University of Illinois at Urbana-Champiagn

ME 470: Senior Design

Design of a Vehicle Hail Damage Detection System

Final Design Report

Friday, May 3rd, 2019

Authors:

Kevin Cruse	kcruse2@illinois.edu	1 (630) 903-5544
Vedant Jain	vjain13@illinois.edu	1 (217) 550-2974
Michael Lee	mlee141@illinois.edu	1 (847 347-1874
Wentao Ni	wni5@illinois.edu	1 (217) 904-9616
Dingyu Peng	dpeng4@illinois.edu	1 (217) 650-9045

Faculty Advisor:
Prof. Sascha Hilgenfeldt

TA:
Christopher Marry

Submitted To:
Mr. Randy Harless
Ally Financial
900 N. Arlington Heights Rd
Itaska, IL 60143

Contents

1 Executive Summary **1**

2 Introduction **2**
2.1 Problem Statement 2
2.2 Project Constraints 2
2.3 Literature Review 2
2.3.1 Distortion Imaging 2
2.3.2 Acoustic Wave Imaging 4
2.3.3 LIDAR 5
2.3.4 3D Scanning 6
2.4 Concept Selection 6

3 System Design **8**

4 Hardware **9**
4.1 Test Stand 9
4.2 Image Acquisition Hardware 11

5 Image Preprocessing **13**

6 Dent Detection **14**
6.1 Clustering 15
6.2 Conversion 16
6.3 Graphical User Interface 17

7 Budget **18**

8 Conclusions and Recommendations **18**

A Acknowledgement **20**

B Standards **20**

C Dent Detection Python Code **20**

D User Manual **29**
D.1 Vehicle-hail-dent-detection 29
D.2 Image Acquisition 29
D.3 Installing 29
D.4 Running 30
D.5 File Transfer 30

D.6 Dent Detection 30
D.7 Installing Software 30
D.8 Running 31
D.9 Finishing 31

References **32**

List of Figures

1 Schematic of column matrix projection used to scan sheet metal [1] 3
2 Schematic of dot matrix projection [2] 4
3 General Surface topography of book arrangement recreated using ultrasonic senors [3] 4
4 Damage Detection in pipe using EMAT [4] 5
5 Diagram of TOF LIDAR 5
6 System Schematic 9
7 Engineering drawing for test stand 9
8 Schematic for imaging apparatus 10
9 Schematic for imaging apparatus 11
10 Image of completed test stand 11
11 Raspberry Pi and camera 12
12 Raspberry Pi GUI 12
13 Raw images and stitched image 13
14 Binarized, extracted, subtracted, and denoised images 14
15 Clustered image 15
16 Labeled image 16
17 Processing GUI 17

List of Tables

1 Decision Matrix 7
2 Mapping from pixel size to coin size for dent detection 17
3 Budget 18

1 Executive Summary

Hail damage is responsible for hundreds of millions of dollars in vehicle damage annually. This directly affects Ally Financial, which is responsible for insuring car dealership lots. Ally currently performns vehicle inspection manually but has looked into automating the process for the sake of efficiency. Our team was tasked with researching, designing, and building a prototype capable of taking images of dented car panels and an image processing algorithm that can detect, count, and sort dents. Regarding research, our team looked into various methods such as ultrasonic imaging, 3D scanning, distortion imaging, and LIDAR. We also determiend advantages and disadvantages of a previous prototype developed by a 2018 Senior Design team. By drawing from our own research in the literature review and reusing the working elements of the previous team's prototype, our team was able to select a main concept. The main concept we chose was the column matrix projection method, which the previous team used. Once the main concept was established, the team was split between two groups: the hardware team and the software team. The hardware team designed and built a physical rig with a camera and light mount capable of taking pictures consistently and without the need to remove car panels. The software team reused the Raspberry Pi and camera from the previous prototype, but used Python rather than Matlab to write a shape extraction method for image processing. A GUI was also developed so that the user would be able to intuitively use the image processing algorithm. This project incurred a cost of $1,324.05, with the largest portion of cost going toward the development of a test stand. Overall, the project greatly improved upon the previous team's work through the physical rig and the shape extraction method. The previous team's work was expanded upon by allowing the inspection of all types of vehicles with different colored surfaces. The accuracy of dent detection was increased through the new image processing algorithm. However, there are still improvements that can be made by implementing an alternate rig design and an encoder to keep track of the slider's position.

2 Introduction

2.1 Problem Statement

This project was sponsored by Ally Financial, a major bank holding company. Ally provides an array of services including car financing, corporate lending, online banking and automotive insurance. A majority of their revenue comes through ensuring automotive dealership lots. Ally estimates that hail causes roughly damage worth $125 million to cars annually. Cars in dealership lots are particularly susceptible to hail damage. Assessing hail dents is currently a manual process, where assessors inspect every car, panel by panel, and count the number of dents using fluorescent lights to locate the dents. Dents are then classified based on their size in terms of coins (dime, nickel, quarter and half dollar). They then report the damage condition of the vehicle based on the number and size of dents and provide a repair cost estimate based on industry standard charts. The sponsor seeks a way to make the assessing of hail damage simpler and less time-consuming.

2.2 Project Constraints

Since Ally financial needs this device to assess cars at dealerships, it should be mobile and robust due to the fact that it will be transported from dealership to dealership. It needs to be able to accurately and effectively scan and identify dents. In terms of accuracy, the image processing algorithm should be able to correctly identify dents based on their size. The dents are to be categorized into four categories: dime, nickel, quarter, and half dollar sizes. In terms of efficiency, the process of scanning a car should take between 5 to 10 minutes. In addition, it also needs to be easy to use and require no prior training to operate. Other constraints include safety and cost effectiveness. As for the other constraints such as global, cultural, societal, and etc. our team did not find them particularly applicable to our project.

2.3 Literature Review

2.3.1 Distortion Imaging

Distortion imaging is a method to detect surface abnormalities. A predetermined light pattern is created on a surface. Surface abnormalities cause the pattern to distort. An image of the distortions is captured, and is then used to determine the size and the position

of the abnormalities. The two methods considered here are column matrix projection and dot matrix projection.

Column matrix projection is a form of distortion imaging where a striped pattern is projected. The straight lines distort and form curves or even circles, depending on the size of the pattern with respect to the size of the abnormality, when a dent is present. This method is currently used in the quality control of sheet metal, a set up for which is shown in Figure 1. A projector is used to create a striped pattern, and a very high definition line scan camera is used to capture the distortions.

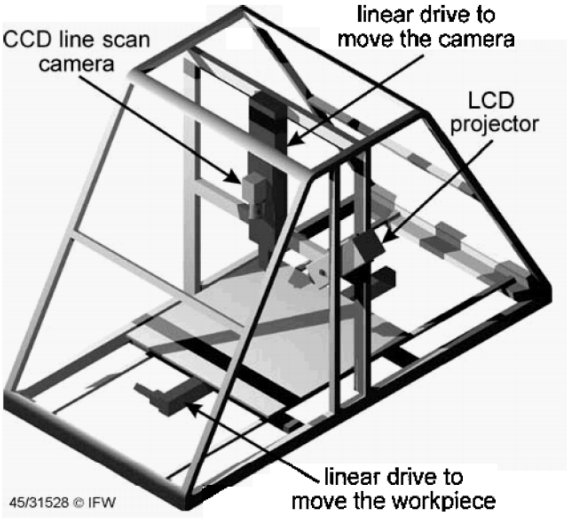

Figure 1. Schematic of column matrix projection used to scan sheet metal [1]

Dot Matrix Projection is a variation of the distortion imaging and photogrammetry technique in which a grid of dots or a random pattern are projected onto a surface and their reflection is used to determine the size and position of surface abnormalities. The flight time of the projected light is used to calculate depth. This technology is widely used for applications such as facial recognition (such as Apple's FaceID) as well as for motion detection (such as Microsoft's Kinect). A schematic for the application of dot matrix projection is shown in Figure 2.

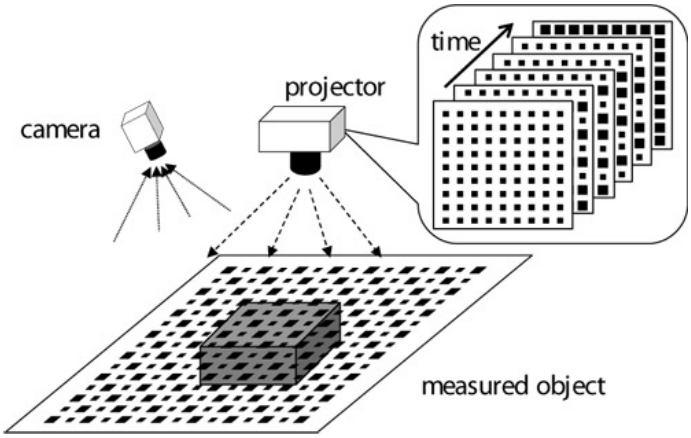

Figure 2. Schematic of dot matrix projection [2]

2.3.2 Acoustic Wave Imaging

Ultrasonic imaging utilises sound waves to detect dents. Ultrasonic imaging holds advantages over optical imaging desigsn in its ability to operate effectively outdoors. Ultrasonic sensors are also inexpensive, making it easy to form an array of sensors for point reconstruction. A comparison of the mechanism for piezoelectric and electromagnetic acoustic imaging is shown in Figure 3.

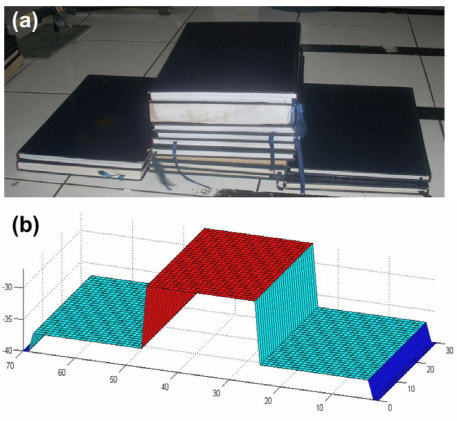

Figure 3. General Surface topography of book arrangement recreated using ultrasonic senors [3]

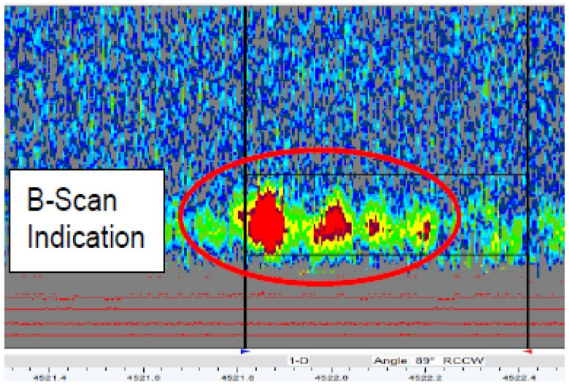

Figure 4. Damage Detection in pipe using EMAT [4]

2.3.3 LIDAR

One of the methods considered for detecting dents was the diffuse laser or LIDAR (Light Detection and Ranging). LIDAR uses laser pulses to measure a distance from a specified object. The laser pulses are sent to its target and then reflected off back to a sensor. Figure 5 shows an example of LIDAR below. Once the reference light is reflected onto the photosensor, the timer measurement circuit calculates the distance R from the object.

LIDAR is currently used for terrain model generation and is able to map vegetation, building features, topography, etc. [5] This application of LIDAR would be useful for our project. LIDAR is also used for self-driving vehicles in combination with cameras and radar. [6]

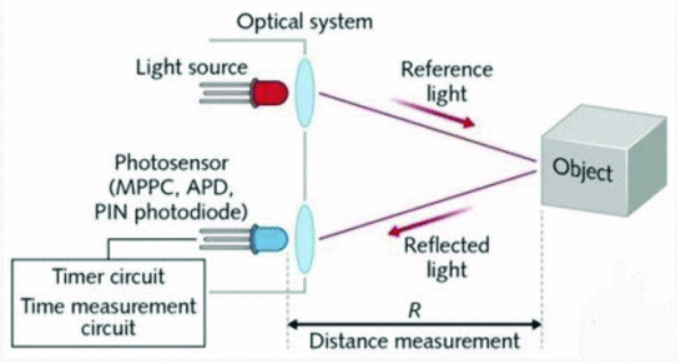

Figure 5. Diagram of TOF LIDAR

2.3.4 3D Scanning

3D scanning is a non-contact, non-destructive imaging technology that uses a laser to capture the depth image of an object. It creates a 3D point cloud with location and depth information. The point cloud can be rendered into a mesh to reconstruct the object, which is frequently used in reverse engineering, manufacturing, and art and design. With a similar concept but different scope to the LIDAR technology, 3D scanning is more widely seen in civilian industries and commercial products.

2.4 Concept Selection

After concluding the literature review, a pugh decision matrix was used to determine the optimal method for capturing dents. Experts from the Imaging Technology Group at the Beckman Institute, as well as our sponsors at ally were also consulted.

After creating the Pugh Decision Matrix shown in Table 1, 3D scanning, and distortion imaging had the highest scores. However, after consultation with Ally, 3D scanning as an idea was dropped. Ally has done their own testing with Creaform 3D scanners and found them to be inaccurate. While LIDAR has many advantages such as high resolution and low CPU consumption, it was simply too expensive and difficult to use for this project. In addition, the resolution scale of LIDAR was too small to detect dents. LIDAR can work up to a resolution scale of centimeters rather than millimeters. For our project, our dents were on the scale of millimeters and therefore the consideration of LIDAR was dropped. While conceptually promising, ultrasonic imaging also shows disadvantages. Macro-scale applications of ultrasonic imaging provide inexact information general object contour and tumor or crack location. This technique will likely not fulfill the resolution requirements for this project, entailing automatic dent size detection in addition to location.

As a result, distortion imaging was our method of choice. We experimented with various patterns, including stripes, dot matrices, and checkerboard patterns. However, due to the lack of existing research in both dot matrix projection and checkerboard projection, both of those concepts were dropped, and we decided to proceed with column matrix projection.

Table 1. Decision Matrix

	Criteria	Grid Projection (Column Matrix)	Grid Projection (Dot Matrix)	Ultrasonic Topography	Diffused Laser (LIDAR)	3D Scanner
	Resolution		1	0	1	-1
	Ease of use		0	-1	-1	1
	Ease of installation		0	0	-1	1
	Use for different models/colors	D	0	1	1	0
	Robustness	A	0	-1	0	-1
	Ease of image processing	T	1	0	0	1
	Use in ambient conditions	U	0	1	-1	1
	Size and weight	M	0	0	0	1
	Cost		0	0	-1	-1
	Portability of whole device		0	0	0	1
	Detection of small dents (15mm dia.)		-1	1	0	0
	Stability as robotic attachment		-1	0	0	-1
		Total Score	0	1	-2	2

3 System Design

The previous team developed a physical prototype and an image processing algorithm that used Fourier transforms. The previous prototype was a meter long LED panel with 3D printed grates on the panel. The panel with grates had the purpose of projecting a column matrix pattern or striped pattern onto car panels. A 3D printed case for the Raspberry Pi and camera was attached at the center of the LED panel. The case also had handles on the sides for the user to grab onto while operating the prototype. During operation, the user would manually pan the prototype over a car panel and the camera would take several pictures auomatically. The pictures would then be juxtaposed into one single image ready for processing. The image would be processed by using a fourier transform analysis and then manually adjusting settings on the GUI. The GUI would then report the number of detected dents to the user. Although the previous team's design sounds viable on paper, in reality the execution leaves much to be desired in two ways. First, the car panel under inspection must be removed so that the prototype can scan over it. This greatly increases the time required to detect hail dents. Second, the fact that the user must manually pan over the car panel means that the images taken by the camera will be of poor and inconsistent quality. The inconsistency then affects the juxtaposing or stitching of multiple images into one. With such a poor quality image, the image processing algorithm is barely able to detect any dents.

Our system design is similar to the previous team's design, reusing their Raspberry Pi, camera, and file transfer method. However, the LED panel is replaced with fluorescent lights which have better intensity, coverage, and lower cost. The striped pattern is produced by using a piece of acrylic covered in window tint and transparency film. The window tint serves the purpose of diffusing the light and the transparency film have striped patterns printed on them. The lights, Raspberry Pi, and the camera are all mounted on the rig. The mount is able to slide on a rail on the rig and is moved by hand. Even though the Raspberry Pi was reused, our team used Python, instead of MATLAB, due to Python's powerful image processing libraries. During operation, the Raspberry Pi camera takes photos while another operator moves the light/camera along the rig. After a total of 23 photos are taken, a third operator accesses the files in Raspberry Pi from a Windows machine via WinSCP SSH and processes the photos using the processing software. A flow schematic for our team's system design is shown in Figure 6

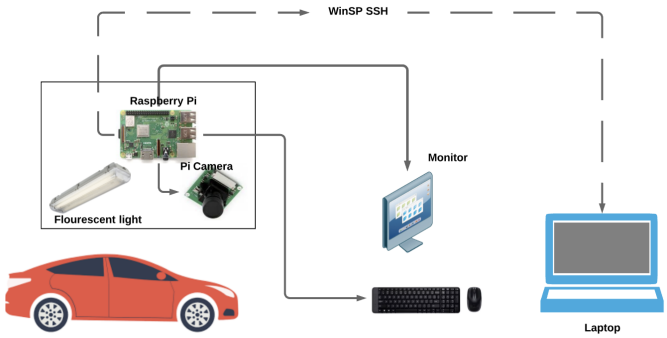

Figure 6. System Schematic

4 Hardware

4.1 Test Stand

A test stand for vehicle inspection was built in order to address two prototype improvements: better image stitching quality and the ability to scan an entire car without removal of damaged panels. Our project team collaborated extensively with Neff Power, a St. Louis engineering firm to build the rail rig system shown in Figure 7.

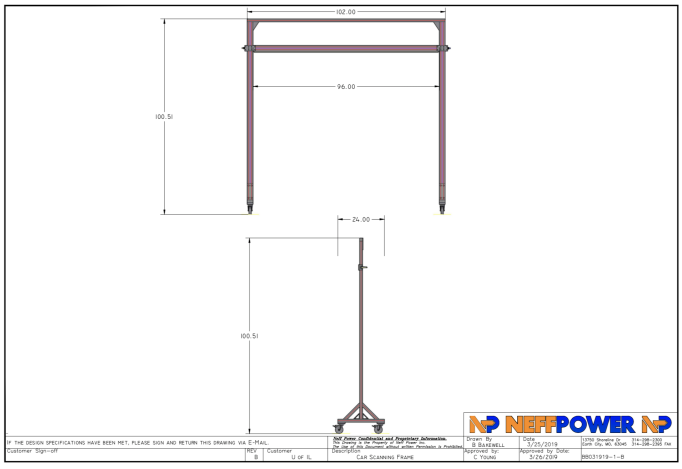

Figure 7. Engineering drawing for test stand

The middle horizontal bar is mounted with a PBC linear guideway, features and details of which are shown in Figure 8.

A mount for fluorescent shop lights as well as the requisite image acquisition hardware

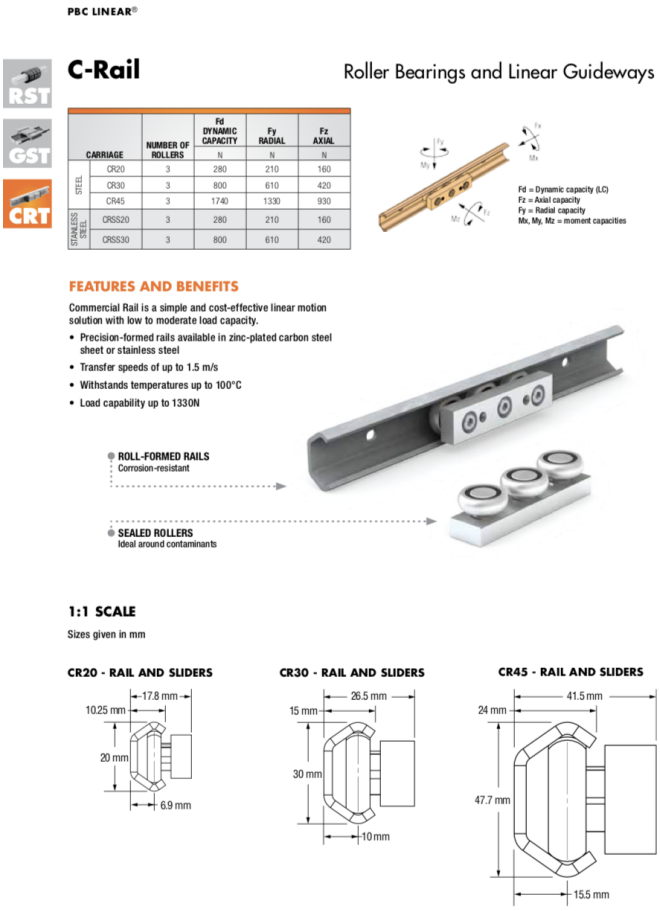

Figure 8. Schematic for imaging apparatus

is attached to the PBC sealed rollers. A schematic for this mount is shown in Figure 9.

The construction of a test stand with roller bearings has greatly improved the results of image stitching. The dimensions of the stand were picked while keeping in mind the goal to be able to be able to scan most cars. The width of the stand is 102 inches as that is roughly the width of the average American parking space. The height is 100.51 inches as the height of the roof of larger SUV's is 72 inches, and the imaging apparatus needs roughly 20 inches of clearance from the panel that is being scanned. An image of the physical test stand with imaging mount is shown in Figure 10.

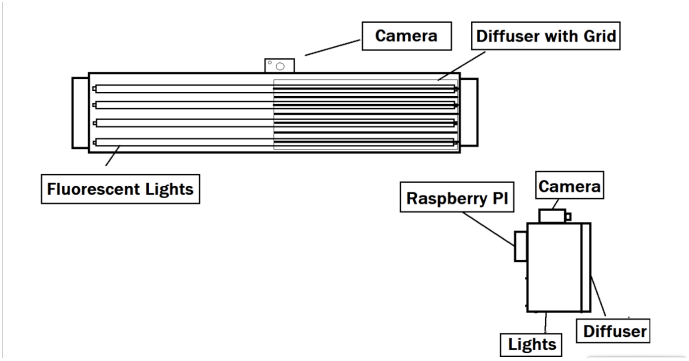

Figure 9. Schematic for imaging apparatus

Figure 10. Image of completed test stand

4.2 Image Acquisition Hardware

The image acquisition components include a Raspberry Pi and a 5-megapixel camera, shown in Figure 11. These components were reused from the previous team's prototype

since the camera provides sufficient resolution and the majority of the cost for our team's prototype was put into building the test stand.

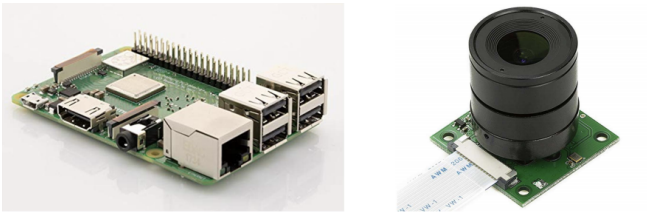

Figure 11. Raspberry Pi and camera

A GUI for image acquisition was developed and installed on the Raspberry Pi, shown in Figure 12. It takes one photo at a time and saves them in the /Desktop/ME470 folder, named "raw01.jpg" through "raw23.jpg", which can be later accessed by the Windows machine via WinSCP. To start and use the GUI, a keyboard and a mouse needs to be plugged into the Raspberry Pi. A monitor is also required and can be connected via HDMI cable.

Figure 12. Raspberry Pi GUI

5 Image Preprocessing

A few preprocessing steps are required before formal detent detection. First, all the raw images need to be stitched together via the `stitch()` function. In the `stitch()` function, the user could set the number of images required to cover the panel as a parameter. Then the program would take an equal portion of each raw image and combine them into a panorama, as shown in Figure 13.

Figure 13. Raw images and stitched image

It should be noted that due to the curvature of the surface, the `stitch()` function will shift the position linearly of the portion taken from each image. The user could manually change shifting inside the `stitch()` function.

During the stitching, each portion of the image will be normalized before stitched together. Normalization is required since the light intensity of the surface could be uneven and it would be difficult for the computer to binarize the image with a threshold. In

our case, we normalize and binarize each image portion instead of the stitched panorama because the binarization according to a local threshold creates a better-quality image.

The next procedure is cropping. The background in the panorama needs to be removed because it may be misidentified as a dent. We applied a smart shape cropping method using a mask image. The mask image contains a shape in white color and the background in black ground. The mask will be read as a grayscale image. The white color pixel will be read as 1 and black as 0. The mask will be manually adjusted to the shape of the panorama and the program will merge two images via a logical `and` operation. The pixels in the panorama merged with the white colored shape will remain in their current state while the pixels merged with the black color background of the mask will be turned black. The user needs to create a mask of the same shape of the parts they scan to correctly remove most of the background.

6 Dent Detection

After the preprocessing, we extract the striped pattern distorted by dents by the shape extraction method from mathematical morphology. First, we choose a short horizontal line consisting of pixels as the structuring element and erode the preprocessed image. The erode operation removes all components except for the long horizontal line shape in the original image.

For clarity, a single image instead of a stitched image is used to demonstrate the process. As shown in Figure 14, the shape extraction method is demonstrated on a binarized image.

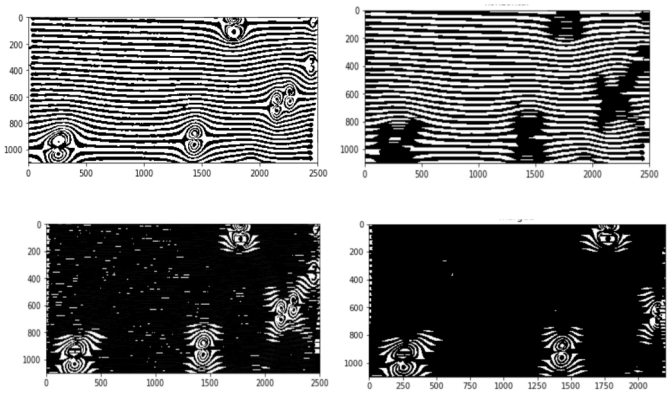

Figure 14. Binarized, extracted, subtracted, and denoised images

A rectangular structuring element is applied to the binarized image and removes everything that does not conform to the rectangular shape, leaving us the straight lines in the photo. This photo is then subtracted from the binarized photo and that leaves the third image in Figure 14, which contains the regions of interest - the distortions caused by dents. However, due to the roughness of a car panel surface, the stripe patterns are not perfectly rectangular, which causes the leftover white noises. These are removed by shifting the third image vertically by a few pixels and combining with the original using a bit-wise-and operator. As seen in the fourth image, the image is denoised.

6.1 Clustering

Continuing from the previous example, a Gaussian blur combined with a sharpening kernel is applied to cluster nearby segments together, as shown in Figure 15.

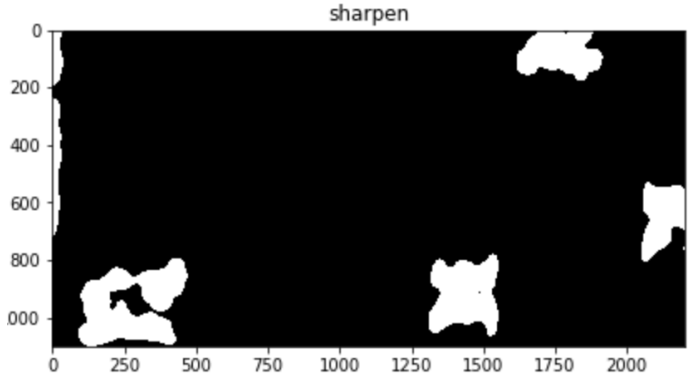

Figure 15. Clustered image

An aspect ratio check and a size check are performed to prevent counting the shapes that are obviously not dents. Then, a labeling function is used to identify them as dents, giving us the pixel areas of all the dents. A rectangle fitting function is used to visualize the dents in the latter image in Figure 16. In the actual implementation, the rectangles are color-coded to indicate sizes.

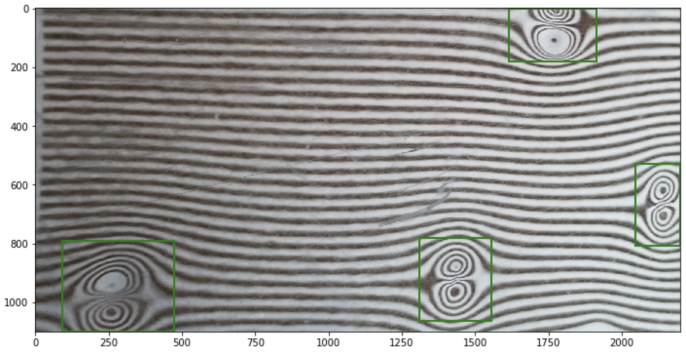

Figure 16. Labeled image

6.2 Conversion

The actual size conversion uses the principle of image formation of lenses. The formulaic form of the principle is:

$$\nu = \frac{u * f}{d} \tag{1}$$

where

u = object actual size

ν = object pixel size

f = focal length

d = distance to lens

We captured the image of a quarter coin at three different distances to the Pi camera. The actual size of the quarter is known (0.955 in). This gives 3 equations with 3 unknowns. By solving the system of equations, we established a linear function between object actual size and object pixel size at the fixed distance (20 in). It should be noted that the size refers to length, not the area. We could square each side of the function to obtain a function between the actual area and pixel area.

For the cost estimation, we need to classify dents into four types according to their area sizes: half dollar, quarter, nickel, and dime. The sizes refer to the area of the coin. According to the function between the actual area and pixel area, the pixel area ranges of each type of dent are shown in Table 2.

Table 2. Mapping from pixel size to coin size for dent detection

Pixel Area Range	Nominal Area in Terms of Coin Sizes
2700, 7000	Half dollar size or greater
2000, 2700	Quarter size
1400, 2000	Nickel size
1000, 1400	Dime size

6.3 Graphical User Interface

Other than the Graphical User Interface (GUI) that is run on the Raspberry Pi to acquire images, there is a processing GUI that is run on the Windows machine to stitch and process the photos. There are two slide bars to tune the input parameters. The first one being the thickness of the structuring element for shape extraction, the second being the Gaussian blur intensity. They are set at a default value, and typically don't need to be tuned. After clicking through all the buttons, the GUI returns the resultant number of dents in each size category and are visualized using rectangle marks in the image, as seen in Figure 17. The damage condition is assessed according to the assessment chart provided by Ally. The full-size result image is saved in the same folder. The software code and installation instruction are in given in Appendix C and Appendix D, respectively. However, the actual installation requires other files, which will be handed over separately.

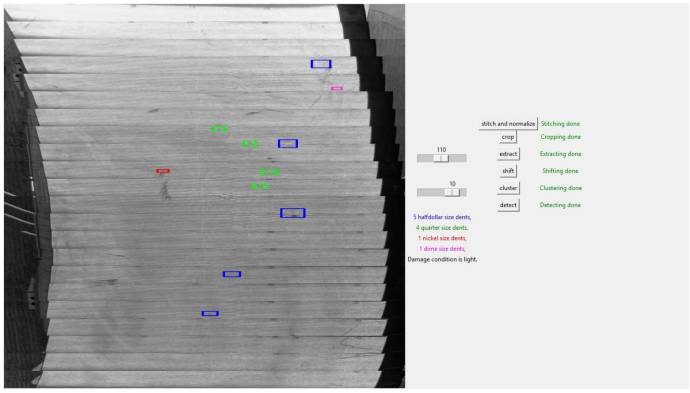

Figure 17. Processing GUI

7 Budget

Table 3 reflects the total cost incurred for this project. The majority of the budget was spent purchasing parts for the rail rig. Image pre-processing and post-processing was accomplished using open-source software, so not expense was necessary for these aspects of the project.

Table 3. Budget

Item	**Unit Price** $/Unit	**Quantity** Unit	**Total** $
Rail Rig	-	-	940.33
Fluorescent Lights	4.99	3	14.97
Shoplight Frames	17.98	2	35.96
T-nuts	2.30	7	16.10
L-brackets, washers, hex screws	-	-	21.18
Acyrlic	13.48	2	26.96
Transparency Sheets	-	-	43.19
Window Frost Application	-	-	31.46
Miscellaneous Prototyping Materials	-	-	193.88
Total Cost			$1,324.03

8 Conclusions and Recommendations

Over the course of the semester, the work of the previous ME470 Hail detection team was analysed and a new system was designed to detect hail dents for Ally Financial. Design decisions were made based on the advantages and shortcomings of the previous prototype as well as the needs of the sponsor. The distortion imaging method used by the previous team was retained, however the LED panel was replaced with fluorescent lights (which are currently used in the industry to assess hail damage). The raspberry pi and the pi camera were retained as they seemed to be the optimal imaging technology for the scope of this project. A new test stand, with a sliding rail and adjustable height, was designed and fabricated to minimise human error in image stitching which arose due to the manual operation of the previous prototype. The image processing software was rewritten. The fourier transform method was replaced by a shape extraction and computer vision labeling algorithm. Two separate graphical user interfaces, one for image acquisition and the other

for image processing, were designed to make the prototype easy to use.

The final prototype delivered is capable of scanning the flat panels of vehicles and detect hail dents with up to 70% accuracy, which is a significant improvement from the previous technology. Additionally, the new prototype is also capable of working on car panels of all colours and finishes.

However, as there are with every prototype, there is still room for improvement. The greatest drawback is the stitching algorithm, which requires a certain amount of manual tuning based on the shape and size of the panel being scanned, the background in the images, and the ambient lighting conditions. A smarter, more automated stitching algorithm could significantly improve results. Further study of computer vision would improve stitching, especially for surfaces with large curved features. One possible solution is to detect the boundary of the region under scanning and then crop the image according to the boundary. Utilising a significantly larger lighting module, along with a camera with a higher resolution would eliminate the need for stitching and significantly improve results. However, this avenue could not be explored due to budgetary constraints.

Another future step could be to automate the scanning process. The current prototype requires the camera to be moved manually along the rail. But manually moving the light-camera system could still induce error. An automatic system that periodically moves the light-camera system with a fixed distance would exclude any possibility of human error and make the scanning process more convenient. A rotary encoder could also be implemented to digitise the position of the camera at all points. This will also enable the user to stitch images with much greater quality.

Finally, the current method of pattern projection is not suitable for surfaces with large features, such as air vents or other pieces of body work. Alternate methods such as 3D scanning would be more suitable, however with the current state of the technology, mobile 3D scanners do not provide a resolution high enough to identify features that are the size of a hail dent.

A Acknowledgement

We would like to thank Ally Financial for commissioning this project. We'd like to thank Dr. Blake Johnson, the course TA's and the mechanical engineering department for running this course. We'd like to thank our faculty advisor Professor Sascha Hilgenfeldt and our teaching assistant Christopher Marry for guiding and advising us throughout the semester. We'd like to thank our Grainger Library Assistant Bethany Murphy for assisting us with our research throughout the semester. We'd like to thank Neff Power for assisting us with the design for the test stand and for helping us source the materials required to build it. We'd like to thank the Imaging Technology Group at the Beckman Institute for Advanced Sciences, for helping us determine the optimal imaging method for the scope of this project. We'd like to thank the people at the MechSE Undergraduate office, and the MechSE Business for helping us with the logistics and purchasing required for this project. Finally, we'd like to thank the Hail Detection Team from Fall 2018 for all their hard work, and for setting us up for a successful semester

B Standards

We worked to comply with two standards: ANSI C78.5-2017 and ISO/IEC 7942-1. ANSI C78.5-2017 applies to fluorescent lights and their use. The lights are required to have an input voltage of 120 or 277 volts for 60 Hz. Our project used fluorescent lights, which complied with the standard. The fluorescent lights were purchased from a store, which would have to be in compliance to the standards. ISO/IEC 7942-1 applied to our project due to the fact that it covers image processing. The standard covers the modification, storage, and production of two dimensional pictures on devices. Our image processing algorithm would fall under this standard.

C Dent Detection Python Code

```
Class
from tkinter import *
from PIL import ImageTk,Image
import tkinter
import cv2
import numpy as np
```

```
from skimage import io
import skimage.morphology as morp
from skimage.filters import rank
from skimage.transform import AffineTransform, warp
from skimage import data
from skimage.segmentation import clear_border
from skimage.filters import threshold_otsu
from skimage.measure import label, regionprops
from skimage.morphology import closing, square
from skimage.color import label2rgb
from skimage.filters import gaussian
import skimage
import time

root = Tk()
root.title("Dent Detector")
upper = Canvas(root, width = 1000, height = 1000)
upper.grid(row=0, column=0)
canvas = Canvas(upper, width = 1000, height = 1000)
canvas.grid(row=0, column=0)
control = Canvas(upper, width = 200, height = 1000)
control.grid(row=0, column=1)
lower = Canvas(root, width = 1000, height = 50)
lower.grid(row=1, column=0)


result = Label(lower, text="Click the buttons from top down, like what a
    normal human would do.")
result.grid(row=0, column=0)
background = np.ones([1080, 1920], dtype=np.uint8)
cv2.imwrite("background.jpg",background)
origin = Image.open("background.jpg")
origin = origin.resize((800, 800), Image.ANTIALIAS)
origin = ImageTk.PhotoImage(origin)
panel = tkinter.Label(canvas, image=origin)
panel.grid(row=0, column=0)

class RawImage:
    def __init__(self,rawinput = None):
        self.rawinput = rawinput
        self.norm = None
```

```
        self.denoised = None
        self.extracted = None
        self.shifted = None
        self.clustered = None
        self.result = None
        self.detectNumber = None
        self.detectarr = []
    def crop(self, ds1 = 50,ds2 = 20,t_norm = None):#ds1 equalize disk size;
        ds2 enhance contrast disk size
        self.rawinput = cv2.imread('origin.jpg')
        img= cv2.imread('norm.jpg')
        mask = cv2.imread("mask.jpg")
        mask = cv2.resize(mask, (img.shape[1], img.shape[0]))
        masked = cv2.bitwise_and(img, mask)
        cv2.imwrite("crop.jpg", masked)
        self.norm = cv2.imread('crop.jpg')
        # show
        imgshow = Image.open("crop.jpg")
        imgshow = imgshow.resize((800, 800), Image.ANTIALIAS)
        imgshow = ImageTk.PhotoImage(imgshow)
        panel.configure(image=imgshow)
        panel.image = imgshow
        # update text
        t_norm.destroy()
        done = Label(control, text="Cropping done", foreground="green")
        done.grid(row=2, column=2)
        return


    def extract(self,save = False, t_extract = None,param = 1):
        # process
        bw = self.norm
        horizontal = np.copy(bw)
        # Create structure element for extracting horizontal lines through
            morphology operations
        cols = horizontal.shape[1]
        horizontal_size = int(cols / param)
        horizontalStructure = cv2.getStructuringElement(cv2.MORPH_CROSS, (
            horizontal_size,2))
        horizontal = cv2.erode(horizontal, horizontalStructure)
        horizontal = cv2.dilate(horizontal, horizontalStructure)
```

```
        subtracted = cv2.subtract(bw, horizontal)
        self.extracted = subtracted
        cv2.imwrite("extracted.jpg", subtracted)
        # show
        imgshow = Image.open("extracted.jpg")
        imgshow = imgshow.resize((800, 800), Image.ANTIALIAS)
        imgshow = ImageTk.PhotoImage(imgshow)
        panel.configure(image=imgshow)
        panel.image = imgshow
        # update text
        t_extract.destroy()
        done = Label(control, text="Extracting done", foreground="green")
        done.grid(row=3, column=2)

    def Shift(self, vector = (0,3),t_shift = None):
        # process
        img = self.extracted
        transform = AffineTransform(translation=vector)
        shifted = warp(img, transform, mode='wrap', preserve_range=True)
        shifted = shifted.astype(img.dtype)
        anded = cv2.bitwise_and(img, shifted)
        self.shifted = anded
        cv2.imwrite("shifted.jpg", anded)
        # show
        imgshow = Image.open("shifted.jpg")
        imgshow = imgshow.resize((800, 800), Image.ANTIALIAS)
        imgshow = ImageTk.PhotoImage(imgshow)
        panel.configure(image=imgshow)
        panel.image = imgshow
        # update text
        t_shift.destroy()
        done = Label(control, text="Shifting done", foreground="green")
        done.grid(row=4, column=2)
    def Cluster(self,t_cluster = None, param=10):
        img = self.shifted
        skgaussian = gaussian(img, sigma=param)
        _, threshed = cv2.threshold(skgaussian, 0.1, 255, cv2.THRESH_BINARY)
        skgaussian2 = gaussian(threshed, sigma=0)
        mykernel = np.array([[-1, -1, -1], [-1, 10, -1], [-1, -1, -1]])
        sharpen = cv2.filter2D(skgaussian2, -1, mykernel)
        self.clustered = sharpen
```

```
        cv2.imwrite("clustered.jpg", sharpen)
        # show
        imgshow = Image.open("clustered.jpg")
        imgshow = imgshow.resize((800, 800), Image.ANTIALIAS)
        imgshow = ImageTk.PhotoImage(imgshow)
        panel.configure(image=imgshow)
        panel.image = imgshow
        # update text
        t_cluster.destroy()
        done = Label(control, text="Clustering done", foreground="green")
        done.grid(row=5, column=2)
        return

    def Detection(self,t_detect = None,result = None):
        # process
        origin = self.rawinput
        image =skimage.color.rgb2gray(self.clustered)
        thresh = threshold_otsu(image)
        bw = closing(image > thresh, square(1))
        bw = clear_border(bw)
        label_image = label(bw)
        quarter, nickel, dime,halfdollar = 0, 0, 0,0
        my_color = (0,0,0)
        for region in regionprops(label_image):
            # take regions with large enough areas

            if 7.8311e+04 > region.area >= 1082:
                # print("region area: %d" %region.area)
                minr, minc, maxr, maxc = region.bbox
                if float(maxc - minc) / float(maxr - minr) > 0.667 and float(
                    maxc - minc) / float(maxr - minr) < 1.5:
                    if region.area > 2.8689e+03:
                        halfdollar += 1
                        my_color = (255,0,0)
                    elif  2.8689e+03>region.area > 1.9561e+03:
                        quarter += 1
                        my_color = (0, 255 , 0)
                    elif 1.9561e+03>region.area > 1.3776e+03:
                        nickel += 1
                        my_color = (0, 0, 255)
                    elif 1.3776e+03>region.area >= 1082:
```

```
                    dime += 1
                    my_color = (204, 51, 255)

                # draw rectangle around segmented area

                cv2.rectangle(origin, (minc, minr + (maxr - minr)), (minc
                    + (maxc - minc), minr), my_color, 10)
    self.result = origin
    cv2.imwrite("detected.jpg", origin)
    # show
    imgshow = Image.open("detected.jpg")
    imgshow = imgshow.resize((800, 800), Image.ANTIALIAS)
    imgshow = ImageTk.PhotoImage(imgshow)
    panel.configure(image=imgshow)
    panel.image = imgshow
    # update text
    t_detect.destroy()
    done = Label(control, text="Detecting done", foreground="green")
    done.grid(row=6, column=2)
    # show result
    if halfdollar + quarter + nickel + dime <= 5:
        value = dime * 65 + nickel * 75 + quarter * 100
        condition = "very light"
    elif halfdollar + quarter + nickel + dime <= 15:
        value = dime * 100 + nickel * 125 + quarter * 150
        condition = "light"
    elif halfdollar + quarter + nickel + dime <= 30:
        value = dime * 150 + nickel * 200 + quarter * 225
        condition = "moderate"
    else:
        value = dime * 200 + nickel * 225 + quarter * 225
        condition = "severe"
    result.destroy()
    result1 = Label(control, text="{} halfdollar size dents,".format(
        halfdollar), foreground="blue")
    result2 = Label(control, text="{} quarter size dents,".format(quarter
        ), foreground="green")
    result3 = Label(control, text="{} nickel size dents,".format(nickel),
         foreground="red")
    result4 = Label(control, text="{} dime size dents,".format(dime),
        foreground="magenta")
```

```
        result5 = Label(control, text="Damage condition is {}.".format(
            condition), foreground="black")
        result1.grid(row=7)
        result2.grid(row=8)
        result3.grid(row=9)
        result4.grid(row=10)
        result5.grid(row=11)
raw = RawImage()


def normalizeforStitch(img, ds1=50, ds2=20, t_norm=None):  # ds1 equalize
    disk size;ds2 enhance contrast disk size
    kernel = morp.disk(ds1)
    kernel2 = morp.disk(ds2)
    norm_image = rank.equalize(img, selem=kernel)
    norm_image = rank.enhance_contrast(norm_image, selem=kernel2)
    # show
    return norm_image

def stitch(num = 23,mode = 'auto',arr = [],b_norm = None):
    # process
    background = np.ones([int((num)*1080/4.5),1920],dtype=np.uint8)
    background2 = np.ones([int((num)*1080/4.5),1920],dtype=np.uint8)
    background.fill(255)
    length = len(background)
    image = []
    for i in range(1,num+1):
        image.append(cv2.imread("raw" + str(i)+'.jpg',0))
    sec = int(1080/4.5)
    mid = int(1080/2)
    start = 0
    if mode == 'auto':
        k = 180
        for i in range(0,num):
            if i == 0:
                start = i*sec
            else:
                start = i*sec-50
                temp = int(1080*5/12 + 190 - i*(12+i/9))
                obj = image[i][temp:temp+sec][0:1920]
                background[start:start+sec]= obj
```

```
                obj2 = normalizeforStitch(img = obj)
                background2[start:start+sec]= obj2
    if mode == 'manual':
        count = 0
        for i in arr:
            start += i
            if start < length:
                if start + sec < length:
                    for j in range(0, sec):
                        background[start + j] = image[count][mid - int(sec /
                            2) + j][0:1920]
                else:
                    for j in range(0, length - start):
                        background[start + j] = image[count][mid - int(sec /
                            2) + j][0:1920]
            else:
                break
            count+=1


    cv2.imwrite('origin.jpg',background)
    cv2.imwrite('norm.jpg', background2)
    # show
    imgshow = Image.open("norm.jpg")
    imgshow = imgshow.resize((800, 800), Image.ANTIALIAS)
    imgshow = ImageTk.PhotoImage(imgshow)
    panel.configure(image=imgshow)
    panel.image = imgshow
    # update text
    b_norm.destroy()
    done = Label(control, text="Stitching done", foreground="green")
    done.grid(row=1, column=2)
```

Main

```
from Dent import *
#
b_stitch = Button(control, text="stitch and normalize", activeforeground="
    blue")
b_norms = Label(control, text="normalize takes about 10s", foreground="blue")
b_norms.grid(row=1, column=2)
```

```
b_stitch['command'] = lambda text=b_norms: stitch(b_norm = text)
b_stitch.grid(row=1, column=1)

# update button
t_norm = Label(control, text="crop take less than 1s", foreground="blue")
t_norm.grid(row=2, column=2)
b_norm = Button(control, text="crop", activeforeground="blue")
b_norm['command'] = lambda text=t_norm: raw.crop(t_norm = text)
b_norm.grid(row=2, column=1)

# update button
# slider
var = DoubleVar()
slider = Scale(control, variable=var, orient=HORIZONTAL, to=10, from_=200)
slider.set(110)
slider.grid(row=3, column=0)
# button
t_extract = Label(control, text="Extracting takes 2s", foreground="blue")
t_extract.grid(row=3, column=2)
b_extract = Button(control, text="extract", activeforeground="blue")
b_extract['command'] = lambda text=t_extract, slider=slider: raw.extract(
    t_extract = text, param= var.get())
b_extract.grid(row=3, column=1)

# update button
t_shift = Label(control, text="Shifting takes 4s", foreground="blue")
t_shift.grid(row=4, column=2)
b_shift = Button(control, text="shift", activeforeground="blue")
b_shift['command'] = lambda text=t_shift: raw.Shift(t_shift=text)
b_shift.grid(row=4, column=1)

# update button
# slider
var2 = DoubleVar()
slider2 = Scale(control, variable=var2, orient=HORIZONTAL, to=5, from_=30)
slider2.set(10)
slider2.grid(row=5, column=0)

t_cluster = Label(control, text="Clustering takes 7s", foreground="blue")
t_cluster.grid(row=5, column=2)
b_cluster = Button(control, text="cluster", activeforeground="blue")
```

```
b_cluster['command'] = lambda text=t_cluster, slider=slider2: raw.Cluster(
    t_cluster= text, param= var2.get())
b_cluster.grid(row=5, column=1)

# update button
t_label = Label(control, text="detecting takes 1s", foreground="blue")
t_label.grid(row=6, column=2)
b_detect = Button(control, text="detect", activeforeground="blue")
b_detect['command'] = lambda text=t_label: raw.Detection(t_detect=text,
    result=result)
b_detect.grid(row=6, column=1)

root.mainloop()
```

D User Manual

D.1 Vehicle-hail-dent-detection

This is the software part of the hail detection system, and should be used along with the hardware system. Download this folder to your desktop, then start terminal to type in the following commands. You can also skip to the dent detection section and run the software with pre-taken photos in the folder.

D.2 Image Acquisition

This is the GUI installation guide for Raspberry Pi.

D.3 Installing

(If you are our sponsor and have our Rasberry Pi, you don't need to install anything, just skip to running.)

Install Python3

```
sudo apt-get install python3-dev libffi-dev libssl-dev -y
```

Install Tkinter

```
sudo apt-get install python-tk
```

Copy the pigui.py file into a folder

D.4 Running

Go to the folder of the pigui.py, for example:

`cd Desktop/ME470`

Run program

`python3 pigui.py`

Click "clear files" to delete old images. Then click take photo for 23 times as another operator pushes the light along the 23 ticks. Then switch to a Windows machine and use WinSCP to copy the 23 raw images into the working folder.

D.5 File Transfer

Obtain Raspberry Pi MAC address:

First turn on SSH: Type the following command in Raspberry Pi terminal

`sudo raspi-config`

Choose 'Interfacing options' and then click on 'P2 SSH' to enable SSH Then obtain IP dress: Type the following command in Raspberry Pi terminal

`ifconfig wlan0`

The IP address should be something like "192.168.1.10"

Install WinSCP

Connect to Raspberry Pi

Host name is the ip address of Raspberry Pi. (port number: 22, username: pi, password: me470)

Copy all images from "raw1.jpg" to "raw23.jpg" to Windows machine. Make sure to store them in the same folder as the files from here.

D.6 Dent Detection

It is recommended to install and run on Windows.

`cd Desktop/Vehicle-hail-dent-detection/`

D.7 Installing Software

Install virtual environment

`pip install virtualenv`

Create virtual environment for python3

`virtualenv -p python3 env`

Start virtual environment

`source env/bin/activate`

install dependencies

`pip install -r requirements.txt`

D.8 Running

`python3 Simpler.py`

Assuming there are already 23 raw images copied over from raspberry pi, the program should run with no problem. Click through the buttons. After the last step, go to the folder and open detected.jpg for a clearer result.

D.9 Finishing

Quit virtual environment

`deactivate`

References

[1] B. Denkena, H. Ahlers, F. Berg, T. Wolf, and H. Tönshoff, "Fast inspection of larger sized curved surfaces by stripe projection," pp. 499–502, 2002.

[2] J. Che, Q. Gu, T. Aoyama, T. Takaki, and I. Ishii, "Blink-spot projection method for fast three-dimensional shape measurement," pp. 430–443, 2015.

[3] M. M. Munir, M. A. Billah, A. Surachman, M. Budiman, and Khairurrijal, "Design of 3d scanner for surface contour mapping by ultrasonic sensor," *American Institute of Physics*, 2015.

[4] L. Torres, G. Vignal, K. Korol, J. Sutherland, and S. Tappert, "Detection of crack-related features within dented pipe using electromagnetic acoustic transduction (emat) technology," *American Society of Mechanical Engineers*, 2016, 2016 11th International Pipeline Conference Calgary, Alberta, Canada.

[5] C. Z, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2014.

[6] J. Liu, *IEEE 3rd Optoelectronics Global Conference (OGC)*, 2018.