiverse [WWW Document]. URL http://www.thingiverse.com/thing:18291 (accessed 12.3.16).
76. Troxler, P., van Woensel, C., 2016. How Will Society Adopt 3D Printing? T.M.C. Asser Press. 183–212. doi:10.1007/978-94-6265-096-1_11
77. Ultimaker, 2017. Cura 3D Printing Slicing Software [WWW Document]. URL https://ultimaker.com/en/products/cura-software (accessed 3.12.17).
78. Vera, J., 2010. Promoting Tools that integrate LCA into the Product Design Process: a Case Study in Ontario.
79. Volpato, N., Aguiomar Foggiatto, J., Coradini Schwarz, D., 2014. The influence of support base on FDM accuracy in Z. In Rapid Prototyping Journal. 20, 182–191. doi:10.1108/RPJ-12-2012-0116
80. Wang, W.H., Hong, G.S., Wong, Y.S., Zhu, K.P., 2007. Sensor fusion for online tool condition moni-toring in milling. In International Journal Production Research. 45, 5095–5116. doi:10.1080/00207540500536913
81. Wijnen, B., Anzalone, G.C., Haselhuhn, A.S., Sanders, P.G., Pearce, J.M., 2016. Free and open-source control software for 3-D motion and processing. In Journal of Open Research Software, 4(1).
82. Wittbrodt, B.T., Glover, A.G., Laureto, J., Anzalone, G.C., Oppliger, D., Irwin, J.L., Pearce, J.M., 2013. Life-cycle economic analysis of distributed

manufacturing with open-source 3-D printers, In Mechatronics.23(6), 713-726.doi:10.1016/j.mechatronics.2013.06.002
83. Wohlers, T., 2016. Wohlers Report 2016. Wohlers Associates, Inc.
84. Wu, H., Wang, Y., Yu, Z., 2016. In situ monitoring of FDM machine condition via acoustic emission. In The International Journal of Advanced Manufacturing Technology. 84, 1483–1495. doi:10.1007/s00170-015-7809-4
85. zur Jacobsmuhlen, J., Kleszczynski, S., Witt, G., Merhof, D., 2014. Robustness analysis of imaging sys-tem for inspection of laser beam melting systems, In Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA). IEEE. 1–4. doi:10.1109/ETFA.2014.7005262

Chapter 6: Conclusions and Future Work

6.1 Conclusions

In this dissertation, it was shown that the triangulation-based geometric 3-D reconstruction algorithm is able to reliably reconstruct the 3-D objects and is able to detect errors on the low cost of an open source RepRap style 3-D printer. The triangulation-based geometric 3-D reconstruction algorithm written in MATLAB was used successfully to reconstruct 3-D objects by using two science cameras in chapter 2. In chapter 3 the shape algorithm was added to be more efficient for detecting failed printing in a 3-D printer. Both the shape algorithm and the triangulation-based geometric 3-D reconstruction algorithm written in MATLAB was tested for detecting an error of failed printing on the RepRap 3-D printer from one perspective for both single and two cameras setup by using the science cameras. In chapter 4, webcams were used instead of the science cameras to reduce the cost of this approach. To increase the ability to detect an error around 3-D printed part, six webcams were used by setup each pair of cameras in three different perspectives. Python was used to developed these algorithms instead of MATLAB to make the algorithms are more available for everyone can access with no cost for the low cost of an open source RepRap style 3-D printer, and reducing the computation time to be more effective for 3-D printer that each layer takes a few second depends on the size and how complicated of the design to finish printing. The single camera error detection in Python is tested in normal and failure state. In chapter 5, the double cameras error detection in Python is tested with two different techniques in image pre-processing, and with two different methods in the error detection algorithms.

6.2 Suggestions for Future work

There are some works that can be done to improve the algorithms represented in this dissertation. Background removing is still challenging topic because the different light setting can severely affect the process. The improved method need to be implemented for a better background removal algorithm. The computation time was improved in this dissertation by ported the code from MATLAB to Python that is faster 3X in the single camera setup but in the double camera setup is faster only up to 2X. However, the computation time would be improved if only the different area between the previous and the current printed part is calculated. There are many techniques in the block matching method. In this dissertation used only Sum of Absolute Differences block matching technique. Other techniques should be tested such as Squared difference, Normalized, squared difference, Cross correlation, Normalized cross correlation, Cosine coefficient, or Normalized cosine coefficient. The RepRap printer can print the object in height of 200 mm. But the field of view of webcam used in this work can only cover the printed part of 70 mm in width and 60 mm in height. The hardware need to be installed for moving the webcams location in height based on the height of the 3-D printed object, so that it can continue detect the error as the printer higher than 60 mm. Last, some STL model cause problem during the slicing process by using Slic3r because a facet cannot be removed. The other technique should be applied to slice the STL model.

Appendix A: Supplementary Information for Chapter 2

In chapter 2, we describe our approach to applying the SIFT to rescale and rectify the images. Figure A.1 shows the left and the right images after the background has been removed before rescaling the image.

Figure A.1 Before rescaling the image

After using the SIFT, the key point descriptors are calculated for each key point, and the distance between the closest descriptor pairs are calculated in order to find the matching points between the left and the right image that represent the same point of the object in the image. An example of one matching point between the left and the right image is shown in Figure A.2.

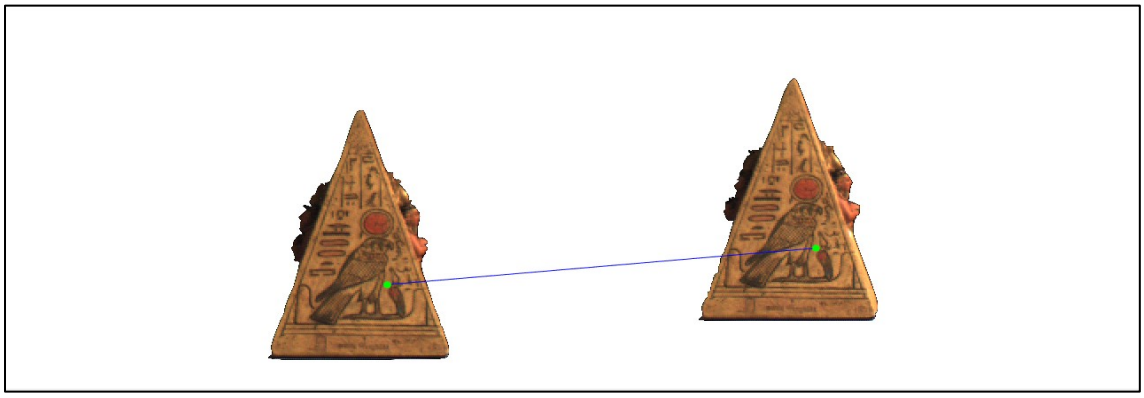


Figure A.2 An example of one matching point between the left and the right image

The height between a pair of matching points in the left image is the difference in y-coordinate between the two matching points in the left image. In addition, the height between a pair of matching points in the right image is the difference in y-coordinate between the two matching points in the right image. They are shown in Figure A.3.

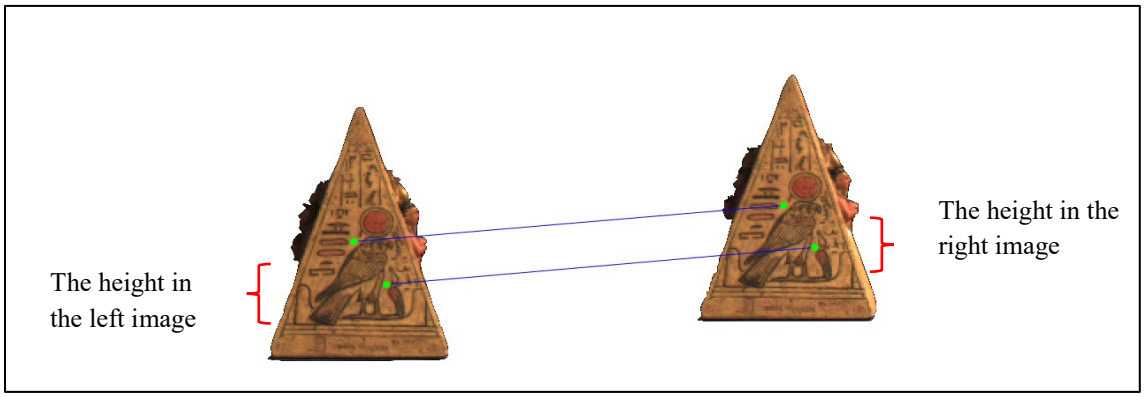


Figure A.3 The difference between a pair of matching points

Next, the summation of the heights in the left image ($\sum H_l$) is the summation of the differences between the heights of matching points from the top 10% of the best matching points in the left image. In addition, the summation of the heights in the right image ($\sum H_r$) is the summation of the differences between the heights of matching points from the top 10% of the best matching points in the right image. They are calculated by

$$\sum H_l = \sum_{i=1}^{n} (y_{left,i} - y_{left,i+1}) \quad \text{(A-1)}$$

$$\sum H_r = \sum_{i=1}^{n} (y_{right,i} - y_{right,i+1}) \quad \text{(A-2)}$$

where $\sum H_l$ is the summation of the heights from the top 10% of the best matching points in the left image, $\sum H_r$ is the summation of the heights from the top 10% of the best matching points in the right image, n is the number of the top 10% of the best matching

points where i is increased by 2 for each iteration, y_{left} is the y-coordinate in the left image, and y_{right} is the y-coordinate in the right image.

In order to rescale the image, there are two possible calculations of the height ratio as shown in Equation (A-3). The first one happens when the summation of the heights in the left image is greater than the summation of the heights in the right image. In this way the height ratio is calculated by Equation (A-3)-(a). Then the left image is rescaled by this height ratio. On the other hand, if the summation of the heights in the right image is greater than the summation of the heights in the left image, the height ratio is calculated by the second possible calculation as shown in Equation (A-3)-(b). Then the right image is rescaled by the height ratio.

$$Height\ Ratio = \begin{cases} \frac{\sum H_l}{\sum H_r} & if\ \sum H_l > \sum H_r \quad (a) \\ \frac{\sum H_r}{\sum H_l} & if\ \sum H_r > \sum H_l \quad (b) \end{cases} \tag{A-3}$$

where $\sum H_l$ is the summation of the heights from the top 10% of the best matching points in the left image, and $\sum H_r$ is the summation of the heights from the top 10% of the best matching points in the right image.

Figure A.4 shows the best 10% of the matching points. The summations of the height of the left and the right images are 32.96 and 32.27 pixels, respectively. Since the summation of the height of the left image is greater than the summation of the height of the right image, the left image is rescaled by the height ratio with 0.98. The result is shown in Figure A.5.

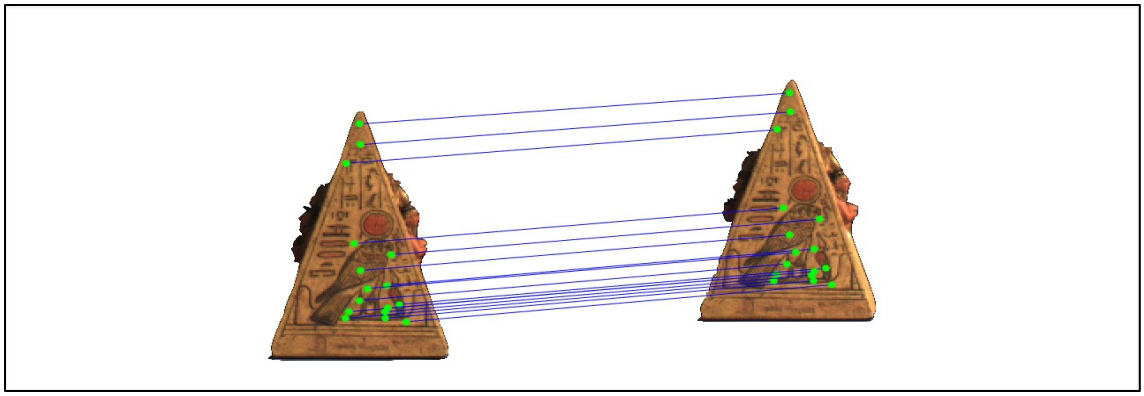


Figure A.4 Matching points for rescale after using SIFT

Figure A.5 After rescaling the image

After the rescaling step, one of the images needs to be rectified. The difference in y-coordinate between the left and the right images for each matching point ($Diff_y$), as shown in Figure A.6, is calculated by Equation (A-4). If $Diff_y$ is greater or equal to zero, the left image is moved up by $Diff_y$ pixels. If not, the left image is moved down by $Diff_y$ pixels.

$$Diff_y = y_{left} - y_{right} \tag{A-4}$$

where $Diff_y$ is the difference in y-coordinate between the left and the right images, y_{left} is the y-coordinate in the left image, and y_{right} is the y-coordinate in the right image.

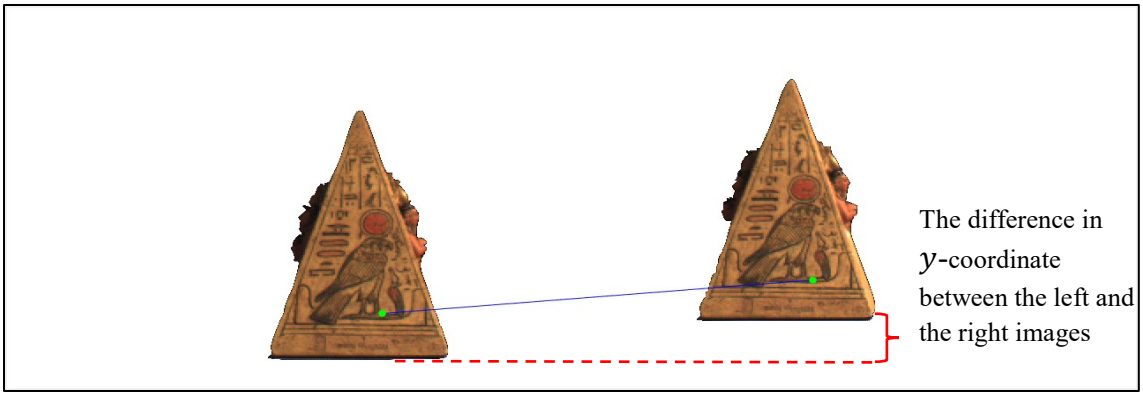


Figure A.6 The difference in y-coordinates between the left and the right images

After re-running the SIFT with the images from Figure A.5, the best 10% of the matching points are shown in Figure A.7. Therefore, the difference in the y-coordinates between the left and the right images is equal to 42 pixels. The result after rectifying the image is presented in Figure A.8.

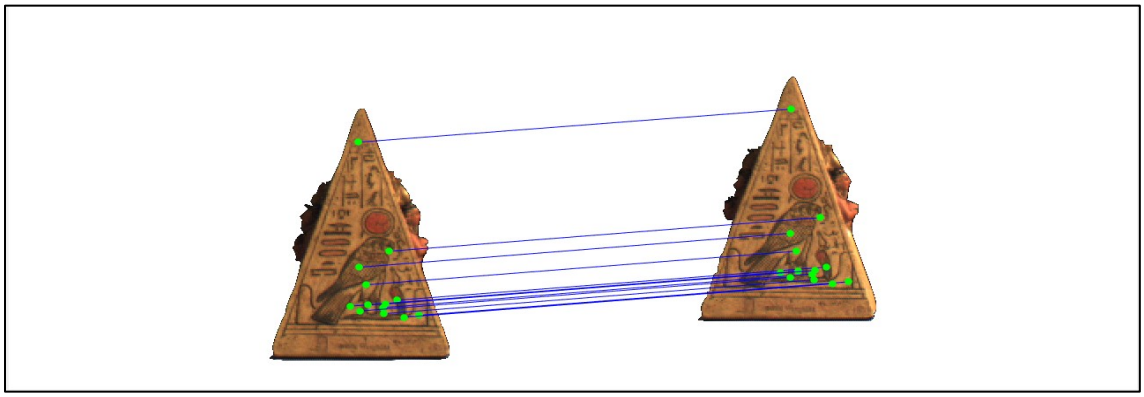


Figure A.7 Matching points for rectification after re-running the SIFT

Figure A.8 After rectifying the image

Appendix B: Supplementary Information for Chapter 4

In chapter 4, The single camera set up experiment was tested with four different models for both normal and failure state. All data tables are shown in table B.1-B.8

B.1 Normal state

Table B.1 Single camera error detection data for sun gear: Normal state

Current layer	1st right camera	2nd right camera	3rd right camera	Computation time (sec.)
30	5.20	4.77	3.96	6.29
60	5.25	4.52	3.81	6.64
90	3.94	3.62	2.67	6.63
120	3.89	3.11	2.56	6.66
150	3.48	2.96	2.18	6.71
180	3.44	2.93	2.80	6.68
210	3.45	2.98	2.92	6.72
240	3.05	3.01	2.81	6.75
268	3.06	2.66	2.97	6.73

Table B.2 Single camera error detection data for Prizm: Normal state

Current layer	1st right camera	2nd right camera	3rd right camera	Computation time (sec.)
30	3.78	6.67	5.80	8.04
60	2.48	3.02	4.31	6.38
90	1.82	2.00	2.93	7.17
120	2.39	2.75	3.09	6.47
150	2.23	2.44	3.03	6.57

Table B.3 Single camera error detection data for gear: Normal state

Current layer	1st right camera	2nd right camera	3rd right camera	Computation time (sec.)
30	4.06	3.38	2.86	10.05
60	2.53	4.70	2.31	6.88
90	2.13	1.46	2.84	7.10
120	5.25	2.12	3.76	6.70
129	1.08	1.30	0.70	6.87

Table B.4 Single camera error detection data for t55gear: Normal state

Current layer	1st right camera	2nd right camera	3rd right camera	Computation time (sec.)
30	4.17	4.81	4.10	6.39
60	2.72	4.52	4.32	6.43
90	3.27	3.94	4.14	6.54
120	2.36	3.18	3.63	6.48
150	1.81	2.52	3.07	6.63

B.2 Failure state

Table B.5 Single camera error detection data for Sun gear: Failure state

Current layer	1st right camera	2nd right camera	3rd right camera	Computation time (sec.)
30 & 60	15.21	15.47	15.07	6.54
60 & 90	13.84	15.95	15.36	8.26
90 & 120	15.78	15.82	15.94	7.06
120 & 150	15.55	15.28	16.23	8.14
150 & 180	13.56	13.12	14.81	6.70
180 & 210	12.81	12.77	14.25	6.79
210 & 240	11.54	11.88	13.65	6.89
240 & 268	10.21	10.29	12.22	7.08

Table B.6 Single camera error detection data for Prizm: Failure state

Current layer	1st right camera	2nd right camera	3rd right camera	Computation time (sec.)
30 & 60	17.02	17.67	19.01	6.77
60 & 90	12.56	11.54	12.98	6.92
90 & 120	9.30	9.11	10.38	7.94
120 & 150	8.62	7.63	9.06	8.08

Table B.7 Single camera error detection data for gear: Failure state

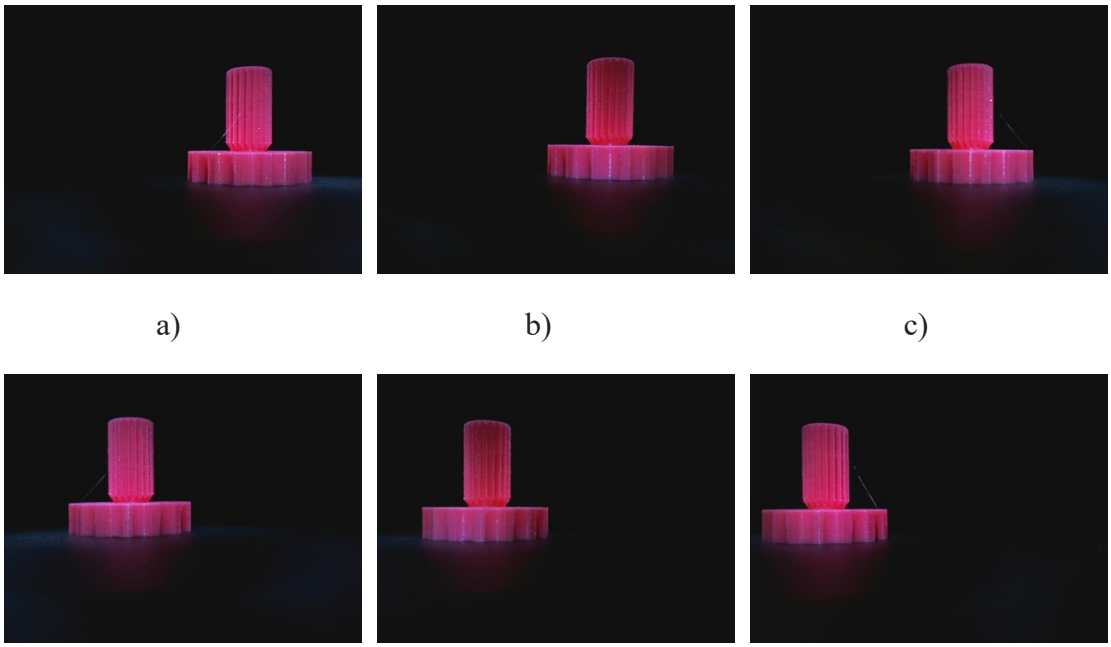
Current layer	1st right camera	2nd right camera	3rd right camera	Computation time (sec.)
30 & 60	15.40	16.17	14.74	6.91
60 & 90	11.81	14.01	12.40	6.73
90 & 129	10.83	10.58	11.59	6.65

Table B.8 Single camera error detection data for t55gear: Failure state

Between layer	1st right camera	2nd right camera	3rd right camera	Computation time (sec.)
30 & 60	15.23	17.03	15.74	6.49
60 & 90	17.73	19.73	19.69	6.51
90 & 120	16.18	16.27	16.50	6.68
120 & 150	13.43	14.28	14.62	6.63

Appendix C: Supplementary Information for Chapter 5

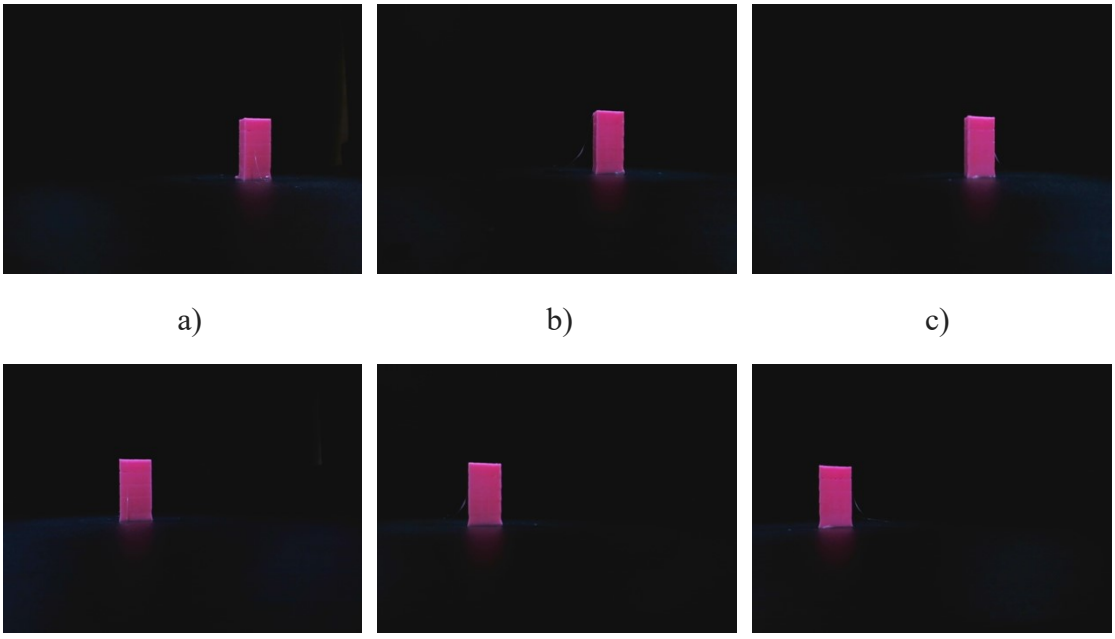
In chapter 5, All image results for each full model display from the first, the second and the third pair of cameras respectively: a-c) the images from the left camera, and d-f) the images from the right camera as shown in Figure C.1-C.4.



a) b) c)

d) e) f)

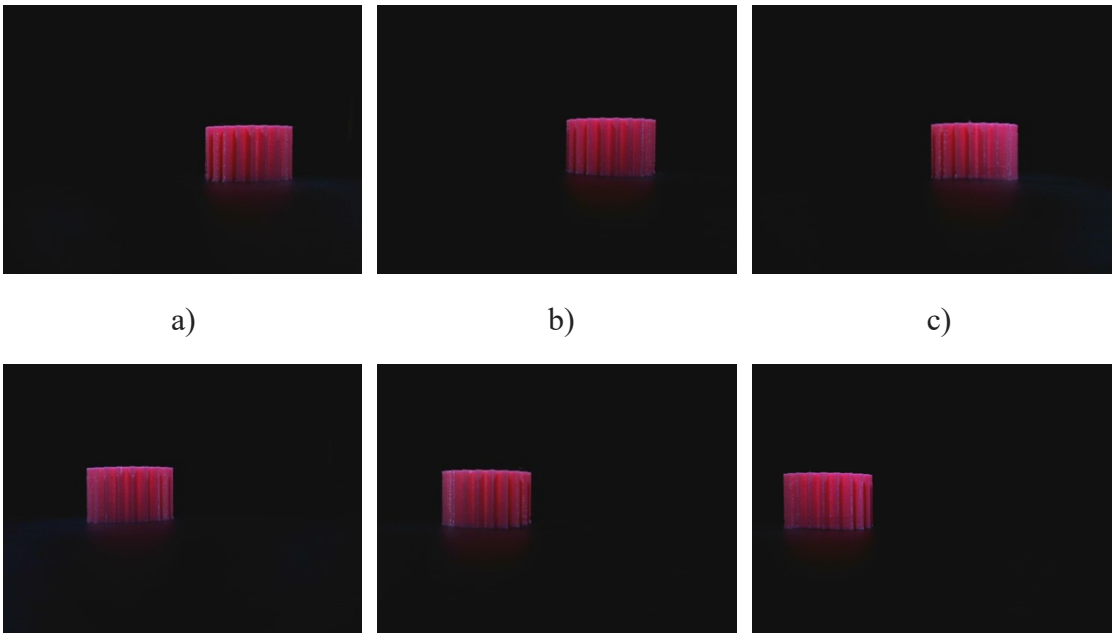
Figure C.1 Full model of sun gear image



a) b) c)

d) e) f)

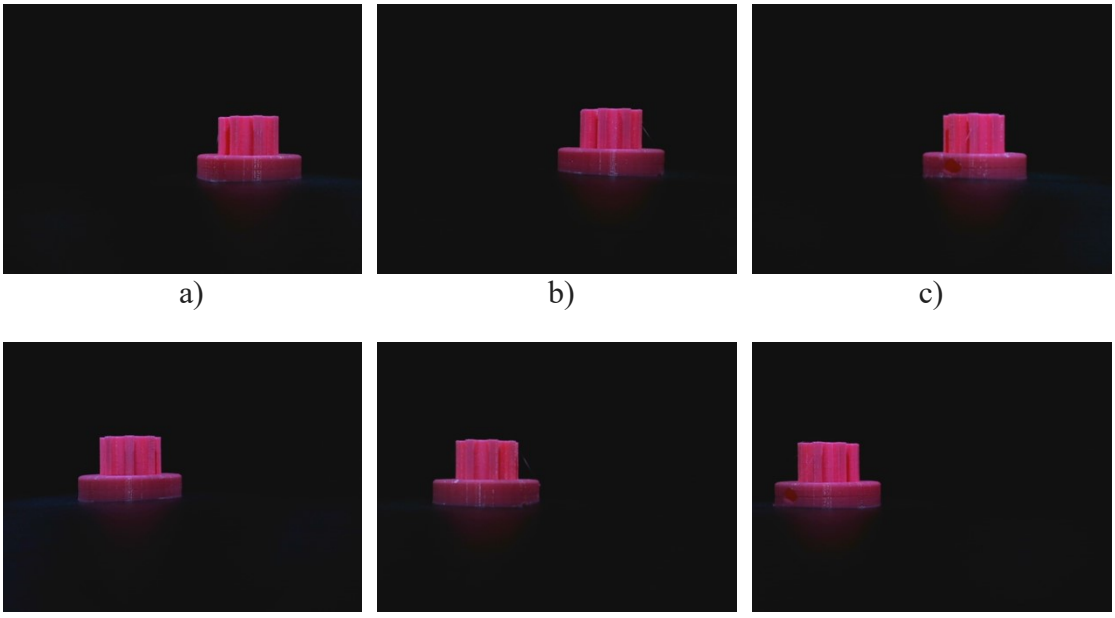
Figure C.2 Full model of prism image



a) b) c)

d) e) f)

Figure C.3 Full model of gear image

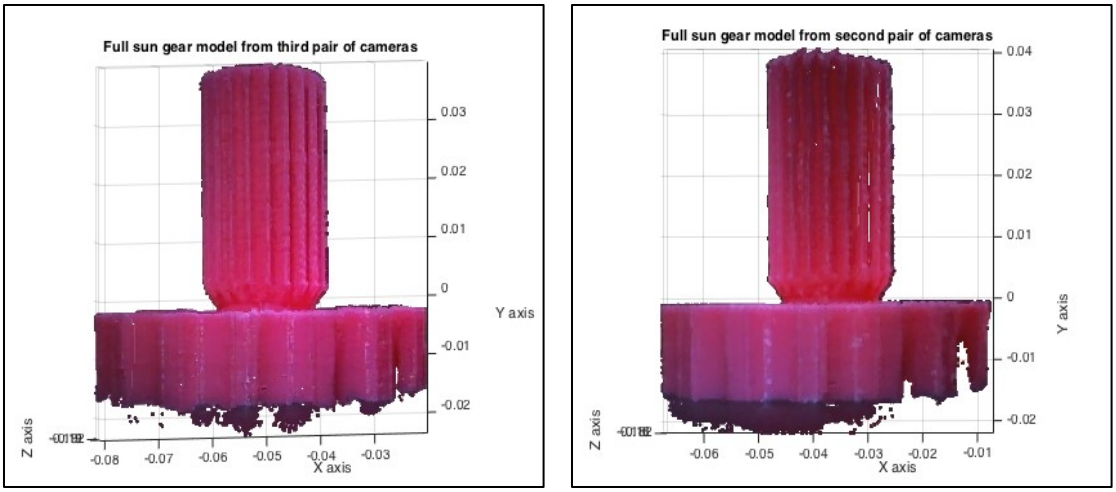


a) b) c)

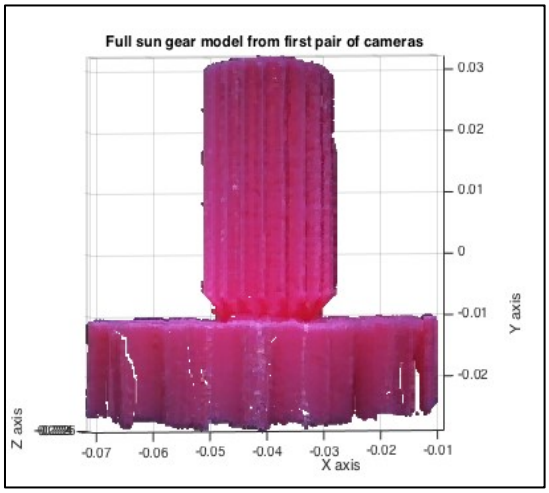
d) e) f)

Figure C.4 Full model of t55gear image

The double cameras set up were tested with two different experiments: image pre-processing and error detection. The image pre-processing was run by two different techniques: SIFT and RANSAC to rescale and rectify, and no rescale and rectification. The error detection is tested with two different methods: horizontal magnitude, and horizontal and vertical magnitude. For all experiments are tested with four model under normal printing and failure state. All data tables are shown in table C.1-C.30. From the results showed that no rescale and rectify in the image pre-processing step with horizontal and vertical magnitude algorithm was success to detect the error 100%, and the 3-D reconstruction results for full model of different four geometries in three different perspectives are shown in Figure C.5-C.8.



a) b)



c)

Figure C.5 3-D reconstruction of sun gear model: a) first pair of cameras, b)second pair of cameras, and c) third pair of cameras

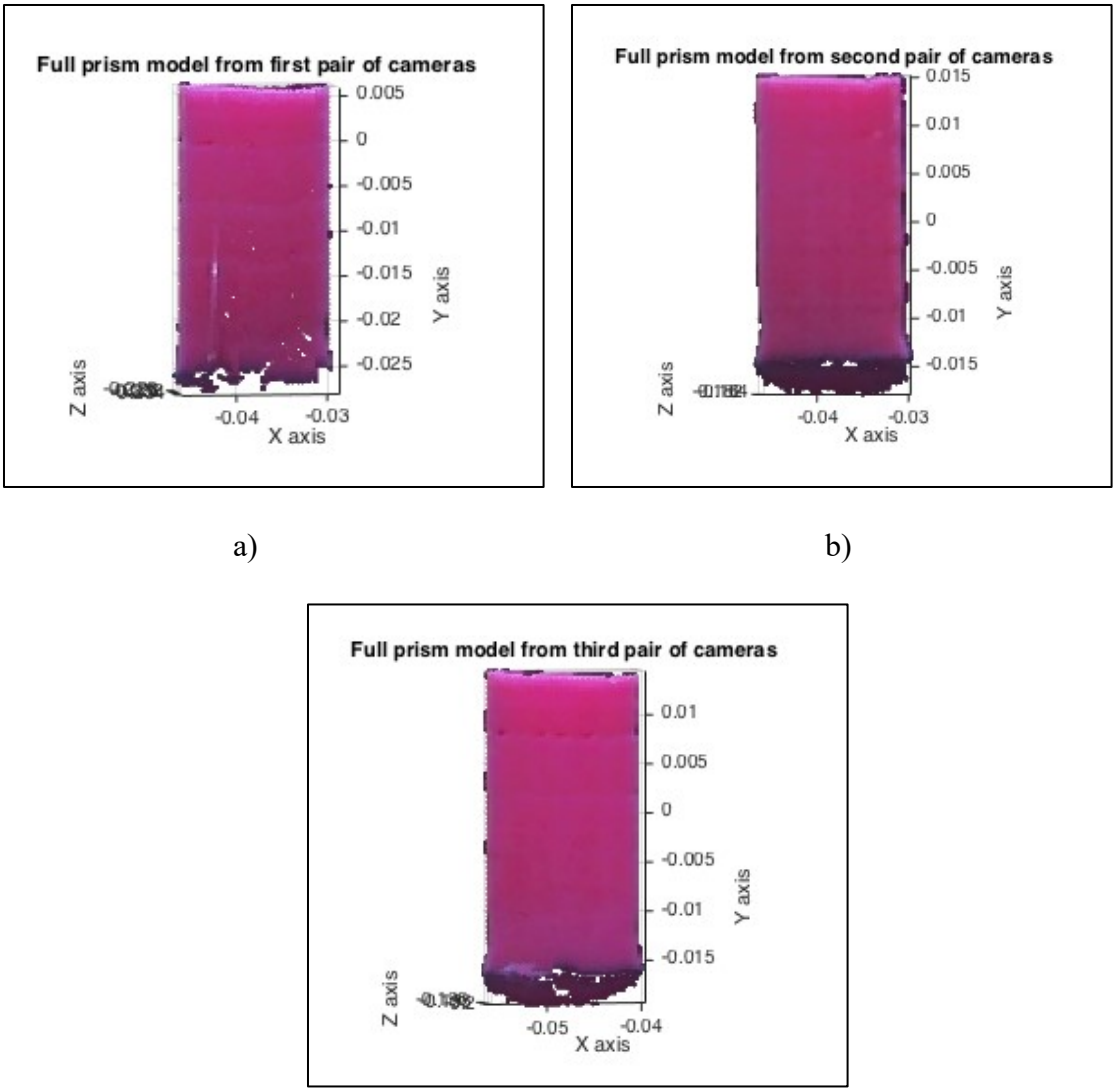
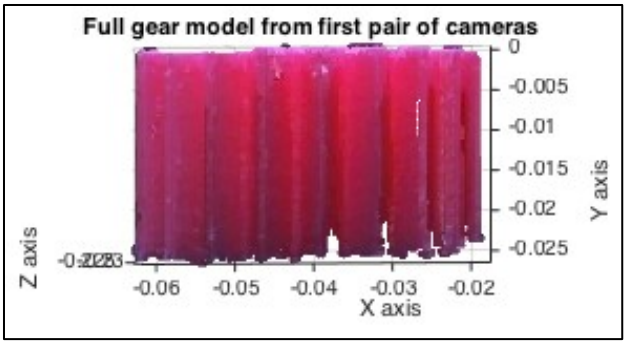
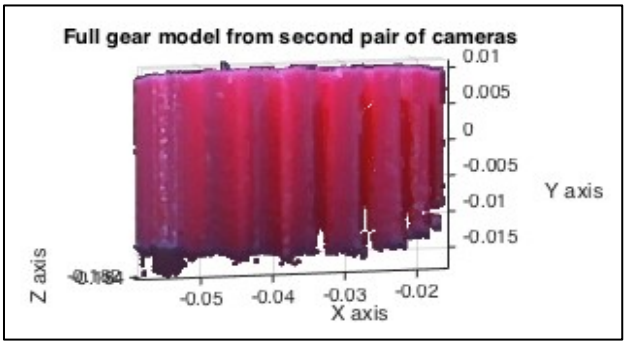


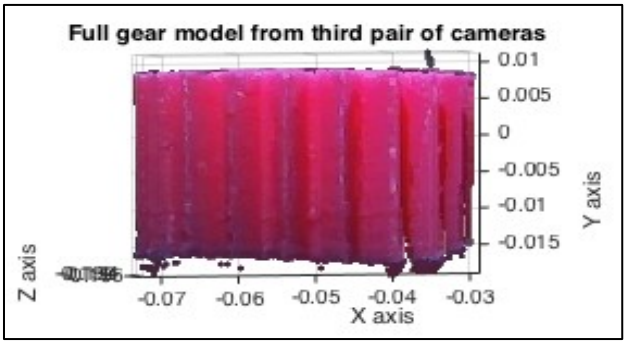
Figure C.6 3-D reconstruction of prism model: a) first pair of cameras, b)second pair of cameras, and c) third pair of cameras



a)

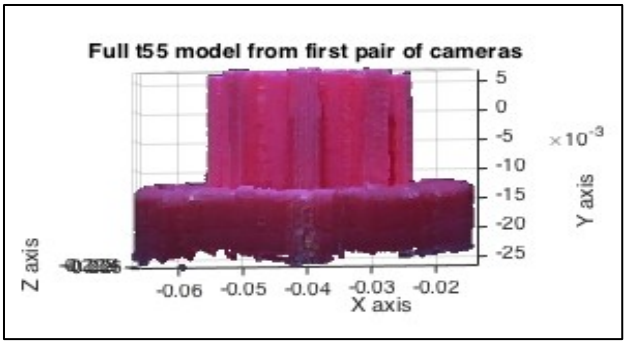


b)

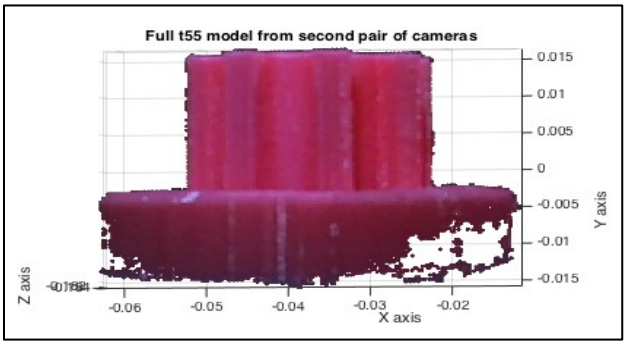


c)

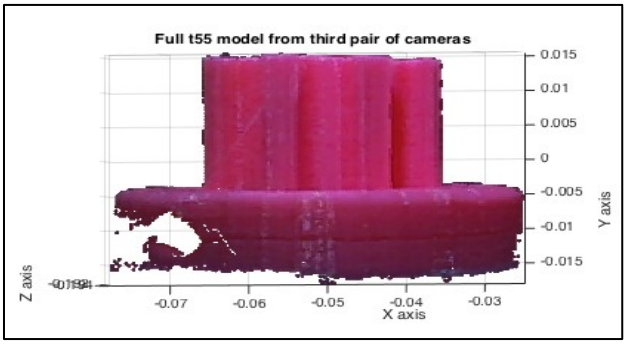
Figure C.7 3-D reconstruction of gear model: a) first pair of cameras, b)second pair of cameras, and c) third pair of cameras



a)



b)



c)

Figure C.8 3-D reconstruction of t55gear model: a) first pair of cameras, b)second pair of cameras, and c) third pair of cameras

C.1 Image Pre-Processing

C.1.1 SIFT and RANSAC to Rescale and Rectify

A) Normal Printing State

Table C.1 Double camera error detection data for sun gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	32.67	50.37	28.04	47.91	116.09	49.12
60	30.41	72.94	22.83	58.91	6.64	70.51
90	23.16	74.17	16.59	89.41	5.03	87.39
120	28.64	92.15	18.84	96.04	6.65	101.83
150	29.14	102.35	18.32	116.01	3.16	112.99
180	27.52	110.61	18.86	127.31	8.22	118.94
210	29.05	132.08	18.32	137.29	4.64	136.54
240	27.47	140.47	17.46	146.16	3.72	145.56
268	25.01	152.11	15.36	164.09	50.01	149.93

Table C.2 Double camera error detection data for prism: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	0	0	0	0	0	0
60	19.31	38.23	16.79	16.79	20.53	17.26
90	26.05	36.21	28.13	20.39	26.78	18.32
120	0	0	0	0	0	0
150	0	0	0	0	0	0

Table C.3 Double camera error detection data for gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	31.05	29.77	20.39	29.21	5.08	30.85
60	30.49	45.57	23.88	45.32	7.57	48.61
90	23.02	61.99	17.98	68.74	6.22	64.99
120	96.32	83.36	18.37	87.16	3.59	84.54
129	29.84	85.61	13.36	88.26	6.38	91.51

Table C.4 Double camera error detection data for t55gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	56.49	33.64	20.13	34.22	41.76	36.25
60	47.77	53.51	19.51	19.41	14.82	55.74
90	38.42	67.81	23.36	69.63	99.98	70.93
120	37.21	81.37	27.75	85.53	12.11	84.21
150	32.74	95.02	34.04	98.19	8.34	97.24

B) Failure State

Table C.5 Double camera error detection data for sun gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	32.67	49.41	28.04	46.91	116.09	48.58
60&90	238.39	74.34	218.74	59.54	176.72	71.78
90&120	23.16	74.24	16.59	89.78	5.03	87.54
120&150	28.64	91.93	18.84	95.46	6.65	100.03
150&180	29.14	102.42	18.32	117.91	3.16	114.49
180&210	27.52	111.67	18.86	127.52	8.22	118.41
210&240	29.05	131.83	18.32	138.52	4.64	136.17
240&268	27.47	149.02	17.46	146.92	3.72	145.63

Table C.6 Double camera error detection data for prism: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	0	0	0	0	0	0
60&90	38.23	19.63	16.79	16.64	17.26	20.52
90&120	36.21	25.97	20.39	27.85	18.32	26.55
120&150	0	0	0	0	0	0

Table C.7 Double camera error detection data for gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	31.05	30.51	20.39	29.52	24.98	31.76
60&90	30.49	45.07	23.88	45.95	15.87	47.65
90&129	23.02	23.02	17.98	68.06	15.91	65.41

Table C.8 Double camera error detection data for t55gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	72.87	33.72	32.71	34.45	56.59	35.67
60&90	134.35	53.66	27.65	19.78	82.09	55.79
90&120	38.42	68.05	23.36	69.52	99.98	71.97
120&150	31.21	81.55	27.75	83.72	12.11	83.96

C.1.2 Non-Rescale and Rectification

A) Normal Printing State

Table C.9 Double camera error detection data for sun gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	5.36	16.58	1.29	22.51	3.49	22.26
60	5.33	24.21	1.29	34.15	2.98	32.52
90	3.79	30.63	0.46	44.84	2.08	41.87
120	2.88	35.5	0.78	54.01	2.52	46.89
150	0.21	38.8	0.05	59.97	2.52	53.54
180	3.79	43.72	0.78	64.17	2.52	58.86
210	3.79	48.29	0.78	72.66	1.2	64.39
240	1.34	52.41	0.78	81.1	2.52	71.61
268	1.97	57.73	0.87	83.59	1.2	77.29

Table C.10 Double camera error detection data for prism: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	6.28	5.47	0.44	7.36	4.14	6.94
60	8.56	7.62	1.04	10.4	4.97	9.34
90	6.94	9.94	0.84	13.37	4.12	11.98
120	9.26	11.93	0.41	17.55	2.94	15.05
150	6.94	14.73	1.32	19.07	1.02	20.31

Table C.11 Double camera error detection data for gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	3.95	10.94	0.42	14.03	2.71	14.61
60	3.81	16.78	0.26	20.48	3.02	21.04
90	3.21	20.94	0.26	30.93	2.31	29.79
120	3.21	27.79	0.26	41.14	2.01	37.48
129	3.21	28.27	0.26	40.58	2.01	36.91

Table C.12 Double camera error detection data for t55gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	4.48	13.04	0.51	16.45	3.59	16.41
60	4.48	19.31	1.65	25.24	3.61	24.81
90	8.26	23.55	0.85	31.84	5.41	30.52
120	8.69	28.89	3.85	38.54	5.41	37.46
150	7.83	32.98	4.18	43.19	6.32	41.08

B) Failure State

Table C.13 Double camera error detection data for sun gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	14.85	16.13	9.09	22.14	1.96	21.87
60&90	29.87	25.39	23.28	34.86	19.16	34.35
90&120	28.83	30.61	13.56	44.19	12.43	41.03
120&150	18.26	34.65	7.33	53.33	10.73	46.94
150&180	12.66	39.03	6.41	58.87	6.32	53.29
180&210	10.98	43.69	10.05	63.21	4.89	58.68
210&240	7.44	49.88	2.53	71.4	2.61	64.95
240&268	4.66	52.32	1.24	79.14	0.63	71.24

Table C.14 Double camera error detection data for prism: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	30.62	5.53	30.84	7.22	27.01	6.59
60&90	21.85	7.63	20.01	10.47	17.77	9.32
90&120	19.81	9.61	15.72	13.42	13.45	12.15
120&150	13.53	11.96	10.74	16.95	10.17	15.01

Table C.15 Double camera error detection data for gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	27.92	10.98	28.82	15.18	24.98	14.31
60&90	15.63	16.6	25.77	20.37	15.87	21.41
90&129	22.92	21.26	16.96	30.47	15.91	28.56

Table C.16 Double camera error detection data for t55gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	52.73	13.31	57.08	16.43	51.85	16.54
60&90	37.46	19.16	42.87	25.15	38.49	24.17
90&120	31.7	23.5	31.29	32.33	27.96	30.86
120&150	22.42	29.29	24.45	41.17	17.95	40.91

C.2 Error Detection

C.2.1 Horizontal Magnitude

A) Normal Printing State

Table C.17 Double camera error detection data for sun gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	5.36	16.58	1.29	22.51	3.49	22.26
60	5.33	24.21	1.29	34.15	2.98	32.52
90	3.79	30.63	0.46	44.84	2.08	41.87
120	2.88	35.5	0.78	54.01	2.52	46.89
150	0.21	38.8	0.05	59.97	2.52	53.54
180	3.79	43.72	0.78	64.17	2.52	58.86
210	3.79	48.29	0.78	72.66	1.2	64.39
240	1.34	52.41	0.78	81.1	2.52	71.61
268	1.97	57.73	0.87	83.59	1.2	77.29

Table C.18 Double camera error detection data for prism: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	6.28	5.47	0.44	7.36	4.14	6.94
60	8.56	7.62	1.04	10.4	4.97	9.34
90	6.94	9.94	0.84	13.37	4.12	11.98
120	9.26	11.93	0.41	17.55	2.94	15.05
150	6.94	14.73	1.32	19.07	1.02	20.31

Table C.19 Double camera error detection data for gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	3.95	10.94	0.42	14.03	2.71	14.61
60	3.81	16.78	0.26	20.48	3.02	21.04
90	3.21	20.94	0.26	30.93	2.31	29.79
120	3.21	27.79	0.26	41.14	2.01	37.48
129	3.21	28.27	0.26	40.58	2.01	36.91

Table C.20 Double camera error detection data for t55gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	4.48	13.04	0.51	16.45	3.59	16.41
60	4.48	19.31	1.65	25.24	3.61	24.81
90	8.26	23.55	0.85	31.84	5.41	30.52
120	8.69	28.89	3.85	38.54	5.41	37.46
150	7.83	32.98	4.18	43.19	6.32	41.08

B) Failure State

Table C.21 Double camera error detection data for sun gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	14.85	16.13	9.09	22.14	1.96	21.87
60&90	29.87	25.39	23.28	34.86	19.16	34.35
90&120	28.83	30.61	13.56	44.19	12.43	41.03
120&150	18.26	34.65	7.33	53.33	10.73	46.94
150&180	12.66	39.03	6.41	58.87	6.32	53.29
180&210	10.98	43.69	10.05	63.21	4.89	58.68
210&240	7.44	49.88	2.53	71.4	2.61	64.95
240&268	4.66	52.32	1.24	79.14	0.63	71.24

Table C.22 Double camera error detection data for prism: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	30.62	5.53	30.84	7.22	27.01	6.59
60&90	21.85	7.63	20.01	10.47	17.77	9.32
90&120	19.81	9.61	15.72	13.42	13.45	12.15
120&150	13.53	11.96	10.74	16.95	10.17	15.01

Table C.23 Double camera error detection data for gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	27.92	10.98	28.82	15.18	24.98	14.31
60&90	15.63	16.6	25.77	20.37	15.87	21.41
90&129	22.92	21.26	16.96	30.47	15.91	28.56

Table C.24 Double camera error detection data for t55gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	52.73	13.31	57.08	16.43	51.85	16.54
60&90	37.46	19.16	42.87	25.15	38.49	24.17
90&120	31.7	23.5	31.29	32.33	27.96	30.86
120&150	22.42	29.29	24.45	41.17	17.95	40.91

C.2.2 Horizontal and Vertical Magnitude

A) Normal Printing State

Table C.25 Double camera error detection data for sun gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	5.08	15.99	1.27	22.67	3.37	21.92
60	5.06	23.87	1.27	33.97	2.92	33.09
90	3.79	30.54	0.46	43.64	2.08	41.08
120	2.81	35.52	1.98	50.84	2.45	45.82
150	1.95	39.16	0.54	57.82	2.45	53.43
180	3.65	43.45	0.77	63.33	2.45	58.21
210	3.65	48.42	0.77	70.59	1.91	63.76
240	2.79	53.05	0.77	76.97	2.45	69.32
268	4.52	58.39	2.17	82.79	3.78	75.49

Table C.26 Double camera error detection data for prism: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	3.85	5.61	0.44	7.18	3.97	6.71
60	7.88	7.71	1.02	10.34	4.47	9.46
90	3.84	9.46	0.85	13.64	3.96	12.23
120	7.02	11.93	0.85	17.37	2.85	15.16
150	9.01	14.44	0.85	19.82	1.31	17.88

Table C.27 Double camera error detection data for gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	3.81	11.15	0.42	14.32	2.65	14.28
60	3.67	17.39	0.26	21.43	2.93	22.42
90	3.11	21.04	0.26	31.35	2.26	29.24
120	3.11	27.76	0.26	40.17	2.26	37.17
129	2.34	27.65	0.44	40.39	1.97	37.09

Table C.28 Double camera error detection data for t55gear: Normal Printing State

Current layer	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30	12.61	13.04	0.51	16.42	3.47	16.14
60	19.31	19.31	1.65	25.24	3.61	4.8
90	23.86	23.55	3.71	32.93	5.13	30.48
120	28.92	28.89	3.71	39.47	5.13	37.28
150	35.52	32.98	3.11	47.11	5.13	44.21

B) Failure State

Table C.29 Double camera error detection data for sun gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	100	12.18	100	22.17	100	21.51
60&90	100	25.42	100	35.72	100	34.55
90&120	100	30.94	100	44.64	100	41.45
120&150	100	34.76	100	50.86	100	45.67
150&180	100	39.34	100	57.54	100	54.02
180&210	100	44.06	100	63.65	100	57.43
210&240	100	49.05	100	70.03	100	64.14
240&268	100	52.32	100	79.14	100	71.24

Table C.30 Double camera error detection data for prism: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	100	5.59	100	7.22	100	6.83
60&90	100	7.61	100	10.52	100	9.45
90&120	100	9.56	100	13.62	100	20.59
120&150	100	12.09	100	16.85	100	14.97

Table C.31 Double camera error detection data for gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	100	11.19	100	14.14	100	14.07
60&90	100	17.27	100	21.57	100	21.16
90&129	100	21.34	100	31.24	100	28.64

Table C.32 Double camera error detection data for t55gear: Failure State

Between layers	1st pair of camera		2nd pair of camera		3rd pair of camera	
	Error (%)	Time (sec.)	Error (%)	Time (sec.)	Error (%)	Time (sec.)
30&60	100	19.23	100	16.52	100	18.42
60&90	100	19.48	100	26.12	100	24.27
90&120	100	23.61	100	33.25	100	30.22
120&150	100	28.87	100	39.55	100	37.04