A preliminary study of remote control skidder operation

by

Chennan Xue

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 6, 2016

Keywords: skidder, gaze tracking, pose estimation, heat map, video-based driving

Approved by

Abstract

In timber harvesting, skidders are widely used by operators who are in charge of pulling the woods. However, operators have to visually monitor multifarious objects such as loggers, machinery, trail, terrain and trees are being cut for the sake of work plan and personal safety. Since distraction may lead to poor efficiency, injuries or even death, it is indispensable to determine not only how the operators handle the skidder but also what they look at along with the distribution of their attention and why they concentrate on specific area under relevant task. Three GoPro cameras were responsible for the 3D orientation model of the gaze. The first GoPro camera was attached to bracket in front of the hard hat in order to get the frame of the gaze. The second GoPro camera was mounted on the back of the cab so as to capture the head motion during forward looking. The third GoPro camera was then installed on the left side of cab so that the head action during backward looking can be tracked. A computer was then used to process the image, restore the view of the operators, calculate the density of the gaze behaviors and generate a heat map of the gaze overlaid by a virtual perspective. In order to reduce the accident rate and fatigue caused by the long time in-cab timber harvesting operation, as well as improve the efficiency such as multiple tasks execution, the video-based driving system built for testing the feasibility of remote driving was set up in the cab. A fisheye camera was installed on the head of the skidder, a monitor was mounted on the region over the steering wheel, and a laptop which was in charge of process the video was connected with the cameras and the monitor. Also, the GPS information like location and speed was recorded. The results indicated that the remote control based on the live video was feasible.

Acknowledgments

The completion of this study was a team effort. First of all, I would like to thank my advisor Dr. Timothy McDonald for his support that provided me with so much. He is also the mentor of my life and provided me an opportunity to study in the U.S. I would like to thank Dr. Mathew Smidt and Dr. Jeremiah Davis for being my committee, supporting the experimental facilities and giving suggestions for my thesis. I would like to thank Dr. Thomas Gallagher and his students who helped me operating equipment. I also would like to thank Dr. Yifen Wang who introduced me to biosystems engineering in Auburn University. I would like to send special thanks to Caterpillar and its employees for the fabrication and operation of the equipment. I would like to thank all the students and staffs in our department for their kind help, especially Rees Bridges, Pengmin Pan and Dawayne ”Doc” Flynn. I also want to acknowledge my father Mr. Rujun Xue, my mother Mrs. Jinghua Chen and all my families for their love, support and encouragement through this process.

Contents

Abstract ii
Acknowledgments iii
List of Figures vii
List of Tables x
1 Introduction 1
1.1 Background 1
1.2 Objectives 2
1.3 Organization of thesis 2
2 Literature Review 4
2.1 Human eye structure 4
2.2 Object detection 5
2.3 Gaze tracking methods 7
2.3.1 Limbus tracking 8
2.3.2 Pupil tracking 8
2.3.3 Cornea-pupil reflection 9
2.3.4 Video-based 10
2.3.5 Electrooculography-based 12
3 Gaze tracking system design 13
3.1 Program language 13
3.2 Cameras and mounts 14
3.3 Face and eye detection 14
3.3.1 Haar-like feature detection 15
3.3.2 Weak classifier 16

3.3.3 Strong classifier 17
3.3.4 Cascade classifiers 19
3.3.5 Integral image 20
3.3.6 Face and eye detection implementation 21
3.4 Pupil tracking design 22
3.4.1 Binarization 23
3.4.2 Morphological transformations 26
3.4.3 Histogram equalization 28
3.4.4 Loop steps 28
3.4.5 Main program test 29
3.4.6 Algorithm performance 32
3.5 Head motion estimation system design 36
3.5.1 Camera calibration 36
3.5.2 Pose estimation 38
3.5.3 Loop steps 38
3.5.4 Head pose algorithm test 39
3.5.5 Calibration 40
3.5.6 Accuracy 41
3.6 Conclusions 42
4 Gaze estimation with head motion 44
4.1 General algorithm 44
4.1.1 Definition of parameters 44
4.1.2 Gaze vectors 46
4.1.3 Gaze points 50
4.2 Expression of gaze points 51
4.2.1 Heat map 51
4.2.2 Reliability test under lab condition 52

4.2.3 Field calibration 58
4.3 Tracking system field test 60
4.4 Results from heat maps 62
4.5 Conclusions 64
5 Video-based driving test 65
5.1 Video system setup 65
5.2 Path mapping 67
5.3 Results and discussions 69
6 Conclusions and future work 72
6.1 Conclusions 72
6.1.1 Hardware and software 72
6.1.2 Tracking system and results 72
6.1.3 Video-based driving test 73
6.2 Future work 73
6.2.1 Hardware and software 73
6.2.2 Methods of tracking 73
6.2.3 Accuracy of the overall tracking system 74
6.2.4 Video-based driving 74
Appendices 79
A Python code 80
A.1 Gaze tracking 80
A.2 Head pose estimation 83
B Equipment 87

List of Figures

2.1 Human eye structure 4

2.2 Purkinje image 9

3.1 Haar-like features 15

3.2 Flow diagram of weak classifier 17

3.3 Integral image 21

3.4 Face and eyes detected within rectangles 23

3.5 A target image and its histogram 25

3.6 Two types of morphological transformation 27

3.7 Comparison of image and histogram before and after equalization 29

3.8 Flow diagram of the pupil tracker 30

3.9 A detected pupil image and sub-images in progress: (a) pupil detection in one frame; (b) grayscale; (c) binarization & morphological transformations; (d) pupil image 31

3.10 Pupil detection with front looking 32

3.11 Pupil detection with left looking 33

3.12 Pupil detection with right looking 33

3.13 Pupil detection with up looking 34

3.14 Pupil detection with down looking 34

3.15 Pupil detection failure condition 35

3.16 Flow diagram of head pose estimation 39

3.17 Pose estimation of four directions 40

3.18 Calibration of pose estimation 43

3.19 Limitation cases of pose estimation 43

4.1 3D vectors in the skidder cab 44

4.2 A cropped window of eye 45

4.3 Head and eye coordinates 47

4.4 Top view of eye movement 48

4.5 Sight range test with head fixed 49

4.6 HSL and grayscale color model 52

4.7 Inputs of sight expansion: (a) pupil center location; (b) head pose estimation . . 53

4.8 Calculated gaze points for fixed head orientations while rotating the eyes to look at a sequence of 5 locations. 54

4.9 Boxplots of four positons 55

4.10 Inputs of sight focus: (a) pupil center location; (b) head pose estimation 56

4.11 Gaze points of sight focus 57

4.12 Boxplot of the gaze points 57

4.13 Top view of the tracking system as deployed in the skidder cab. 58

4.14 Gaze points calibration in the skidder cab 59

4.15 Example tracking frame 61

4.16 Example tracking frame: noise filter 61

4.17 Example tracking frame: eye switched 62

4.18 Views in the cab with heat map added 63

4.19 Relative histogram 63

5.1 Fisheye cam: ELP-USBFHD01M-L180 66

5.2 Fisheye camera set on the front of both skidders 66

5.3 Vision comparison between driver's vision and live video 67

5.4 Skidder path in MOT Demonstration Forest 67

5.5 Road condition recording: (a) GoPro camera setup; (b) Example frame of bad road condition 68

5.6 Path mapping by vertical acceleration with severe road conditions 68

5.7 A view of the skidder cab covered by paper 69

5.8 Speed-location comparisons on mini skidder: (a) Single lap; (b) Multiple laps . . 71

5.9 Speed-location comparison on Caterpillar 555D 71

B.1 Hard hat designed for tracking gaze and head motion 87

B.2 Caterpillar skidder used for gaze and head motion tracking tests 87

B.3 TurboForest mini skidder with windows covered during driving tests 88

List of Tables

2.1 Parameters of the eye 5

3.1 Comparison of the binarization methods 26

3.2 Results of calibration 41

5.1 Average speed and overall time comparison 70

Chapter 1

Introduction

1.1 Background

Skidders are used in extracting tree-length stems in many forest harvesting systems. The machines are large and operate under difficult conditions that, over time, reportedly expose operators to elevated risk of injury (Slappendel paper). They are quite heavy and their use can also lead to negative soil impacts on harvested tracts (any number of citations). They are, however, very efficient, have relatively low cost, and are the preferred method for primary extraction of timber in the southeastern US.

Recent advances in autonomous vehicle research suggests the possibility of removing the operator from a skidder in timber harvesting. This could be of benefit in a) reducing injury rates in harvesting operations, b)eliminating the need for a cab and decreasing machine weight and, therefore, potential for soil damage, and c) a single remote operator may be able to control multiple machines, lowering variable costs. This work was undertaken to simulate a remotely controlled skidder as a precursor to a fully autonomous system, and evaluate its performance relative to a conventional skidder.

Running a remotely controlled vehicle depends on the ability of the operator to react appropriately to the environment in which the machine is working. For skidders, this would imply driving the machine through a forest along *ad hoc* trails with other machines working nearby. Accomplishing this reliably from a remote location means the operator must sense the working environment of the machine as completely as when operated normally. Much of the feedback to an on-board driver would be visual in nature and, therefore, reproducing the surrounding environment visible from the cab to the remote operator would be essential. If operating multiple machines, providing the remote operator with prioritized views containing the most critical information would also be important to minimize distractions and optimize performance. This study was therefore also designed to understand the relative importance of sight lines from within the cab towards which a driver focused the majority of their attention. It was felt these views would be the most crucial to provide a remote driver to maximize the likelihood of success.

1.2 Objectives

The goal of this study was to evaluate assistive technology to get operators out of skidder cabs and, based on this, two main sub-objectives were developed. One was to follow operators' visual attention through developing a camera-based system for tracking gaze and head motion. Output of the two systems was combined to generate a 'map' of visual interest that would inform placement of cameras for remote operation. Another sub-objective was to test skidder productivity when operated remotely. Because this goal required a skidder capable of remote operation and one was not available, a 'blindfolded' driving system was built and tested. The scope of this study encompassed the following specific tasks:

(1) Design, build, and verify both gaze and head motion tracking systems. Test the reliability of the system under lab conditions.

(2) Calibrate the tracking system while in operation and generate heat maps for at least one driver. Set views for remote operation based on the results of the heat maps.

(3) Build a video-based driving system to validate the principle of remote driving and estimate any impacts on productivity of the machine during timber harvesting.

1.3 Organization of thesis

Chapter 2 reviews the literature relating to human eye structure, methods of object detection, and methods of gaze tracking. Chapter 3 is a detailed description of the design of a gaze and head motion tracking system, including principles, functions, methods, and parameters used. Chapter 4 details calibration of the vision system and generation of visual interest maps, while Chapter 5 presents the ‘blindfolded’ driving system and estimates of productivity impacts. Chapter 6 summarizes the overall conclusions and recommendations for future work.

Chapter 2

Literature Review

2.1 Human eye structure

Morimoto and Mimica (2005) summarized the human eye structure. The major components of the human eye was shown in figure 2.1. The eye has an approximately spherical shape with a radius of about 12 mm.

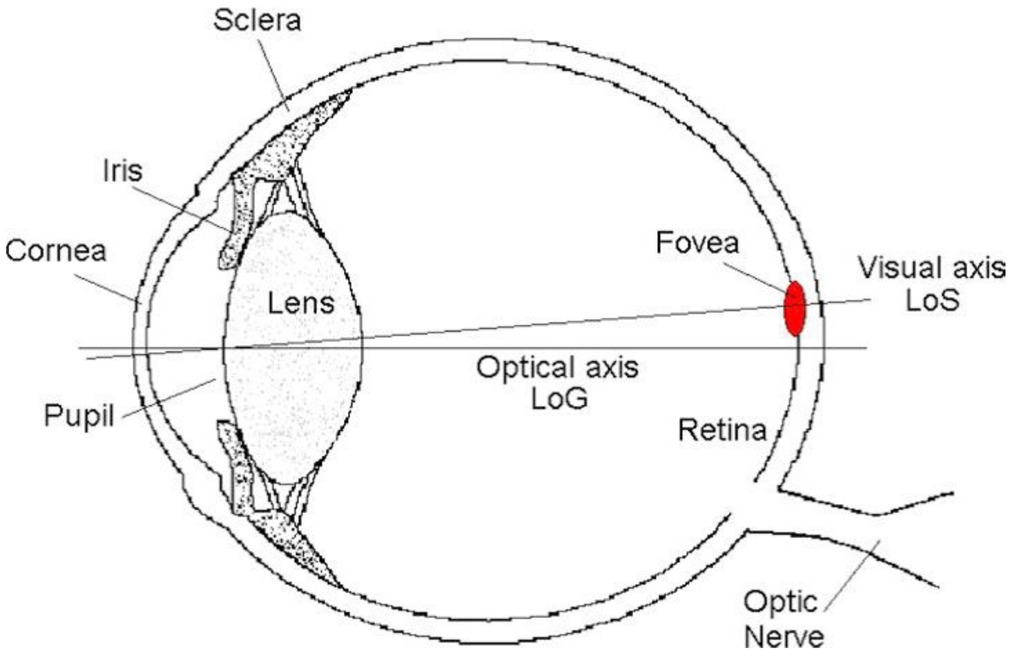


Figure 2.1: Human eye structure

The sclera (the white part), the iris (the color part), and the pupil located in the center of the iris are the external parts of the eye which are visible in the eye socket. The cornea covering the iris is a transparent membrane in charge of protecting the blood vessels. In the center of the iris, a circular aperture is called the pupil. The pupil regulates the amount of light coming into the eye through constantly changing its size. Right behind the iris is the lens. The shape of the lens changes while bringing the image of an object to a sharp focus to the retina. However, the retina is a layer consists of photosensitive cells locating at the back of the eye. The area between the cornea and the lens is filled with watery aqueous humor. And the space between the lens and the retina is the transparent gelatinous vitreous body. The red colored region in the retina is known as fovea, which contains most of the color sensitive cells and is responsible for gathering the details of the scene.

The line of gaze (LoG) is defined as the optical axis of the eye. While the line of sight (LoS) is shown as the line from the fovea through the center of the pupil. Actually, the LoS determines a person's visual attention. But the gaze point can be estimated with the information about objects and the relationship between LoG and LoS. Table 2.1 shows the parameters of the boundary surfaces that are in the light path from cornea to the retina.

Name	**Position (mm)**	**Radius (mm)**	**Refraction index after sureface**
Corneal	0	7.7	1.367
	0.5	6.8	1.336
Lens	3.2	5.33	1.385
	3.8	2.65	1.406
	6.6	-2.65	1.386
	7.2	-5.33	1.336
Retina	24.0	-11.5	

Table 2.1: Parameters of the eye

2.2 Object detection

Object detection is a basic visual research topic, which mainly consists of two different detection tasks: Instance Object Detection and Generic Object Detection. The first task is to detect and locate the perticular one or more objects in the input image, like detecting and judging if an image contains a house in it. This sort of tasks can be the matching problem of the specific object samples and the perticular objects waiting to be detected from the input. The difference between the samples and the detected objects are from the changes of the imaging condition. However, the second task focus on seperating and locating different objects in one input image. For example, detecting and classifying vihecles and pedestrians in one image. Compared with the first one, this task is much more challenging since the visual difference between objects in real world is obvious enough to tell. For the same sort of objects, the difference not onlt results from imaging condition, but also is affected by physical properties. For example, the textural features which can be easily covered by other objects may occupy only a small percentage of the whole scene. That the similar features may exsit in the input image is the big challenge for the object detection.

Two main methods used in object detection were statistic approach and feature-based detection. The statistic approach was initially used in object detection. The processing speed of feature-based detection was slow at that time. Also the detection rate was poor compared with statistic method.

A first statistic method for 3D object detection that can reliably detect faces which varied from frontal view to full profile view was developed by (Schneiderman and Kanade). In this method, they decomposed the 3D geometry of each object into a couple of viewpoints other than the entire object. For each view point, they made a decision rule which determined the object was presented at the defined orientation. The statistics of the object and non-object appearance was applied to each decision rule. Also, it was represented by using a product of histogram. Each histogram indicated the joint statistics of wavelet coefficients and their position on the object. Those histograms was used for presenting a variety of visual attributes. This method was also able to applied for car detection.

Papageorgiou and Poggio (2000) presented a general, trainable system for object detection. This example-based learning approach derived a model of an object class by training a support vector machine classifier through a large set of positive and negative examples. The results were presented on face, car, and people detection. For face detection, detection rate of 90% for every 100,000 patterns and for people detection over 90% accuracy was achieved for every 10,000 patterns processed. Instead of a full pattern approach, a component based approach to car detection that identified different parts of a car including headlights, wheels, and windshield was more efficient.

An example-based framework for object detection in static images by components was developed(Mohan et al., 2001). The system that located people in cluttered scenes was demonstrated with four distinct example-based detections that were trained to find the components of the human body separately. After the components being approved in the proper geometric configuration, a second example-based classifier then combined the results of the component detectors to judge a pattern as either person or non-person. From the results, the system performed significantly better than a similar full-body person detector. The algorithm was also robust than the full-body person detection method.

The feature-based detection method was widely applied after Viola and Jones (2001) built a rapid object detection method using a boosted cascade through detecting several simple features. Also, a new image representation which greatly increased the detecting speed was created by them. Besides, a fast cascade filter allowed the computer to process the object-like image faster. In real-time applications, the detector ran at 15 frames per second without resorting to image differencing or skin color detection. Due to the decision was made by each filter, the accuracy of detection was improved after training the cascade detector.

2.3 Gaze tracking methods

The goal of the gaze tracking system is to determine where the user is looking. In the other word, finding the gaze point is the challenging problem. Video-based tracking system, infrared pupil-cornea reflection tracking system, and electrooculography-based tracking system are mainly used for tracking eye gaze. Generally, existing systems have one of two limitations that either the head should remain fixed in front of the camera, or the user has to wear an obtrusive device in order to allow for head motion.

2.3.1 Limbus tracking

Iris-sclera tracking is also called limbus tracking which tracks the boundary of different two colors in biological tissue. It is the easiest way to separate iris and sclera through image processing. However, this method required a fixed head while acquiring the edge of the iris. Meanwhile, because of that the edge may be blocked by eyelash at vertical direction, Scott and Findlay (1991) pointed that this method was only suitable for measuring horizontal movements of the eyes. Other techniques such as method based on Kalman filter which was raised by Xie et al. (1995) can be combined to increase the accuracy.

2.3.2 Pupil tracking

Compared with the edge of iris and sclera, the contrast between pupil and iris was relatively low. But this method can make up the disadvantage of the dead zone in limbus tracking. In other word, it was not affected by the image of eyelash except blink. To improve the identification rate of the pupil in image, infrared source was later added. Usually the infrared source was placed in the paraxial region of the camera. The light went through pupil and reflected from retina to the camera. Thus, the pupil became a bright pupil other than a dark one. So, the identification rate was promoted. Nguyen et al. (2002) discussed the experimental conditions of the bright pupil.

However, the pupil tracking still cannot ignore the relative movement of the head. Therefore, some other features like eye socket, nose, and mouth were used as reference to eliminate the relative motion in pupil tracking method. Usually, modules of face feature detection were included in these algorithms. The gaze direction was then calculated by corresponding positions of the pupil and the facial features.

The gaze can be estimated by two ways, both the orientations of head and eyes, in this method. Gaze estimation based on the head motion was to assume that no movements or tiny movements were generated from the eyes. The principle was to capture the image of the face by camera and locate as well as track the face and eyes through software calculation.

The advantage was that no limitation was required for detector's activity within a specific area. The main experimental device was camera. However, due to the low accuracy, it was similar to the image based processing method. But unlike image based mothed, it still required calibration of facial features, gaze, and target plane.

2.3.3 Cornea-pupil reflection

Cornea-pupil reflection method was developed from pupil tracking method. The contrast of iris and pupil can be improved by using infrared source. It also generated the reflection on the cornea and lens which was called Purkinje Image (Fig 2.2). The vector that composed by the center of the pupil and the glint of the cornea changed with eye movement correspondingly.

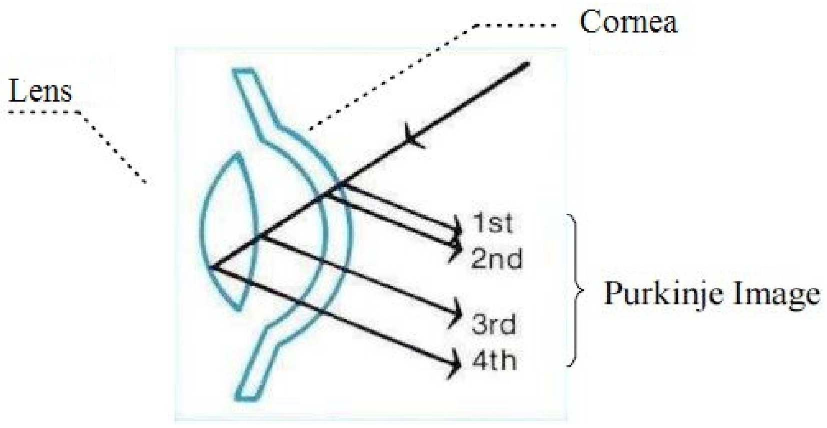


Figure 2.2: Purkinje image

Infrared pupil-cornea reflection tracking system was based on visible light, pupil center, and cornea reflection point. The cornea reflection point kept the eye area well lit without disturbing viewing or pupil dilation due to an infrared light source.

Winfield et al. (2005) built a low-cost head mounted eye tracker with a hybrid algorithm for eye tracking that combines feature-based and model based approaches. Both the cornea reflection and the pupil are located through adaptive feature-based techniques. The results indicated that the error was significantly lower while the vector difference between the pupil center and the cornea reflection center was used compared with that only the pupil center was considered. Also, the results noticed that the radial distortion in each frame was essential to be removed through standard image-processing techniques in order to get better results. Gao et al. (2012) enhanced the contrast between the pupil and the iris by using infrared lighting sources. Distinguishing the target points from the other points by setting a threshold was the general idea of the infrared method. The reflection of the infrared light simplified the process of locating the center of the pupil. This method shortened a potential time consuming task and increased the accuracy of the pupil detection as well. However, there could also be other light sources or noise. Thus, it was hard to calculate the focus point in many cases.

A low-cost, open-source package of hardware and software was designed by Parada et al. (2015). The open software used was called ExpertEyes. From the results, the pupil and cornea reflection were estimated by using a novel forward eye model. The accuracy and precision of the system were better than commercial eye tracking system, with the typical accuracy of less than 0.4° and best accuracy below 0.3°. However, this system was very sensitive to the environment especially the illumination, so it was limited for using under many conditions.

Because of the small interruption of the user, simple principle, and high accuracy, most of the eye trackers for sale from the company, such as SR, ASL, LC IView, and ISCAN, utilize cornea-pupil reflection method.

2.3.4 Video-based

A video-based eye tracking system can be used not only in a remote, but also in a head mounted configuration. The typical setup contained a head-mount video camera that recorded the movement of the eyes. Also a scene camera for recording the user's point of view which was able to map the gaze to the current visual scene was included. In the remote systems, the camera was usually set below the computer screen. However, in the head-mounted systems, the camera was attached either on a frame of eyeglasses or in a custom designed helmet. Head-mounted system often included a scene camera for recording the real point of view, which can be used to map the gaze to the current visual scene.

Morimoto et al. (2002) studied a new model for remote gaze tracking with free head motion. Just a single camera and at least two light sources were used in their study. The simulation results indicated that the accuracy has to be improved by increasing the resolution of the images.

Beymer and Flickner (2003) developed a 3D eye tracking system where head motion is allowed without wearing devices on the head. Instead, they used a pair of wide angle stereo systems to detect the face and steer an active stereo system for sake of tracking the eye at high resolution. With high resolution tracking, the eye was modeled in 3D, including the cornea, pupil, and fovea. With the calibration of the stereo systems, the accuracy of the eye model, eye detection and tracking, as well as estimated gaze point were improved.

Vicente et al. (2015) developed a real-time gaze tracking system motoring if the driver was looking off road by using the video from a camera installed on the steering wheel column. The system contained three main novelties:

(1) Robust face landmark tracker based on the supervised descent method;

(2) Accurate estimation of 3D driver pose, position, and gaze direction robust to non-grid facial deformations;

(3) 3D analysis of car/driver geometry for prediction.

The system was able to detect both at day and night. The overall accuracy of the system achieved above 90% including night time operation. Additionally, the false alarm rate in positive results was below 5%. The experiment data showed that the head pose estimation algorithm was robust to extreme facial deformations.

2.3.5 Electrooculography-based

The eye can be modeled as a dipole with positive pole at the cornea and the negative one at the retina (Majaranta and Bulling, 2014). Electrooculography-based tracking was to attach electrodes to the skin around the eyes. The electrical signal called electrooculogram (EOG) was measured by two pairs of surface electrodes. Once the eyes moved towards one of the electrodes, meanwhile the retina approached this electrode. This change in dipole orientation caused a change in the electric field, which can be measured to track the eye movements. The primary advantage of this method compared with the previous ones was the changing lighting conditions have litter impact on the signals. One drawback of the EOG compared to the previous tracking methods was that the EOG required electrodes to be attached to the skin around eyes. The signals were subject to signal noise and may be corrupted with noises from residential power line, electrodes, or other interfering physiological sources.

Chapter 3
Gaze tracking system design

One objective in this project was to create a gaze tracking system, including pupil and head motion sensing, using simple techniques and inexpensive equipment. Since the logical design of most gaze tracking systems is fairly standard, the design of this system was similar to those existing, although the details of tool selection and integration were unique to this application.

This chapter will present details behind some initial design choices, followed by selection criteria for the particular web camera and programming language employed, and finally move onto the logical and actual design specifics of the gaze tracking system as built for this project.

3.1 Program language

Most of the specifics of any vision system begin with the choice of programming tools used for its implementation. In this project, `OpenCV` (Open Source Computer Vision Library) was used. It is an open source computer vision and machine learning software library that supports five languages as a default: C, C++, `Python`, `Java` and `MATLAB`. The selection of programming language mainly influences ease of development, but also processing speed. `Python` was chosen as the development language in this work because it has simple syntax and there are numerous robust development environments and image processing extensions available, many of which are public domain. Initial image processing tests using `Python` indicated it could operate with acceptable speed and all subsequent development was made using it. All of the algorithms outlined in this chapter were implemented using the functions available in `OpenCV`. The system was developed on a 64-bit Windows 7 Operating System computer with 4GB of RAM and an Intel Core i5 870 2.93GHz Processor. All code was written in `Python` 2.7.10 with `OpenCV` 2.4 & 3.0.

3.2 Cameras and mounts

The web cameras used for development of this gaze tracking system test were Logitech C270 and Microsoft LifeCam Q2F-00013. These were relatively cheap web cameras with limited resolution and suited the purposes of the project well. For field tests, the skidder undergoes severe vibration so a GoPro camera was a good choice because it has built-in image stabilization software. The GoPro models used in these tests were Hero and Hero+. Ram vehicle mounts and a custom designed hard hat were used to hold cameras in place.

3.3 Face and eye detection

The first objective in this project was to develop a system for tracking the focus of visual attention in skidder operators. The general approach applied image processing in accomplishing that task. Cameras were used to capture video images of the operator's eyes and head which were subsequently mapped into a vector defining the line of sight (LoS). The conversion of images to LoS required steps to find the head in an image, then the eyes, and finally the pupils. Each of these steps required interpreting a brightness matrix (an image) to detect an object (face,eyes), which are normally termed "features". A "face" feature, for example, is detected by looking for sub-features such as eyes, eyebrows, mouth, and nose. Their presence, plus their relative locations, define a face within an image containing it and any number of other objects of secondary importance. There are well-established practices for accomplishing this feature extraction, especially for face recognition, most of which belong to the class of machine learning algorithms.

3.3.1 Haar-like feature detection

Assume during face detection there is a small sub-window moving within the image window that contains some feature waiting for detection. For each sub-window location, a value representing the likelihood of it containing that is calculated. For Haar-like features, that value represents the difference between the sum of pixel brightnesses within two sub-regions of the sub-window. In the figure below are the Haar-like features used in object detection. The sub-regions are those represented by the two colors, white and black. They are often used to find linear, or otherwise isolated, features and are commonly applied in facial recognition.

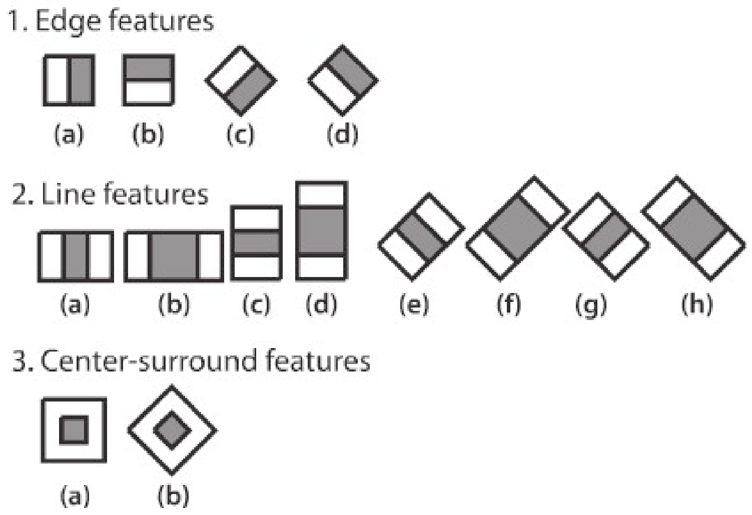


Figure 3.1: Haar-like features

During detection, the suite of Haar-like features are calculated over the entire image. Large values in certain areas tend to show features of interest, such as an eyebrow, for example. The likelihood of any single feature being present, however, is not enough to classify a region as being, for example, a “face” feature. The individual features are therefore combined in two stages, first using what are termed weak classifiers, and at a later stage strong classifiers built from several weak ones to detect a feature.

3.3.2 Weak classifier

In general, a weak classifier is a function that has only slightly higher correlation with some classification value than a completely random process. The mathematical form of a weak classifier applied in image processing is shown below.

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) < p\theta \\ 0 & \text{if } pf(x) \geq p\theta \end{cases} \tag{3.1}$$

A weak classifier $h(x,f,p,\theta)$ contains a sub-window image x, a feature f,inequality control factor p and threshold θ. p is a sign variable to control the direction of the inequality. In essence, the weak classifier assigns a class to an image region based on a single measure (in this case, a Haar-like feature) and a fixed threshold value.

The remaining problem is to find the set of thresholds for all weak classifiers that maximizes the likelihood an image region is correctly categorized. For this purpose, we employed a decision tree method known as a Classification and Regression Tree, or CART. `OpenCV` contains a "CvCARTHaarClassifier" structure in the weak classifier original code.

A simple CART example with three Haar-like features (f_1,f_2,f_3) is shown in Figure 3.2. The function is to judge if the input image is a face.

In the classifier, each path is an output of a decision. Each node in the weak classifier, called a stump, contains only one Haar-like feature so it only has one decision level. The goal for training the best classifiers is to find proper threshold for each stump while reducing the errors made by the classifier through all the samples.

To find a somewhat optimal 'weak classifier', the fundamental process is as follows:

(1) For each feature f, calculate all the feature values of the training samples and rank them. Scan the ranked feature values then calculate four values for each element in the list: t_1 - sum of the weights from all the face samples in the list; t_0 - sum of the weights from all the non-face samples in the list; s_1 - sum of the weights from all the

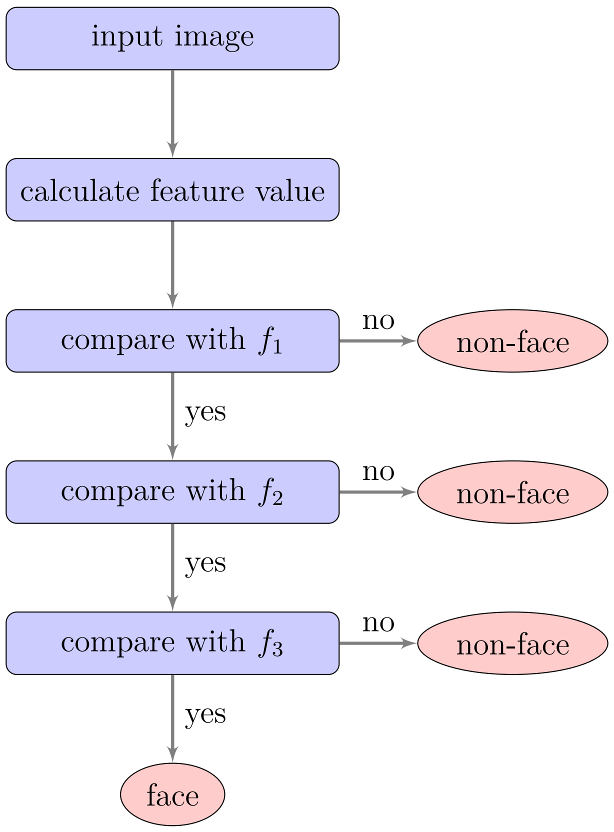


Figure 3.2: Flow diagram of weak classifier

face samples before this element; s_0 - sum of the weights from all the non-face samples before this element.

(2) Calculate the error of classification for each element.

$$r = min(s_1 + (t_0 - s_0), s_0 + (t_1 - s_1)) \tag{3.2}$$

(3) Find the minimum r in the list then set it to the threshold.

3.3.3 Strong classifier

Weak classifiers are built from single images, while strong classifiers use multiple images to combine outputs from weak classifiers and provide improved identification. The strong classifier needs T rounds of iteration before its final generation. A particular form of strong classifier, known as an Adaboost, can be described as follows:

(1) Given the training sample set S having N samples (images), X is the number containing a feature and Y those not. T is the maximum number of training iterations.

(2) The original (uniform) weight of samples, which stands for the original probability distribution of the training samples, is $\frac{1}{N}$.

(3) The first iteration trains N samples to get the first best "weak classifier".

(4) Increase the weight of misclassified samples in the last round.

(5) Add new samples together with misclassified samples and then start a new round of training.

(6) Continue to execute step (4) & (5) T times to generate T best "weak classifiers".

(7) Combine those T classifiers from step (6) to make a new strong classifier. Each weak classifier produces an output, hypothesis $h_t(x)$, for each sample in the training set. At each iteration t, a weak classifier is selected and assigned a coefficient α_t to minimize the summed training error of the boost classifier.

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \sum_{t=1}^{T} \alpha_t h_t(x) < \frac{1}{2} \sum_{t=1}^{T} \alpha_t \end{cases} \tag{3.3}$$

In other words, all the weak classifiers voted for the correct face image and the weighted sum of the misclassified rate was calculated according to the voted result. Finally the result of the weighted sum was compared with the average vote result to make the final decision.

3.3.4 Cascade classifiers

The process above is used to train a strong classifier for a single feature, such as a "face", but it is understood that faces can be composed of other recognizable features, such as eyes or noses, that can add strength to a "face" decision. There are techniques to combine outputs from multiple classifiers one of which is known as a cascade classifier. This technique is the key to improving the accuracy of object detection.

The main objective for cascade detection is to use a larger image as input and detect the feature through multi -area and -size detection of sub-windows. The function of multi-area detection is to divide the large images into several pieces and then perform classification on each piece. 20×20 pixel images are normally used to generate classifiers in these training processes. Once classification has been achieved using these rather small sub-regions, they are increased in size and the training process repeated until regions of a practical size for efficient computation are reached with acceptable classification rates.

The principle of the cascade classifiers is to rank several strong classifiers from simple to complex. Each strong classifier should have high detection accuracy after training, but the misclassified rate can also be higher than expected. For example, 99% of face images can be approved using output from a single string classifier, but 50% of non-face images also can be approved. Therefore, with a cascade of ten strong classifiers, the overall detection rate is $0.99^{10} \approx 90\%$ while the misclassified rate is $0.5^{10} \approx 0.1\%$.

The training of the cascade classifier is to make a balance between the detection rate and the misclassified rate. For example, K is the total level of the cascade classifier, D is the detection rate of the cascade classifier, F is the misclassified rate of the cascade classifier, d_i is the detection rate of i-th strong classifier, and f_i is the misclassified rate of i-th strong classifier. To train the cascade classifier to the assigned D and F, the detection and misclassified rate of each level should be trained to match the value of d and f.

$$d^K = D; \quad f^K = F \tag{3.4}$$

However, the strong classifiers trained using the AdaBoost algorithm have both low misclassification rate and low detection rate. Under normal circumstances, high detection rates can result in high misclassified rates due to the setting of a conservative threshold in strong classifiers. To increase the detection rate of the strong classifiers, the threshold needs to be reduced below what would it might be when used on its own. But, to reduce the misclassification rate of the strong classifiers, the threshold needs to be increased. It is a contradiction but the solution is to increase the number of strong classifiers increasing the detection rate at the same time the misclassification rate is reduced. Balancing these two needs consideration: one is balancing the numbers of the classifiers and the calculation time; another is balancing the detection rate and the misclassified rate in strong classifiers.

3.3.5 Integral image

Applying the above strategies in detecting facial features is a computationally intensive operation, but there are techniques available for speeding the process. Most have to do with strategies to speed feature computation. The "integral image" approach is one such technique.

The integral image can quickly calculate the sum of all the pixels by only one traversal, greatly improving the efficiency of obtaining feature values. The integral image is a matrix expression encapsulating summed pixel values. The generation of the integral image $ii(i, j)$ is the sum of all the pixels from the location (i, j), starting at the left top corner.

$$ii(i,j) = \sum_{k \le i, l \le j} f(k,l) \tag{3.5}$$

(1) $s(i,j)$ stands for accumulation of horizontal direction, initialize $s(i,-1) = 0$.

(2) $ii(i,j)$ stands for an integral image, initialize $ii(-1,i) = 0$.

(3) With progressive scanning, recursive computing the accumulation $s(i,j)$ for each pixel (i,j) and the value $ii(i,j)$ for the corresponding integral image.

$$s(i,j) = s(i,j-1) + f(i,j) \tag{3.6}$$

$$ii(i,j) = ii(i-1,j) + s(i,j) \tag{3.7}$$

(4) Fully scan the whole image. When (i,j) reaches the right bottom corner, the integral image is generated.

The pixel accumulation of any matrix region can be calculated through a simple operation. If α,β,γ,δ are the vertexes of rectangle D (Fig 3.3), the sum of the pixels in D can be presented as the following Equation 3.8, and the Haar-like feature value is just the difference between two matrix sums.

$$D_{sum} = ii(\alpha) + ii(\beta) + ii(\gamma) + ii(\delta) \tag{3.8}$$

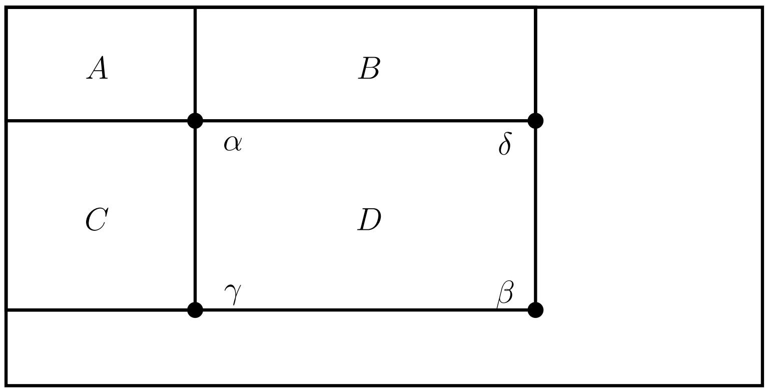


Figure 3.3: Integral image

3.3.6 Face and eye detection implementation

The first implementation of the tracking system was for face detection since all the more interesting features, including eyes and pupils, were on the face and delineating its extent can speed further detection. The Haar-like feature-based detection approach outlined above was used in this step, and also in the eye detection outlined below. The actual implementation of the algorithm was found in the OpenCV function

```
faceCascade=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')
```

To keep the process as simple as possible, it was decided that color information was not necessary and all computations were based on a gray scale image. Converting to gray scale also helped facilitate later processing for more refined feature detection.

The full source of the eye detection function can be found in the appendix. It returns a sequence of faces detected, from which the x and y coordinates, and the height and width of the face region can be extracted. The algorithm permits setting a minimum possible object size to ignore smaller objects. For the sake of verification, a rectangle is then drawn around the face.

Eye detection was done in a manner similar to face detection but with a different classifier. The classifiers used for the eye detection were found in the OpenCV function

```
haarcascade\_eye.xml \ .
```

The method follows the same basic steps as the face detection: set the detection region within the face rectangle, then apply the Haar-like feature detection method. Figure 3.4shows an example screen shot of the results of face and eye detection.

In this study, data from one eye was felt to adequately represent the directional information of both, but (in most cases) two were returned. Because the order in which they were returned was somewhat random, a sorting algorithm was developed to always use the same eye (left) for directional calculations. It was based on relative x positions within the original image.

3.4 Pupil tracking design

Gaze direction calculation was based on pupil location within the eye region, so the next step in the process was to find the pupil center. Since the pupil tended to be much darker

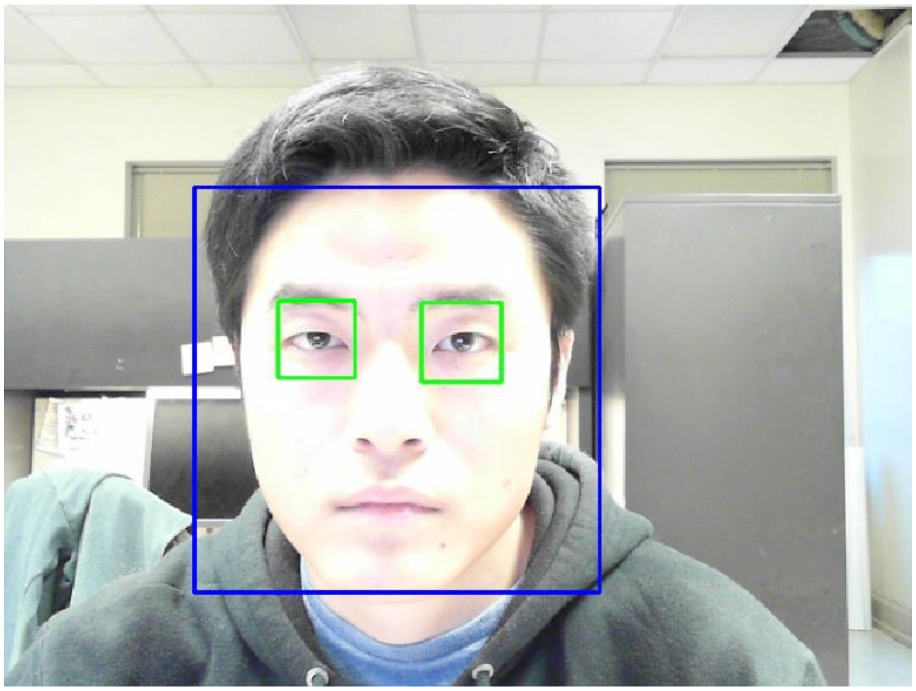


Figure 3.4: Face and eyes detected within rectangles

than the surrounding portions of the eye region, its identification began with conversion to a binary form. Following the binarization, morphological transformations, including dilation and erosion, were applied to help regularize the geometry of the candidate pupil regions. However, noise was not totally removed through morphological transformations, and they often could generate other types of noise through their application. Thus, an area filter was executed to decrease the remaining noise and correct errors in the image. Finally, the minimum enclosing circle around the identified pupil was drawn and the center of the circle was calculated. This value was used in establishing gaze direction.

3.4.1 Binarization

Binary conversion is a basic technology in image processing. In most cases, a gray scaled image is required as input. According to their grayscale value, the binarization is to separate the pixels into two brightness classes by setting a threshold. The pixels with grayscale value lower than the threshold are treated as background while the others are regarded as target pixels. Thus, the grayscale image $f(x,y)$ is converted to the binary image $g(x,y)$ which contains only two values.

$$g(x,y) = \begin{cases} Z_1 & f(x,y) \geq Threshold \\ Z_0 & f(x,y) < Threshold \end{cases} \tag{3.9}$$

The main idea of selecting a proper threshold is to keep the extent of the target image features as close to their original size as possible, and at the same time minimize misclassification of uninteresting areas. Three methods of binarization were investigated for this study: global threshold binarization, the local threshold binarization, and dynamic threshold binarization.

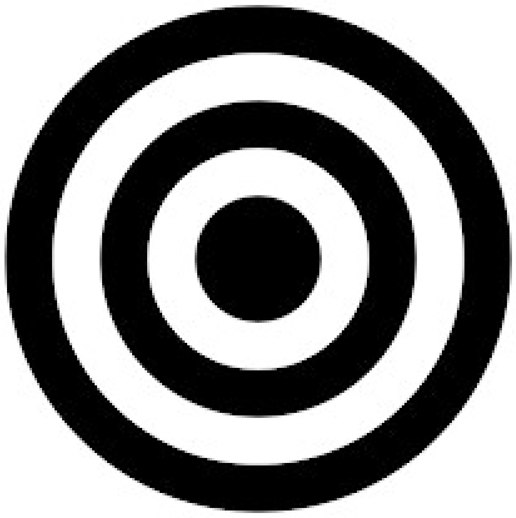
Global threshold binarization is to set the threshold based on experience, or using knowledge of the distribution of grayscale values typically encountered.

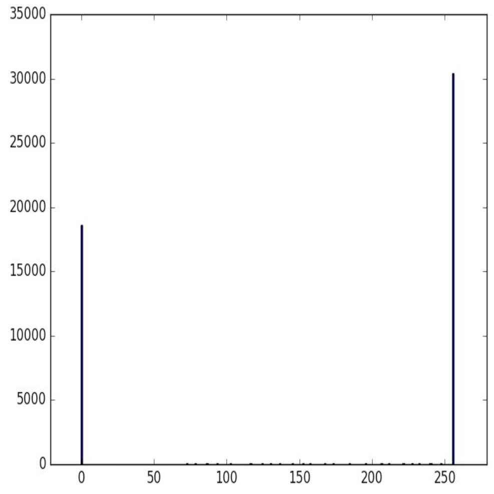
(1) Manually set the threshold: In this case, the threshold is firstly set according to experience and applied to each pixel in the grayscale image using the equation above.

(2) Set the threshold from the distribution of the whole grayscale values: Utilizing histogram distribution of the grayscale values came from original image to describe the overall grayscale values. Assume grayscale value f is an integer from 0 to 255, 0 being black while 255 stands for white. $p(f_k)$ is the probability the grayscale value equals to k. n_k indicates the numbers of pixels with grayscale value that equals to k, while n is the number of the total pixels. Then,

$$p(f_k) = \frac{n_k}{n} \quad (0 \leq f_k \leq 255) \tag{3.10}$$

In the histogram of the grayscale values, f_k is the horizontal axis and $p(f_k)$ is the vertical axis. Take a target image (Fig 3.5a) for example. The original target image only consists of two colors. Therefore there are two peaks in the histogram(Fig 3.5b). The best threshold


(a) Target image



(b) Histogram of target image

Figure 3.5: A target image and its histogram

value should equal to the value around the lowest point in the valley between them. The deeper the valley, the more likely the result of binary converting will be close to what is desired. In most cases, however, it is hard to define an obvious difference between two peaks, so this method is of limited value in practical applications.

Using the global threshold binarization method, the only consideration is to get the average grayscale value without taking the difference of each pixel into account. The positive sides are the fast execution speed, and its simplicity. It is of limited value in the presence of asymmetrical light sources, and it also tends to introduce spotty noise into the image.

For an image with clear features and background, the global threshold binarization returns good results. But for asymmetrical background or large gradients in the grayscale values, the global threshold binarization is no longer suitable. Other methods using the grayscale values $f(x, y)$ for pixel (x, y) based on those in its immediate neighborhood are called local threshold binarization. The local thresholding process involves dividing the whole region into several sub-images, then determining the relative threshold based on each sub-window through the same method as in global threshold binarization.

This method is widely applied on those images with poor lighting qualities and serious detection errors when calculated using a global threshold. Its disadvantages are the slow processing speed as well as that the continuity of the image is not guaranteed. Like the global threshold, it is also prone to introduce noise.

Dynamic, or adaptive, binarization means calculating the threshold for each pixel in real time based on gray values within its own surrounding region. Dynamic threshold binarization is able to achieve reasonable results on images with very poor qualities, even those with histograms consisting of only one peak. The processing time for this method, however, can be very long.

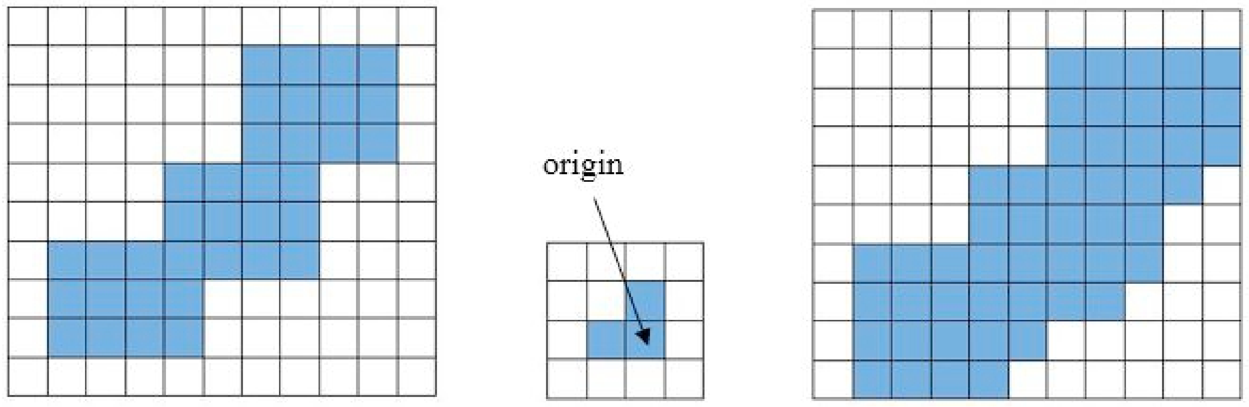
Method	**Applicable Condition**	**Disadvantage**
Global	Images with distinct two peaks in histogram distribution	Weak resistance against various illumination conditions and noises
Local	Images with poor consistency of grayscale values	Slow processing speed; Additional noise
Dynamic	Terrible quality images	Slowest processing speed; Partial distortion

Table 3.1: Comparison of the binarization methods

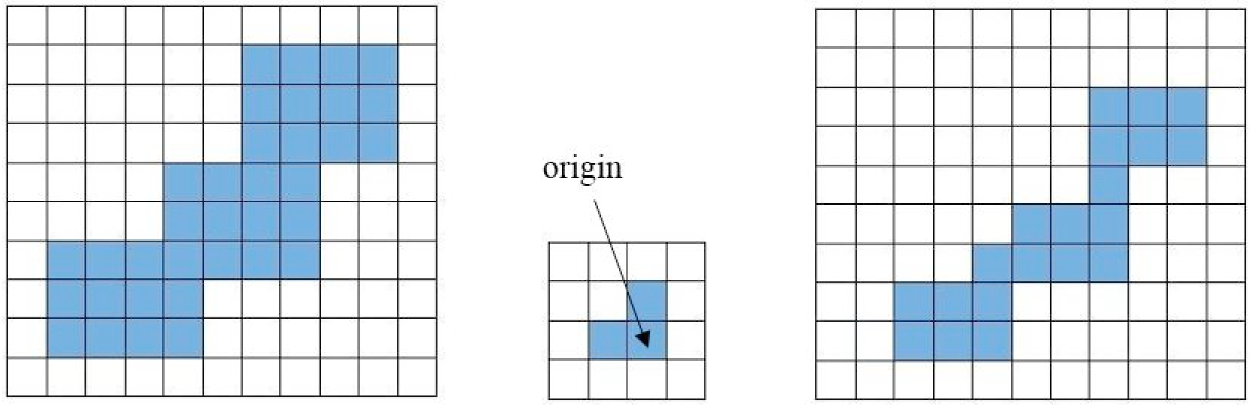
3.4.2 Morphological transformations

Dilation and erosion are two morphological transformations that are set theoretic operations on binary images. In general, they add/take away regions within an image relative to a geometric ‘probe’ known as a structuring element. Both are commonly applied in reducing noise and highlighting features in image processing, especially in feature detection.

In Figure 3.6a, the left binary image (blue pixels) is the image waiting for the dilation process. In the middle is the structure element. The process of dilation is to match the origin of structure element with each pixel p_i in binary image. For any pixel in structure element is within the range of binary image, then this pixel p_i turns into blue. The right image is



(a) Dilation



(b) Erosion

Figure 3.6: Two types of morphological transformation

the result after dilation and illustrates the region of interest is grown. Figure 3.6b shows the opposite operation, known as erosion, which removes areas relative to the structuring element.

The dilation and erosion process concentrate not only external contours but also the internal holes in the pattern. To get better results, combinations of dilation and erosion that called open operation and close operation are used in image processing. An open operation is to do dilation at first and then execute erosion while a close operation is done with the inverse processing order.

3.4.3 Histogram equalization

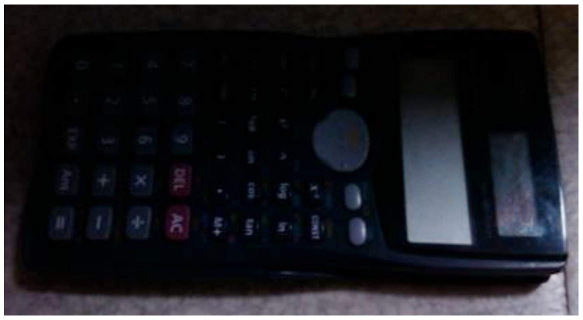
The histogram of an image plots the relative frequency of occurrence of a specific gray value. For a partially dark grayscale image, it is difficult to distinguish features, especially when undergoing the pupil contour tracking process. Histogram equalization is a compensation process for avoiding the effects of partial darkness in the image. The principle of the method is to stretch the histogram along the X axis without changing the pixel value respectively.

For example, the original image of a calculator with partial darkness problem is shown as below (Fig 3.7a). The left top part of keypad was intended to be covered under shadow so that it was not easily recognized. After histogram equalization, the image (Fig 3.7b) became brighter and all the numbers on the keypad as well as the contours of all the buttons were clearly visible. From two histograms (Fig 3.7c & 3.7d), the previous plot was stretched so each pixel value was covered by more than one pixel.

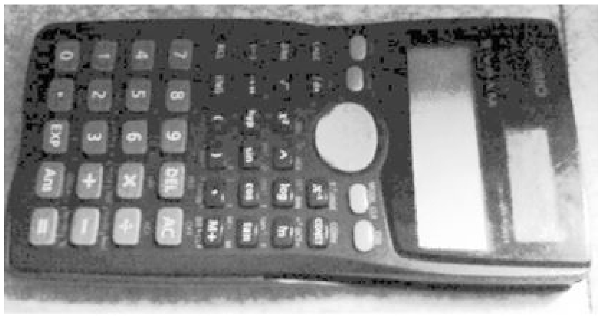
3.4.4 Loop steps

Given all the methods outlined above, the following process was used to extract features down to the pupil in facial images. The flow diagram of the pupil tracker is shown as Figure 3.8.

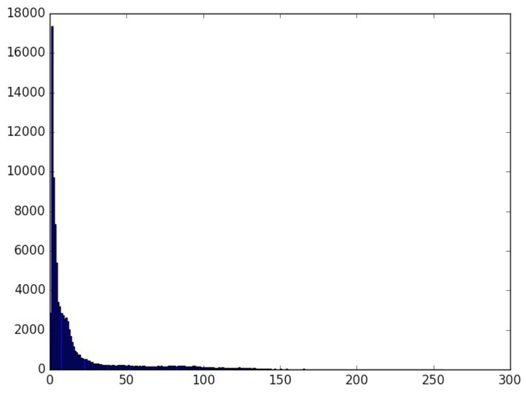
A grayscale image of the operator's eye (after extracting it from the original scene of the operator's face) was the input to the loop function. Because of fickle illumination conditions in the skidder cab, it was crucial to execute histogram equalization on the input image before taking further steps. After binary conversion, the image consisted of regions from the pupil, pieces of eyelash, corners of the eye, and noise. A sequence of morphological transformations was then used to reduce the number of candidate features as much as possible to extract the pupil. At the same time, holes caused by reflected light were filled in the pupil region. Finally, the area filter was applied, a minimum enclosing circle calculated, and it's center found.



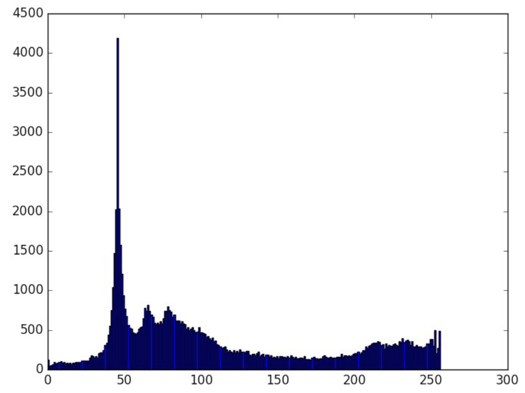
(a) Original image



(b) Processed image



(c) Original histogram



(d) Equalized histogram

Figure 3.7: Comparison of image and histogram before and after equalization

3.4.5 Main program test

The live pupil tracking algorithm was tested as follows (Fig **??**). The captured image was firstly converted to grayscale (Fig 3.9b), then the sub-window containing the eye was cropped. After the binary conversion and morphological transformations (Fig 3.9c), three elements consisting of pupil, eye corner, and eyelash were left. The eye corner and the eyelash were not always there after every conversion, and there might be some other small noise regions left in the binary image. Hence, the area filter then was applied to remove everything except the pupil. For the test subject's pupil, the normal area was usually around 280 square pixels, so the range of the area filter was set from 200 to 300, A relatively small lower size limit was used because the pupil could be partially obscured when attention was focused sharply to the right or left. A clear pupil image was typically generated after this

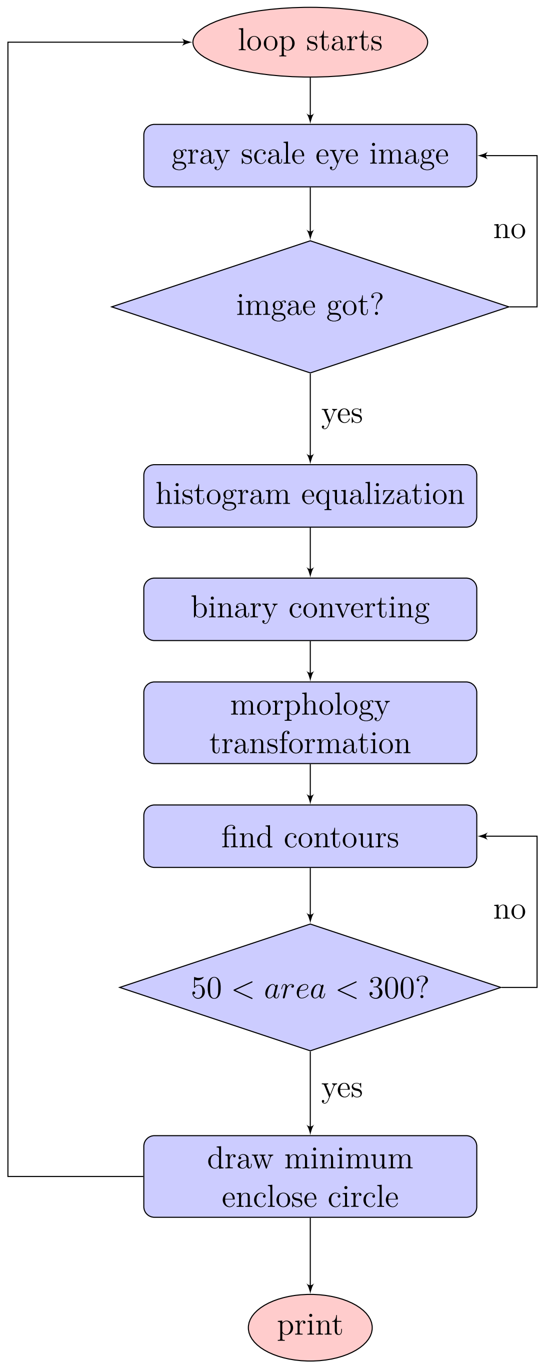
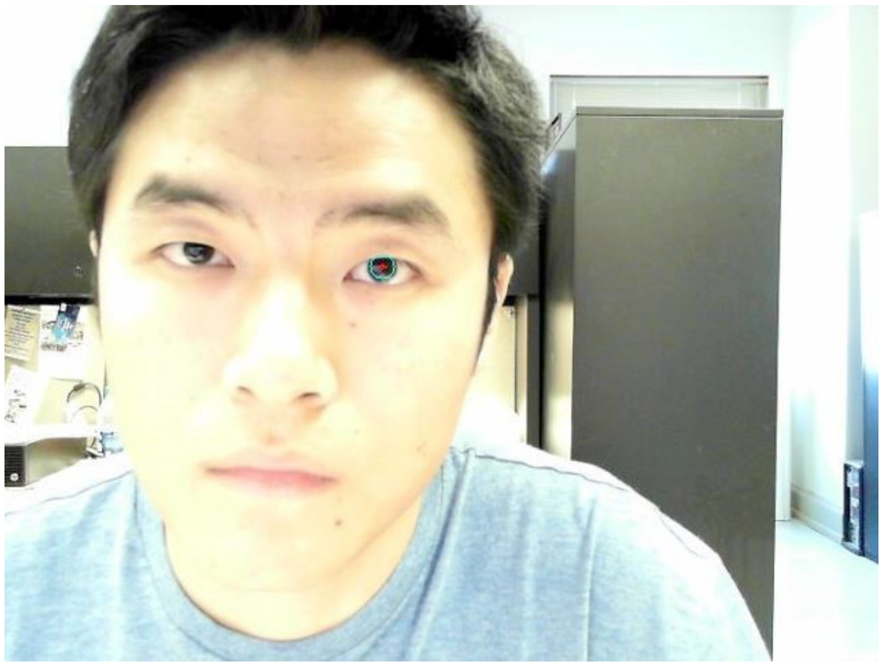
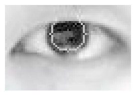


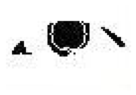
Figure 3.8: Flow diagram of the pupil tracker

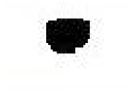
process, and an example result is shown in Figure 3.9c. The minimum enclosing circle was calculated to determine the center location of the pupil.



(a)







(b) (c) (d)

Figure 3.9: A detected pupil image and sub-images in progress: (a) pupil detection in one frame; (b) grayscale; (c) binarization & morphological transformations; (d) pupil image

3.4.6 Algorithm performance

The parameters set in binary conversion, morphological transformations, and filter are the key to successfully applying the algorithm. Although approximate values for thresholds and other parameters can be set for generic situations, tuning to specific conditions improves performance. For the test subject's eyes and face, different directions of pupil tracking were tested (Fig 3.10 & 3.11 & 3.12 & 3.13 & 3.14) to evaluate how well it worked. It performed adequately under ambient light conditions, although if the subject wore glasses it tended to cause problems.

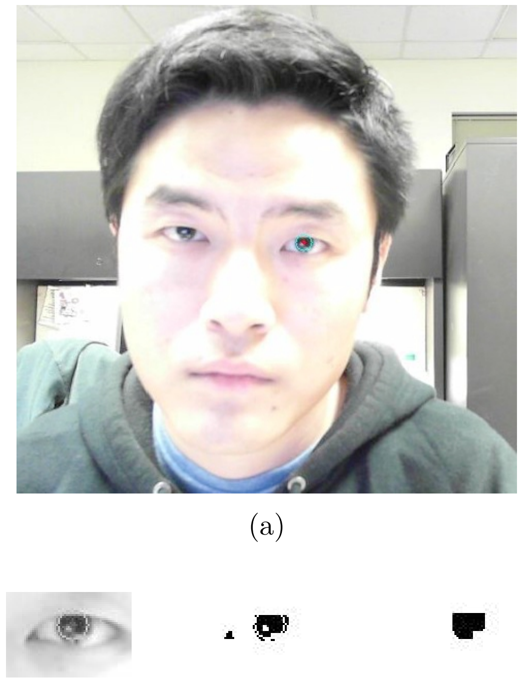
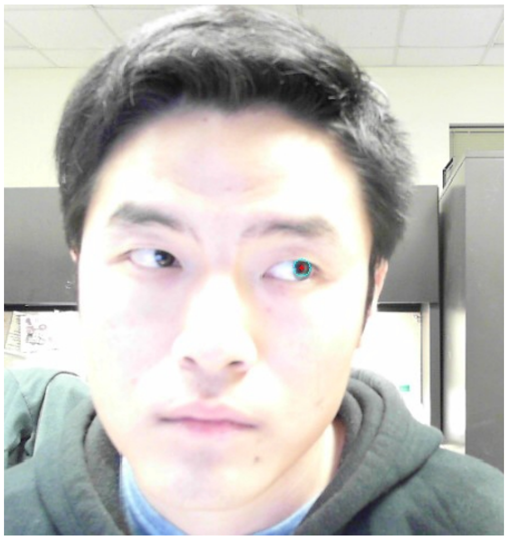
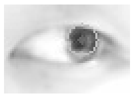
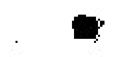


Figure 3.10: Pupil detection with front looking

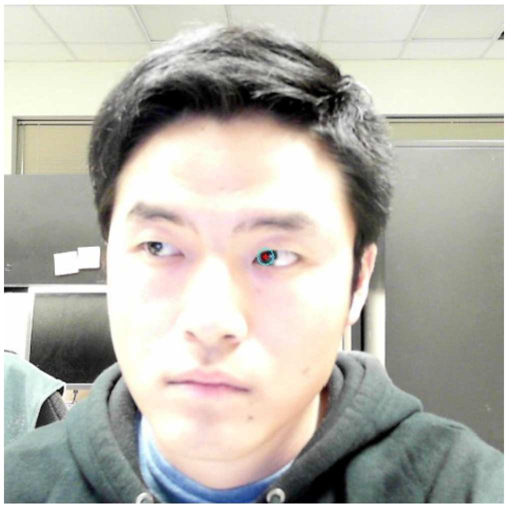


(a)

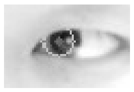
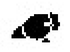
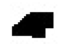



(b) (c) (d)

Figure 3.11: Pupil detection with left looking

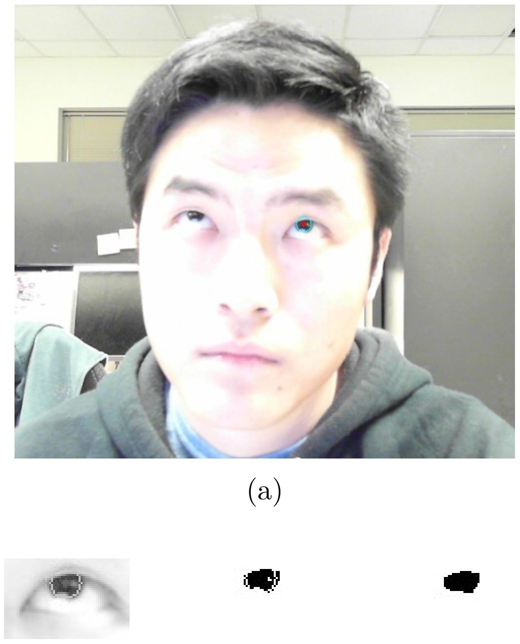


(a)

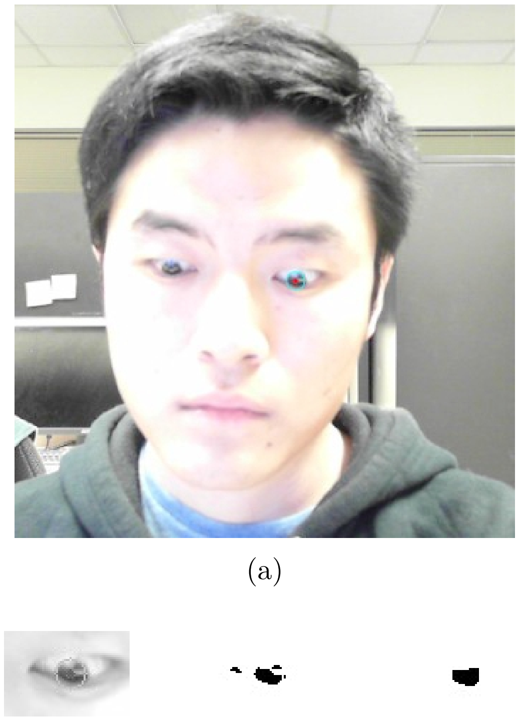
(b) (c) (d)

Figure 3.12: Pupil detection with right looking


(a)

(b) (c) (d)

Figure 3.13: Pupil detection with up looking


(a)

(b) (c) (d)

Figure 3.14: Pupil detection with down looking

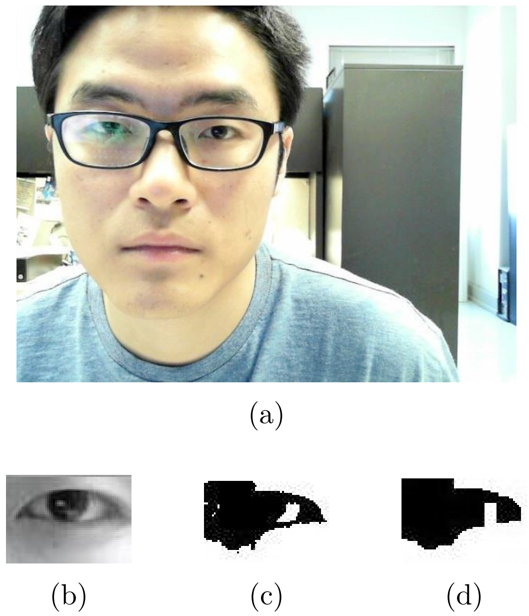


Figure 3.15: Pupil detection failure condition

3.5 Head motion estimation system design

The primary objective in this section was to build a head tracking system representing the rotation angles and transition distances in three dimensions. In this study, estimating head pose/position was accomplished using a chessboard mounted on the back of a hard hat. A camera set right the subject was used to estimate pitch, yaw, and roll based on images of the chessboard. The algorithm returned values of X, Y, and Z coordinates representing the origin of the chessboard, which could be translated easily to an eye position. X and Y represented the distance away from the center of the video window while Z stood for the distance away from the camera.

3.5.1 Camera calibration

No matter how good a camera is, it always creates some distortion in any image it produces. There are two major distortions in any image, radial and tangential. Straight lines will appear curved in the image because of radial distortion, and the effect is more noticeable in areas away from the center of the image. The chessboard pose estimation program needs extremely straight lines and square corners to accurately estimate the positions and angles, therefore it is necessary to calibrate the camera to remove any distortion it might create.

The radial distortion is represented as follows:

$$x_{distortion} = x \times (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.11}$$

$$y_{distortion} = y \times (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.12}$$

Tangential distortion results from the camera lens not being perfectly parallel to the image plane. Its effect is to cause some areas in the image to appear nearer than expected. The tangential distortion is represented as follows:

$$x_{distortion} = x + [2p_1xy + p_2(r^2 + 2x^2)] \tag{3.13}$$

$$y_{distortion} = y + [2p_2xy + p_1(r^2 + 2x^2)] \tag{3.14}$$

Therefore, the distortion coefficients can be given by five parameters:

$$Distortion\ coefficients = (k_1\ k_2\ p_1\ p_2\ k_3) \tag{3.15}$$

To make the necessary calculations for reconstructing angles from the checkerboard images, camera matrices, including intrinsic and extrinsic parameters, are needed. Extrinsic parameters translate 3D coordinates to 3D camera coordinates and are represented using rotation and transition vectors R and T. These parameters are typically represented using a so-called camera matrix, K, and are specific to a given camera. $f_(x)$and f_y present focal length in terms of pixels. c_x and c_y present the principal point which should be ideally in the center of the image.

$$camera\ matrix : K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

Functions in `OpenCV` were available for estimating the above parameters from test images. For this study, ten sample chessboard images were taken under the same illumination conditions and the parameters were calculated.

3.5.2 Pose estimation

With the camera matrix and distortion coefficients estimated from the previous calibration, the `OpenCV` function `cv2.findChessboardCorners` was used to find the corners and axis points on the chessboard. The program was set to search for a 5×4 grid because the grid should be at least 3×3. Also in practice, 5×4 grid can be detected more quickly than larger values and the origin estimation tended to be more stable. For each image loaded, the program searched for a 5×4 grid, the applied the `OpenCV` function `cv2.solvePnPRansac`, which implemented the RANSAC scheme, to calculate rotation and transition vectors, as in the following.

```
rvec,tvec,inliers=cv2.solvePnPRansac(objectPoints,imagePoints,cameraMatrix,DistCoeff)
```

revc stands for rotation vector and *tvec* stands for transition vector. *objectPoints* is the array of object points in the object coordinate space. *imagePoints* is the array of corresponding image points which is the sub-corner array found previously.

A rotation vector is a convenient and compact representation of a rotation matrix, but is unsuitable for calculations of head pitch, roll, and yaw, which requires Euler angles. The `OpenCV` function `cv2.Rodrigues` was used to make the conversion. For the input rotation matrix *src* and the output rotation vector *dst*, the calculation was made as in the following.

```
dst, jacobian = cv2.Rodrigues(src)
```

3.5.3 Loop steps

The flow diagram for the head pose tracking algorithm is shown in Figure 3.16.

The first step was to get a single frame from the camera. Once the frame was retrieved, the frame was converted to grayscale. The next step was to find chessboard corners and store those corners into an array. Then the rotation and transition of the head were calculated.

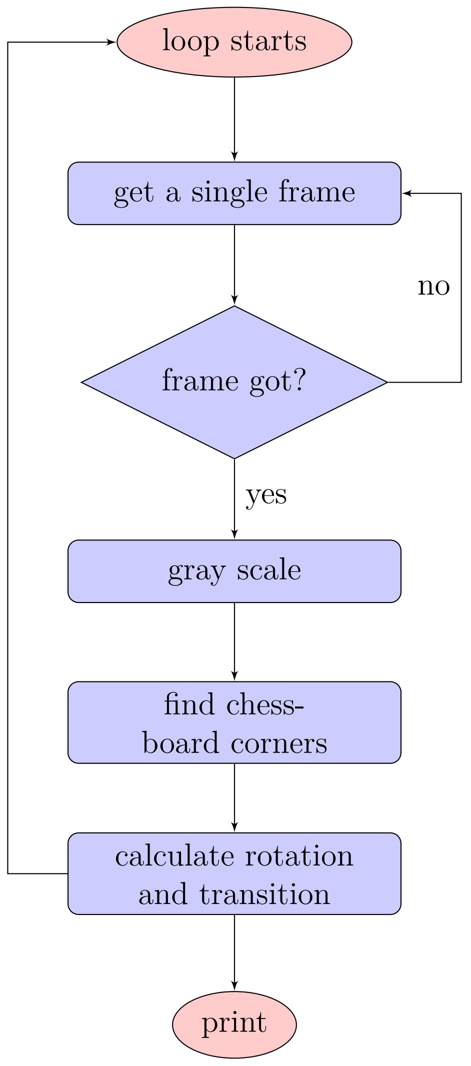
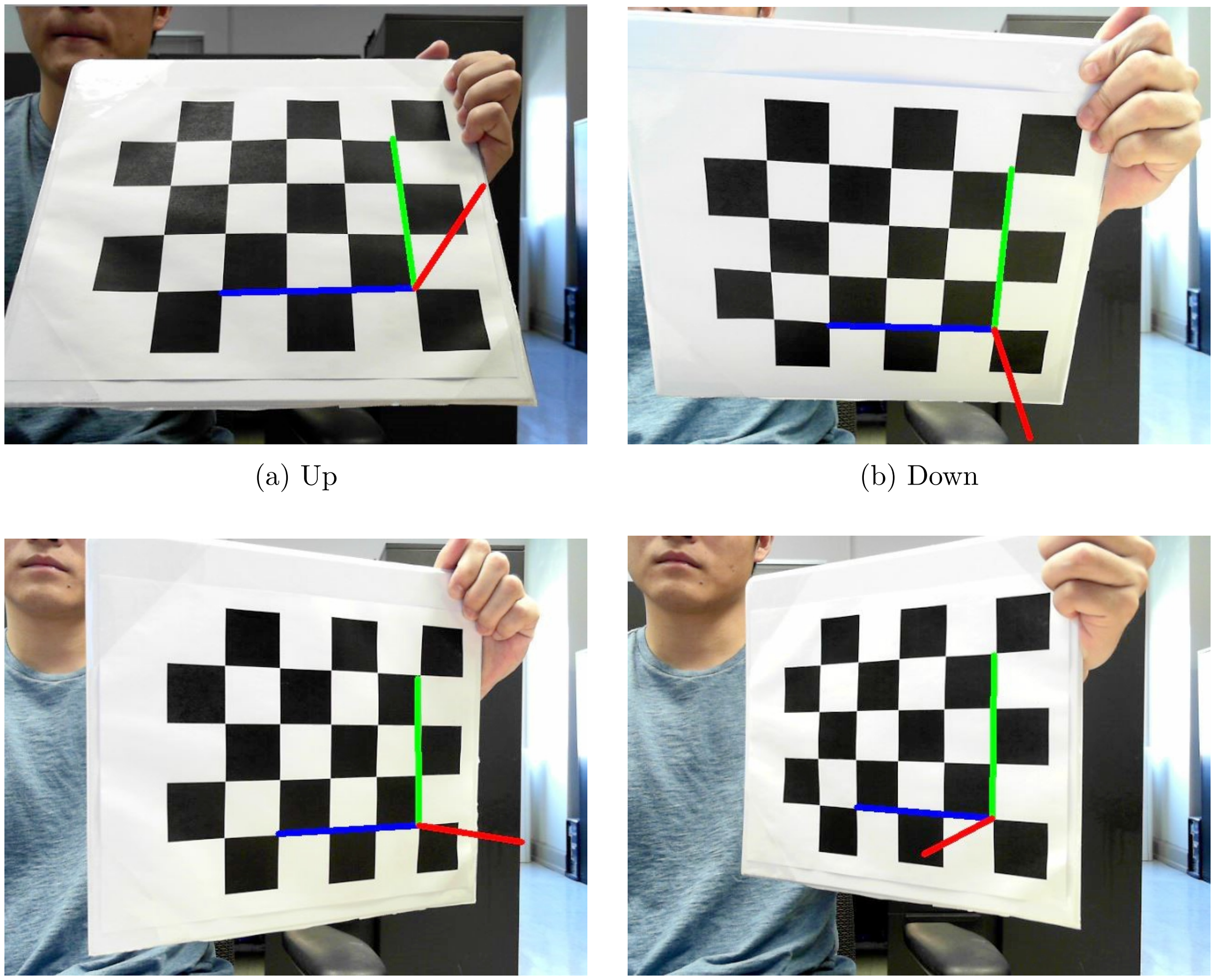


Figure 3.16: Flow diagram of head pose estimation

3.5.4 Head pose algorithm test

The program implementing the algorithm outlined above was used to estimate origin and pitch/roll/yaw for various chessboard orientations. All the data including camera matrix, distortion coefficients, head pitch angle, head yaw angle, head roll angle, head X-axis transition, head Y-axis transition, head Z-axis transition were recorded. Based on those data, a 3D representation of the unit axes at the origin point were generated. The 3D reconstruction test images were shown in figure 3.17. The results were verified visually. It was felt the algorithm was sufficiently accurate and robust to be used in the field tests.


(a) Up (b) Down

(c) Left (d) Right

Figure 3.17: Pose estimation of four directions

3.5.5 Calibration

A calibration was necessary to translate the pitch/roll/yaw and origin location as output by the program into actual positions in space, so the following calibration procedure was done. Two metal brackets were used to support the web camera and the chessboard and a tape measure was used to measure the distance between the web camera and the chessboard. A ruler was placed along the horizontal axis of the chessboard to measure the distance along the X axis. The bracket which held the web camera was fixed. The other bracket holding the chessboard was moved parallel to the ruler and a series of images taken. The experimental setup is shown in 3.18.

$\Delta L/mm$	ΔX	ΔZ	$\Delta D/mm$
50	2.44	10	24
50	2.64	20	47
50	2.52	30	70
100	4.93	30	69
100	5.00	20	45
100	4.87	10	23

Table 3.2: Results of calibration

Table 3.2 indicated the results of the calibration. ΔL was the distance moved along the X axis. Range of X was the value recorded by the program while moving the chessboard. ΔX was the absolute value of X. ΔZ was the value recorded by the program showing the distance from the vertical plane of web camera to the X-Y plane of the chessboard. Regardless of the values of x, y, as long as the chessboard stayed in the same X-Y plane, the value of Z did not change. ΔD was the distance measured between chessboard and web camera.

$$20\Delta X = \Delta L(mm); \quad 2.3\Delta Z = \Delta D(mm) \tag{3.17}$$

3.5.6 Accuracy

The chessboard 3D reconstruction program has a wide detection range under good light conditions. The limitation of pitch angle is from -70 degree to +70 degrees. The limitation of yaw angle is from -70 degree to +70 degree. There is no limitation of roll angle as long as the chessboard is detected, but this becomes less certain as the roll angle gets near horizontal. The limitation of X and Y axis transition is according to the image window size. The limitation of Z axis transition is based on the area of chessboard in the image window. Usually, when the area is under 5% of the area of the image window, the 3D reconstruction estimation doesn't work. While under poor light conditions, like a dark room or a shadow on the chessboard, the program cannot distinguish the boundary of the squares. The failure rates of estimation were summarized in figure 3.19.

The values calculatrf for the program matched well with the real measurements including angles and transitions. However, the main potential of inaccuracy so far is the failure of detection. If no chessboard or corners were detected, a null list was printed indicating the failure of the 3D reconstruction estimation. In the real field test, if the chessboard moved too fast in the video window, it may also cause the failure of detection. Overall, the detection rate in the field test was over 80% in tests made on over ten thousand images.

3.6 Conclusions

In this chapter, the gaze and head motion tracking system was designed for the sake of estimating the overall direction. The tracking system was able to detect the face and then recognize the eyes within the face. With the gray scaled input image, Haar-like feature detection method was applied to process both face and eye detection. Afterwards, the eye image was processed through histogram equalization, binary conversion, morphological transformations, and noise filtering. Then the clear pupil image was identified from the face image. Also, the tracking system was capable of determining the orientation of the head motion including rotation and transition. The principle was to reconstruct the 3D pose estimation on a chessboard. The camera calibration which would be used in the following process was firstly calculated. Then, the corners on the chessboard were detected and 3D coordinate axis was generated to indicate the pose direction. All the angles of rotation and distance of transition were recorded including warning messages expressing failure of detection.

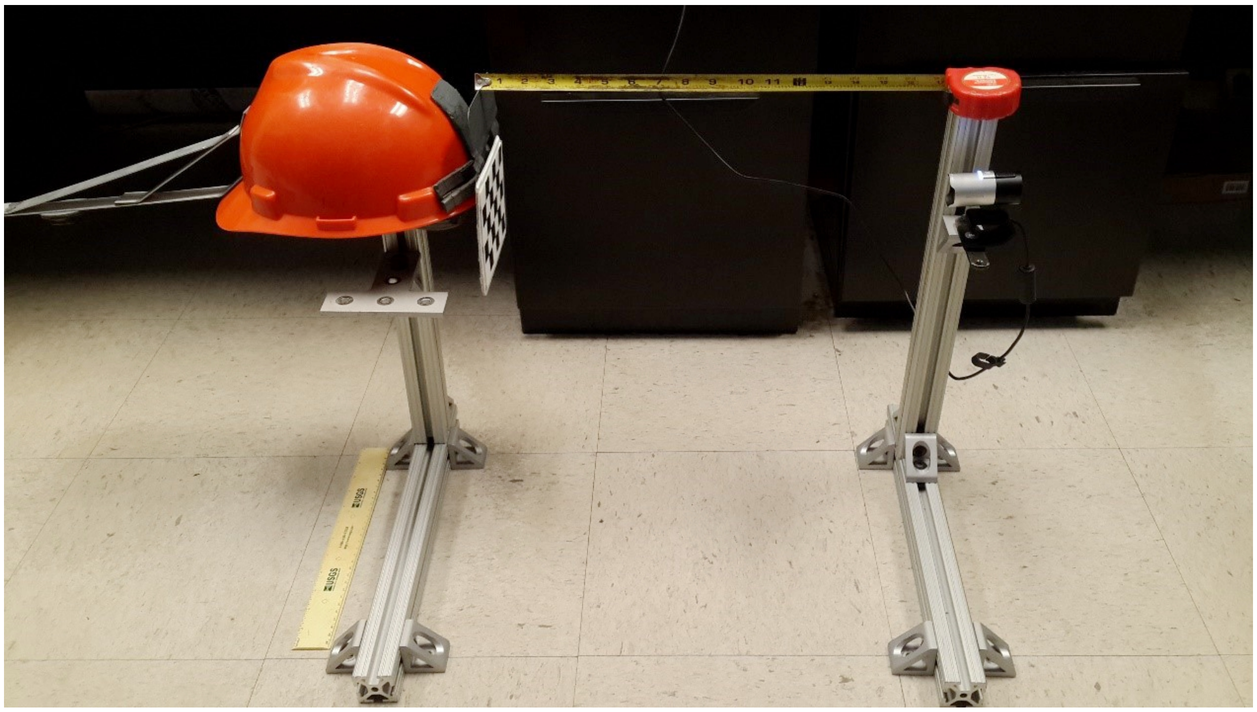
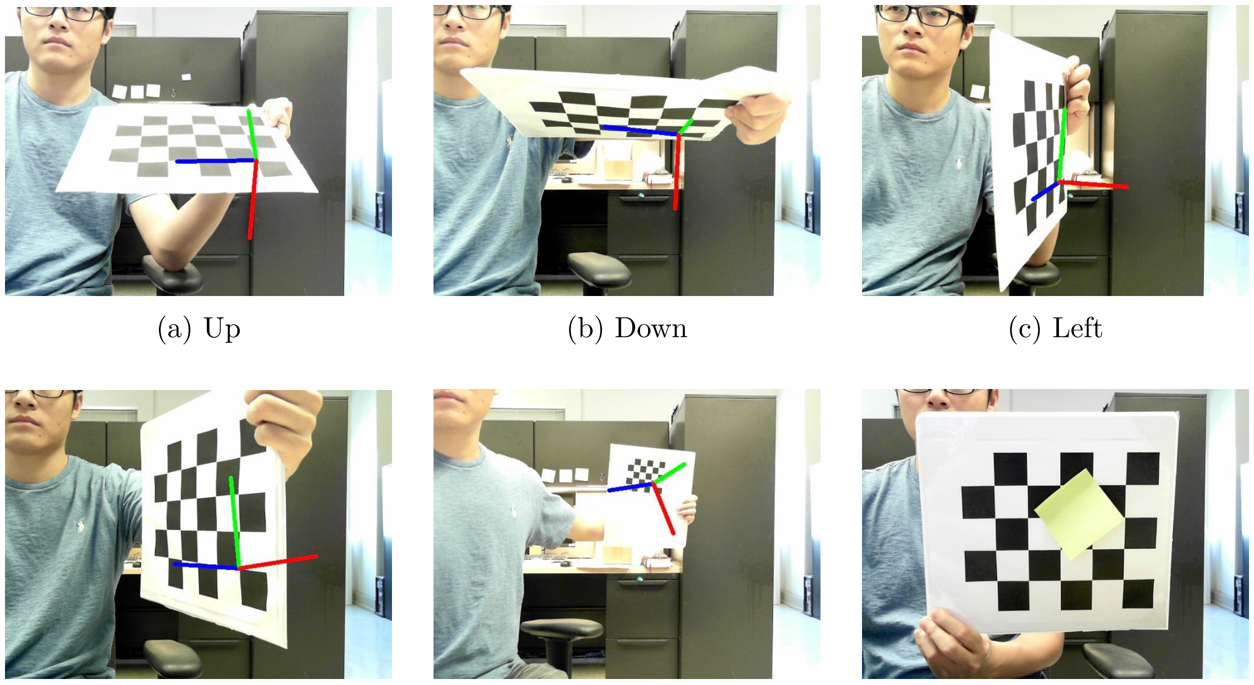


Figure 3.18: Calibration of pose estimation



(a) Up

(b) Down

(c) Left

(d) Right

(e) Away from camera

(f) Shadow

Figure 3.19: Limitation cases of pose estimation

Chapter 4
Gaze estimation with head motion

To get a further understanding of what the skidder operators were looking at and which areas interested them more, it was essential to generate 3D gaze vector and estimate the gaze points on a specific surface. With all the angles and coordinates got in the previous chapter, a 3D gaze vector including the head motion was calculated. Then the meat map indicting the density of the gaze points was plotted in order to examine the interested zones that the operator watched more often during timber harvesting.

4.1 General algorithm

4.1.1 Definition of parameters

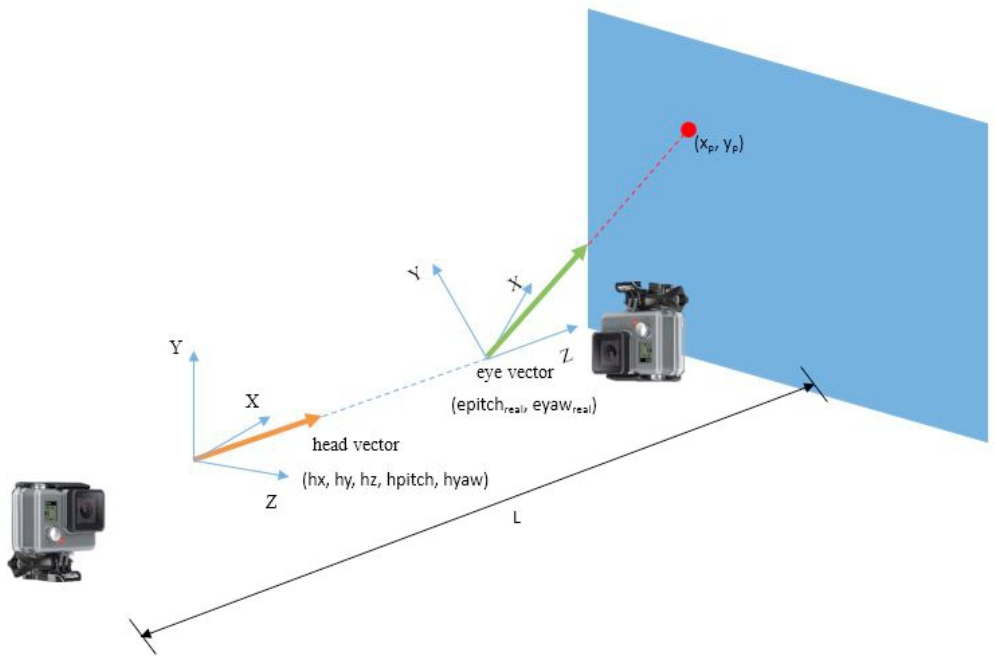


Figure 4.1: 3D vectors in the skidder cab

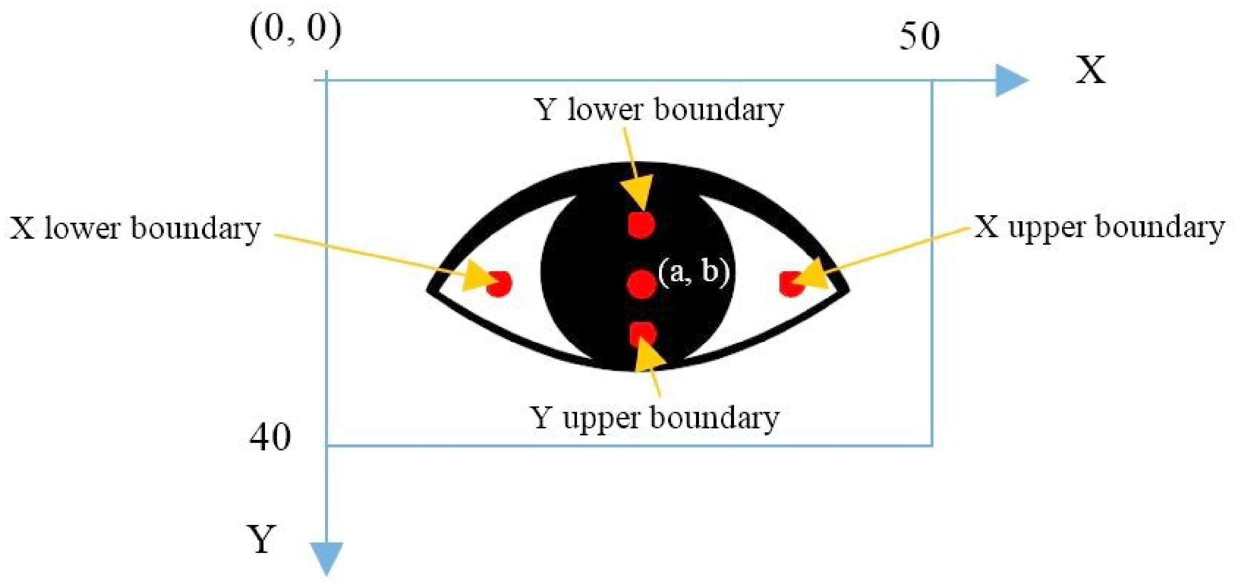


Figure 4.2: A cropped window of eye

A sample of the image after pupil tracking is shown as follows. There should be only a clear pupil and some small noises after processing but in order to have a specific statement of the definitions, the contours of the eyelash still remain in the image. Just to be clear, the origin of the window is always the top left point. The positive direction of X axis is to the right side while the positive direction of Y axis is to the down side. The size of the cropped image window is 50×40 and 50 as well as 40 are the numbers of pixels. The overall eye image is in the center of the window (Fig 4.2).

Descriptions for parameters may be used in the following section:

(1) *eyex*, *eyey*: coordinates printed after pupil tracking showing the location on the cropped eye window.

(a, b) is the estimated center of the pupil while looking forward, through averaging of upper and lower boundary of X and Y coordinates. The reason that the average of all the X values is not considered as a is that the average is not right in the center of the eye while someone tends to look at one side.

The average horizontal moving distance of the pupil from left to right is $15mm$. The average interval length between lower and upper boundary of X is equals to 30 pixels. So, $1eyex = 1eyey = 0.4mm$. The average radius of the eyeball is $12mm$.

(2) *hpitch*, *hyaw*: pitch angle and yaw angle of the head. Positive pitch angle is defined when head is raised while negative pitch angle related to lowering the head.

hx, hy, hz: hx and hy present the distance away from the center of the frame window. However, hz is the distance between camera and chessboard. As mentioned the previous chapter, $1hx = 1hy = 20mm$ and $1hz = 23mm$.

(3) L: length from front windshield of the skidder to the camera installed on the back window, $L = 1500mm$.

(4) Average diameter of the human head is $180mm$.

4.1.2 Gaze vectors

In this study, the vectors of the eyeball and the head are all 3D vectors. The coordinate systems of eyeball and head are shown below. The positive directions of the axis and rotation are then defined. When you are facing forward, in the head coordinate system, the positive direction of X axis is towards the left. The positive direction of Y axis is upward while the positive Z axis is forward. The positive pitch direction is the direction when you raise your head. However, the positive yaw direction is the direction when you rotate you head towards to the right. The X, Y and Z axis are towards the same directions of the head's. When you look up and look right, the positive pitch and yaw angles of eye are generated.

To determine the gaze of the skidder driver along with the head motion, the gaze vector is required to be calculated for the sake of generating the heat map in the later section. However, the gaze vector which is the result of the two vectors is not simply adding the two vectors together. Actually, the coordinate system of the eye vector is relative to the coordinate system of the head.

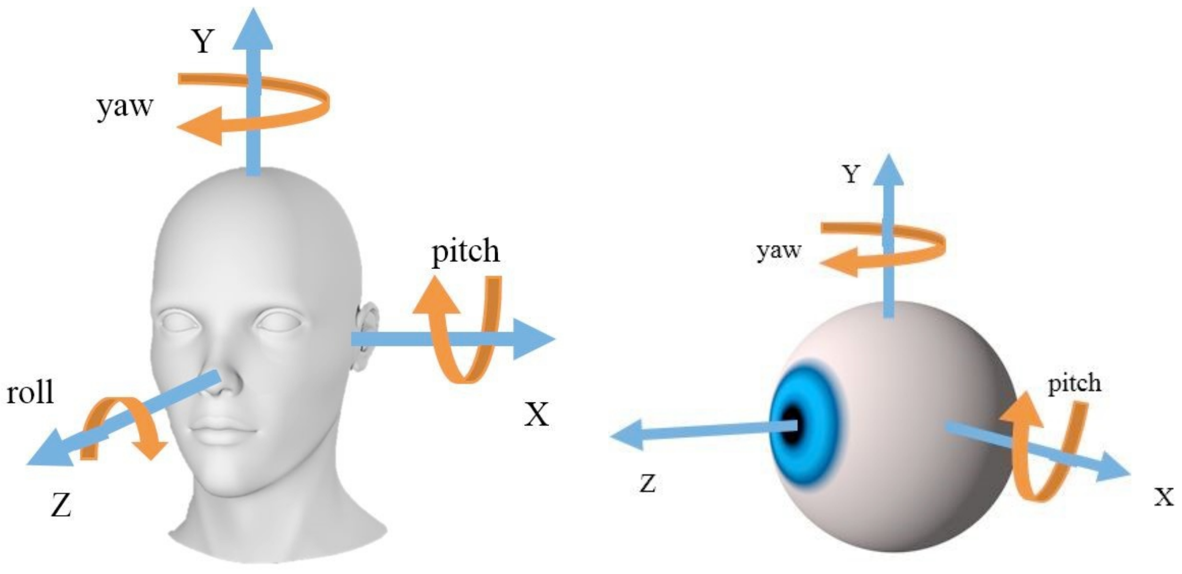
The unit vector $[ex, ey, ez]$ of the eye was firstly calculated. ex and ez are the coordinates directly representing on the screen while ez is the distance from the center of the eyeball to the coating of the eyeball. During the eyeball movement (Fig 4.4), ez can be treated as an constant which equals to the radius of the eyeball, because the difference between each ez and radius of the eyeball can be igored.

$$ex = \frac{(eyex - a) \times 0.4}{12} \tag{4.1}$$

$$ey = \frac{(b - eyey) \times 0.4}{12} \tag{4.2}$$

$$ez = \frac{12}{12} = 1 \tag{4.3}$$

Therefore, the length of the eye vector el can be calculated so the pitch and yaw angle of the eye can also be determined.



(a) (b)

Figure 4.3: Head and eye coordinates

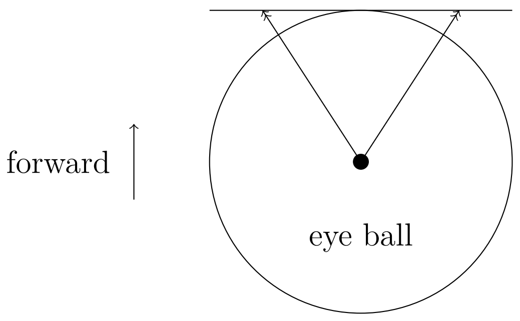


Figure 4.4: Top view of eye movement

$$el = \sqrt{(ex^2 + ey^2 + ez^2)} \tag{4.4}$$

$$epitch = \tanh(\frac{ey}{\sqrt{(ex^2 + ez^2)}}) \tag{4.5}$$

$$eyaw = \tanh(\frac{ex}{ez}) \tag{4.6}$$

But the pitch and yaw angle got from above equations are not the real angle of the view. The approximate field of view of an individual human eye is 60° superior (up), 60° nasal (towards the nose), $70° - 75°$ inferior (down), and $100° - 110°$ temporal (away from the nose and towards the temple). For both eyes the combined visual field is 130° vertical and 200° horizontal. To test the normal behavior of eye movement, a quick test of view range was done. The volunteer was asked to stare at a red dot on the wall without rotating the head. Then the dot was moved towards to the left within the visible range of the volunteer. The principle was not to make his eye feel pain or fatigued since people subconsciously rotated head to expand the view in order to reduce the strain of the eyes. Once he felt tired or uncomfortable of his eyes, the red dot stopped moving and the last location was marked. Then the view ranges of the right side and vertical side were tested. With the distance between the volunteer and the wall, view range of both the horizontal and vertical sides were measured that were 120° horizontal and 90° vertical.

With the upper and lower boundary of the X and Y coordinates of the eyes, the pitch and yaw angle ranges of the eyeball can be calculated through the equations above. The limit pitch angle calculated was 36.4° while the limit yaw angle calculated was 48.6°. According to the tested visible angle range, two coefficients α and β were applied so the real eye vector was estimated, where $\alpha = 90/36.4 = 2.47$ and $\beta = 120/48.6 = 1.55$. These two coefficients were appropriate for most of the eyes tested while may not be suitable for some other eyes.

$$epitch_{real} = \alpha \times epitch \tag{4.7}$$

$$eyaw_{real} = \beta \times eyaw \tag{4.8}$$

Thus, the pitch and yaw angles of the eye were determined. The next step was to combine the eye vector with the head vector. Actually, the orientation of the eye was the exact direction where people looked at. But with the rotation of the head coordinate system, the coordinate system of the eye was no longer the same as that before. The Z axis of the

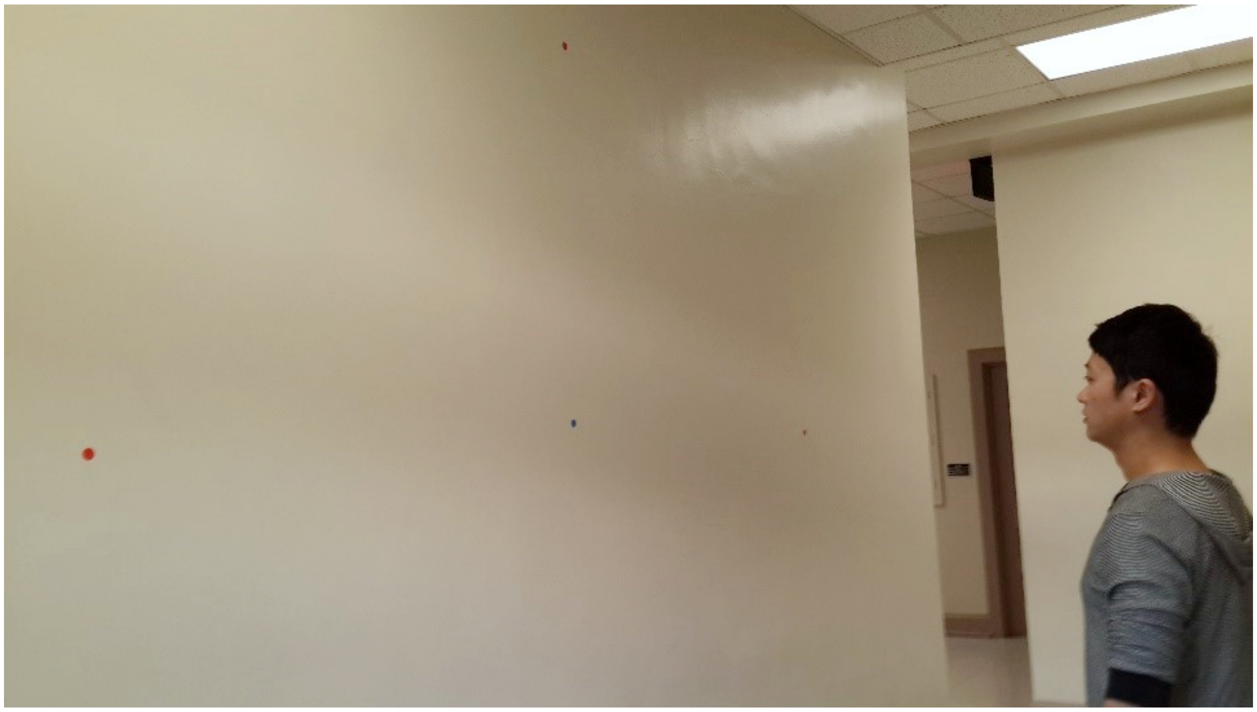


Figure 4.5: Sight range test with head fixed

eye vector was in the same direction of the head vector. Hence, the coordinate system of the eye vector was rotated horizontally and vertically with the corresponding pitch and yaw angle of the head motion. For *hpitch* and *hyaw*, the pitch and yaw angle of head that got from the head tracking program, the eye vector in the original coordinate system can be presented as:

$$epitch' = epitch_{real} + hpitch \tag{4.9}$$

$$eyaw' = eyaw_{real} + hyaw \tag{4.10}$$

4.1.3 Gaze points

In this study, the proper form indicating the combined unit vectors of the gaze along with head motion was to perform a heat map, while the heat map was generated from the 2D scatter plot. But the combined unit vectors that calculated from the above algorithm only expressed the overall pitch and yaw angles. A projection image was then aimed to help generating the 2D scatter plot. The idea plane was the windshield of the skidder, on account of that the driver looked around from near towards far to operate the skidder and detect the environment through the windshield.

Since the coordinate system of the eye was corrected in the previous section, assuming the gaze was like a laser, the projection point on the windshield was the point in 2D. The location of the projection point was determined by the overall pitch and yaw angle as well as the distance from the eye to the windshield. Plus, the transition distance generated by the head motion in X, Y, and Z directions. Also, the distance between the eye vector and the head vector in Z axis direction was the average distance from face to back of the head which was 180mm on average.

Therefore,

$$\tan{(epitch')} = \frac{y_w}{(L - hz \times 23 - 180)/\cos{(eyaw')}} \tag{4.11}$$

$$tan(eyaw') = \frac{x_w}{L - hz \times 23 - 180} \tag{4.12}$$

$$x_p = x_w + hx \times 20 \tag{4.13}$$

$$y_p = y_w + h_y \times 20 \tag{4.14}$$

(x_w, y_w) was the projection point created from the eye vector. While (x_p, y_p) was the overall projection point after the head transition being added.

4.2 Expression of gaze points

4.2.1 Heat map

Heat map is a 2D representation of data with gradual color change. Compared with the scatter plot, heat map is able to indicate the density or frequency of the data visibly. In general, the color in the heat map is presented by HSL. HSL stands for hue, saturation, and lightness. Usually, the red color means the high density zone while blue indicates the low density zone. However, colors like cyan, green, and yellow express the process that the points in a certain area is getting more and more intensive. To obtain a color bar shown as below, H is set from 0 to 240, while S equals to 1 and L is 0.5.

The procedure of generating the heat map contains four steps:

(1) Set a radius for each discrete point and create a buffer zone within the radius.

(2) From interior to exterior, apply a gradual grayscale value (from 0 to 255). The color is darker in the area closer to the point.

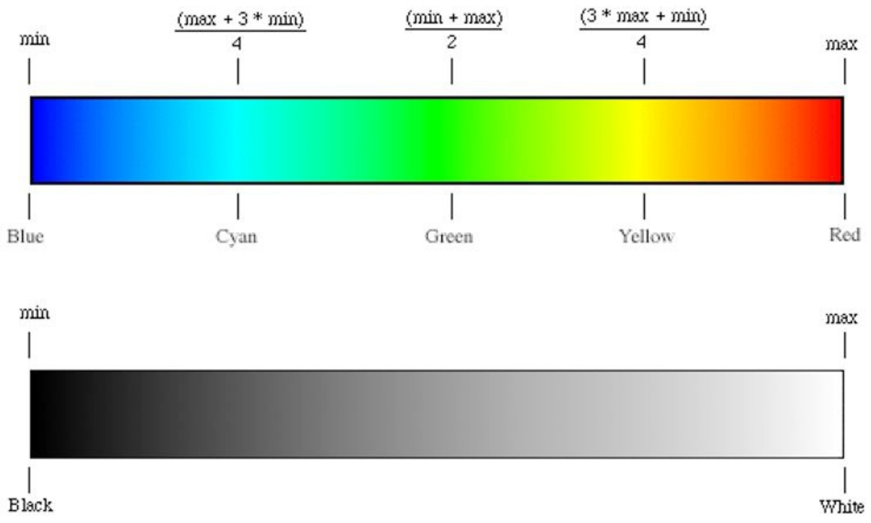


Figure 4.6: HSL and grayscale color model

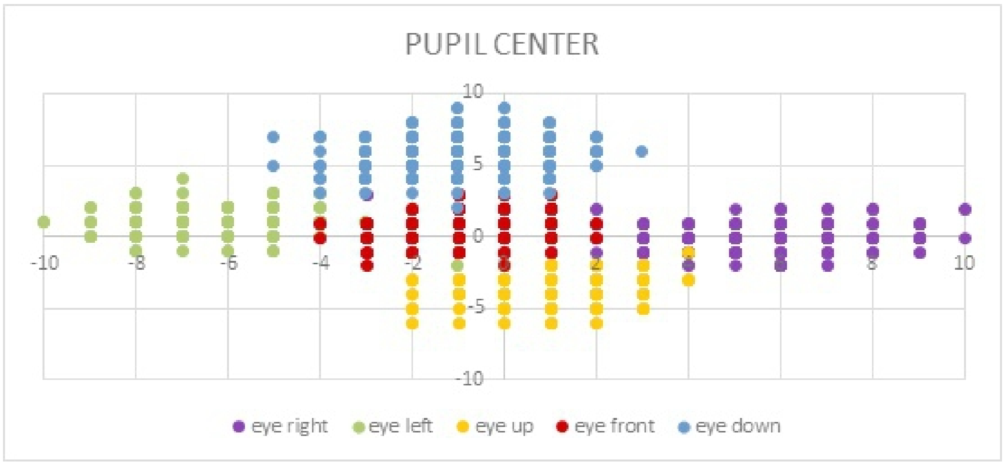
(3) As the grayscale value can be added, the larger value is, the whiter will be. So the intersection of the buffer zones is whiter and whiter with more points. Actually, this area shows hotter than the other zones.

(4) After all the buffer zones are added together, recolor the image with the color bar shown above according to the final grayscale values. Therefore, the heat map is successfully generated.

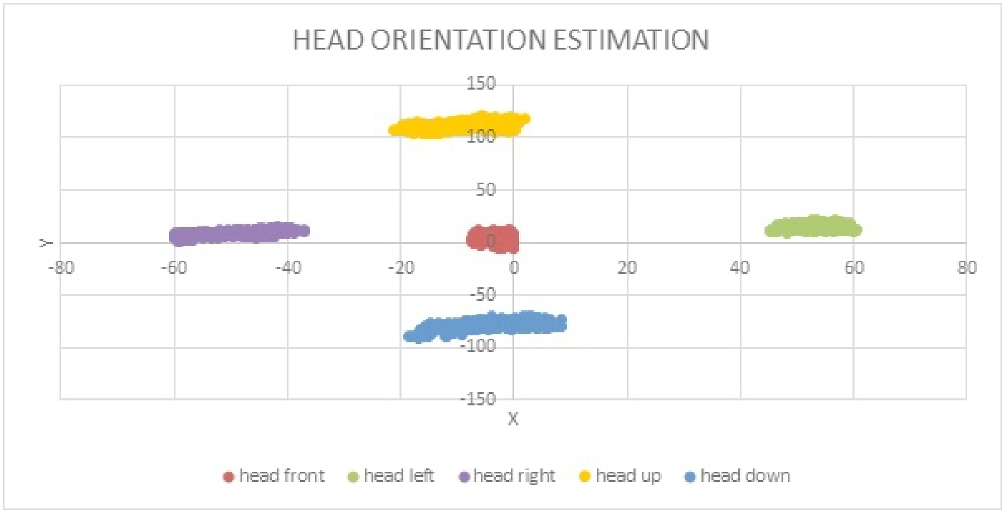
4.2.2 Reliability test under lab condition

Two individual tests were designed to evaluate the methods outlined in the previous section for estimating the gaze point on a 2D surface. The first involved keeping the head relatively stationary while moving the eye to view a total of five points placed on a flat surface in front of the test subject at a distance of 1 m. The five points were a 'center' plus four others placed at a uniform distance in the cardinal directions. The test subject was asked to orient their head towards each point in succession and, while the head remained fixed in that position, to move their eyes to the other points and keep them there briefly. The camera setup was as in the previous chapter with one mounted on a hard hat the subject wore and the camera pointed at the subject's face, and another behind the subject in a fixed location pointed towards the chessboard fixed to the rear of the hat. Video from both cameras was recorded and synchronized using a light pulse. Several images with the test subject's gaze fixed in each permutation of head orientation/view point were extracted from the video and the calculations done to extrapolate the line of sight to the viewing surface.

Figure 4.7 shows the relative eye (4.7a) and head (4.7b) positions for the test images. The eye positions are shown as the pupil center in units of pixels within the eye sub-image. Head positions are represented using the chessboard origin location and units are also in pixels.



(a)



(b)

Figure 4.7: Inputs of sight expansion: (a) pupil center location; (b) head pose estimation

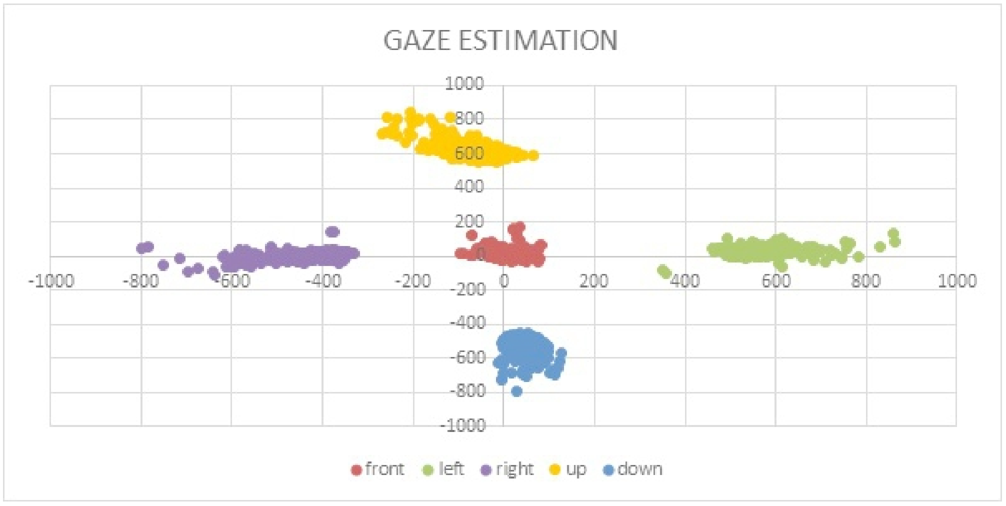
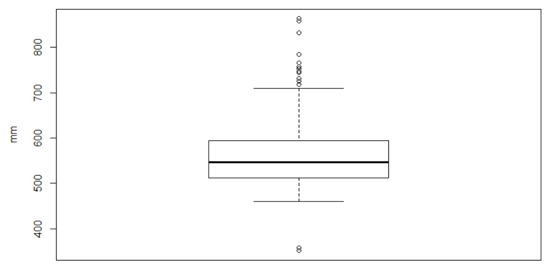


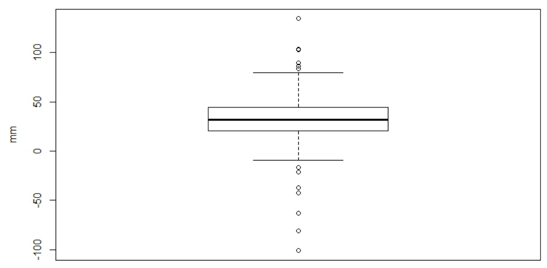
Figure 4.8: Calculated gaze points for fixed head orientations while rotating the eyes to look at a sequence of 5 locations.

The combined data (steady head orientation/roving pupil) are shown in 4.8, with units in mm. In general, the system could project the gaze point within a circular radius of about ±200 mm around the fixed viewing point. It was not possible to determine how much of the inaccuracy was related to subtle uncontrolled motions of the subject's head and eyes and how much was attributable to the algorithm itself. Although not highly accurate, this level of precision was felt to be acceptable for the purposes of this test and, since there was no way to effectively partition the error sources, the algorithm itself was not changed.

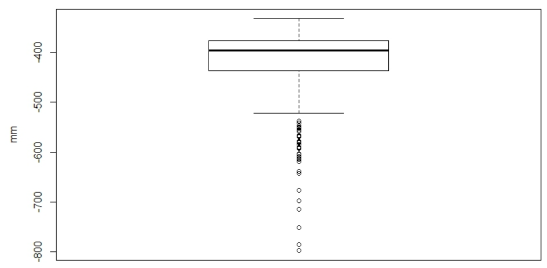
A second experiment was done in which the same test subject was asked to hold their gaze fixed on a single point while moving the head to other orientations, again along the cardinal directions. The mechanics of the test were as in the previous. Figure 4.10 shows the inputs to the gaze point reconstruction algorithm from the pupil (4.10a) and head (4.10b) orientations. Units are again in pixels.



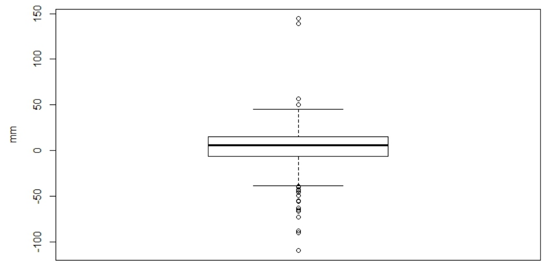
(a) x_left



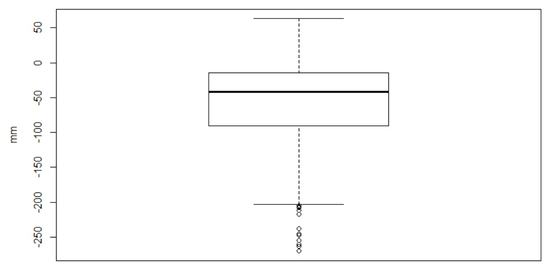
(b) y_left



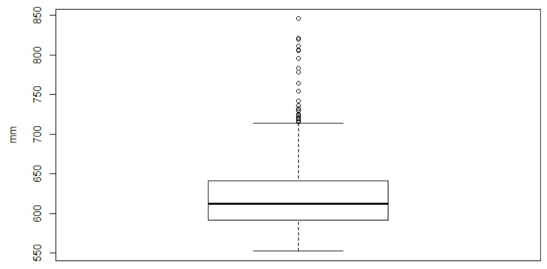
(c) x_right



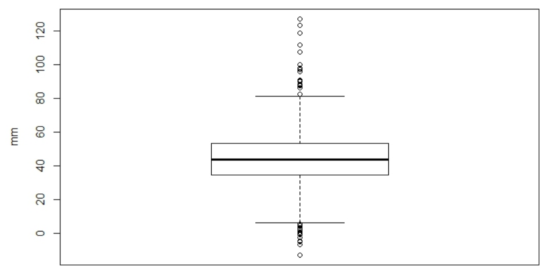
(d) y_right



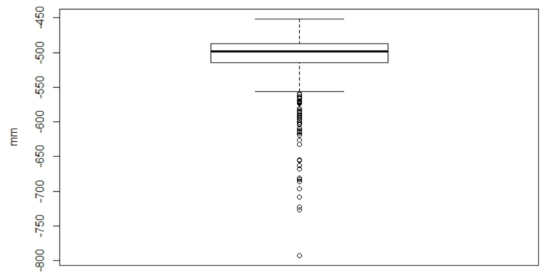
(e) x_up



(f) y_up

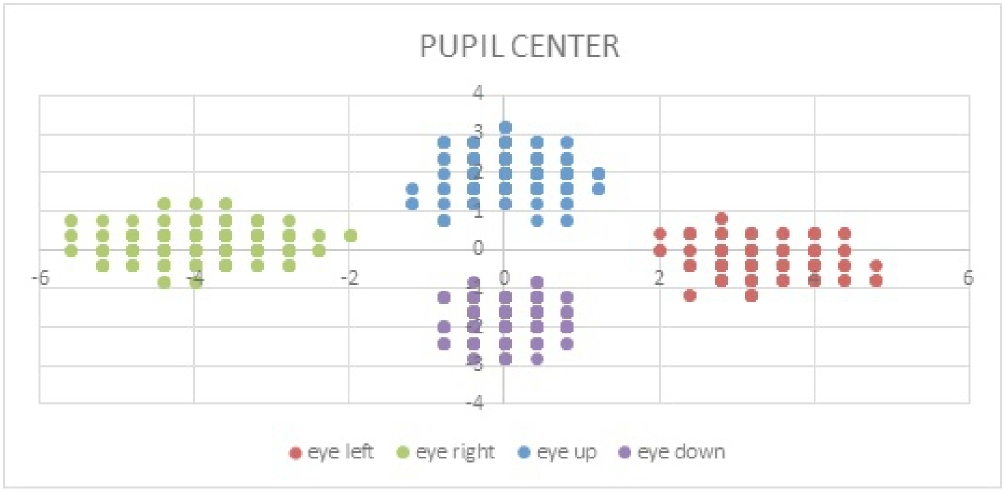


(g) x_down

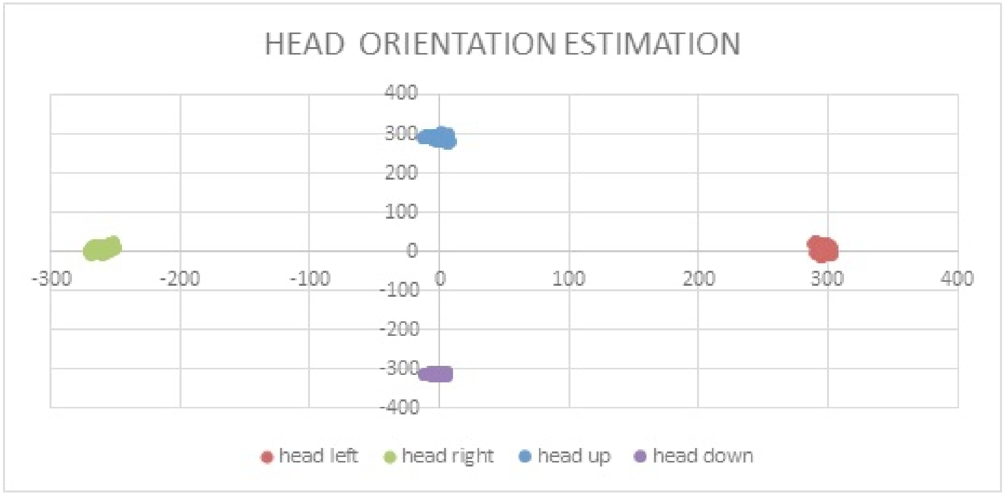


(h) y_down

Figure 4.9: Boxplots of four positons



(a)



(b)

Figure 4.10: Inputs of sight focus: (a) pupil center location; (b) head pose estimation

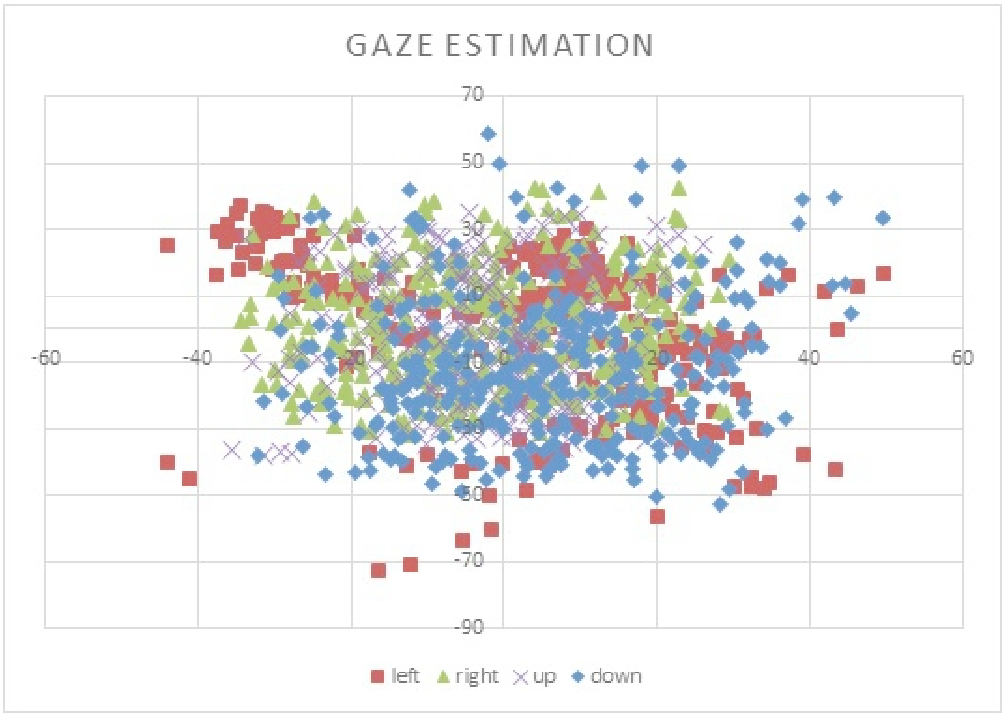


Figure 4.11: Gaze points of sight focus

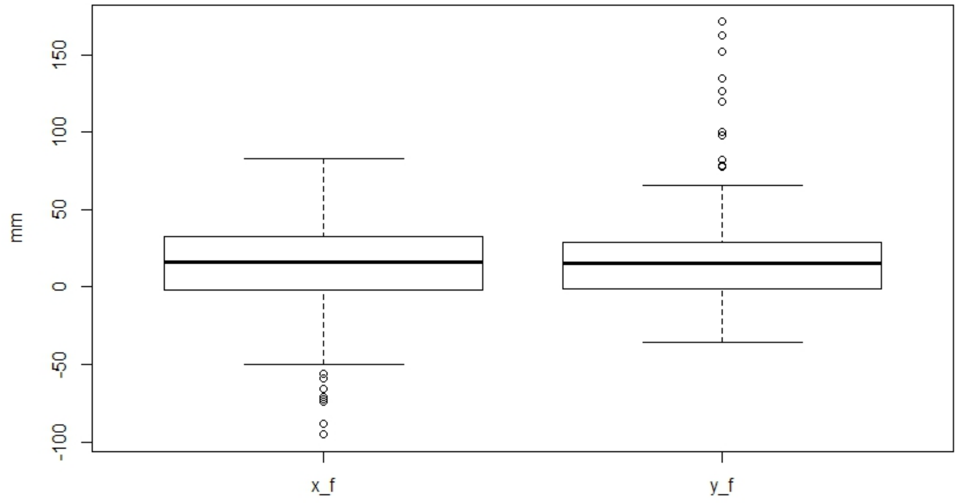


Figure 4.12: Boxplot of the gaze points

Figure 4.11 illustrates the results of the calculation of the gaze point. Ideally, the result should have been a single point at the location (0,0), but the calculated positions instead were clustered somewhat randomly around the origin, most being within a circular area with radius of about 40 mm. Although there may have been some bias of the gaze point prediction based on head orientation, it was not a large amount and was deemed negligible. It was concluded this test was further verification of the relative accuracy of the gaze point projection methods used in the study and no other changes were necessary to them.

4.2.3 Field calibration

To map gaze points to the front windshield in the skidder cab, it was necessary to recalibrate the gaze point projection parameters. A calibration experiment was performed using the setup shown in Figure 4.13. Two cameras were required to track head orientation since some of the time the operator turned to look out the rear of the cab and moved the chessboard out of the range of detection for the front-facing camera.

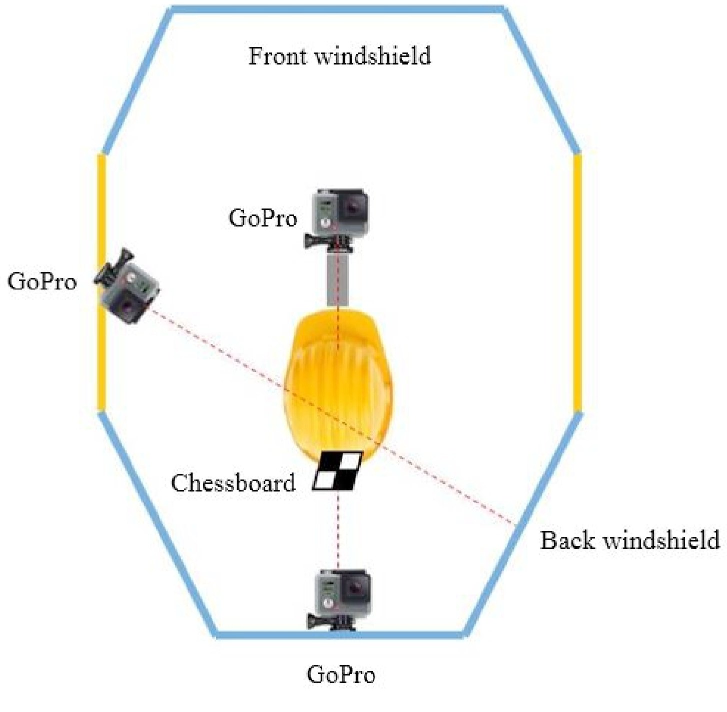
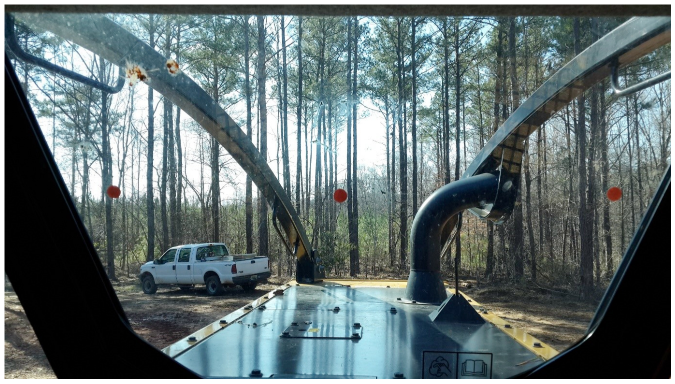
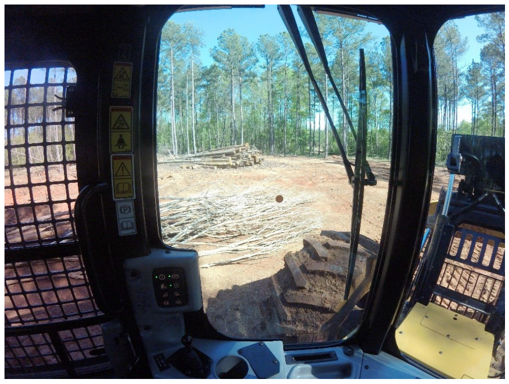
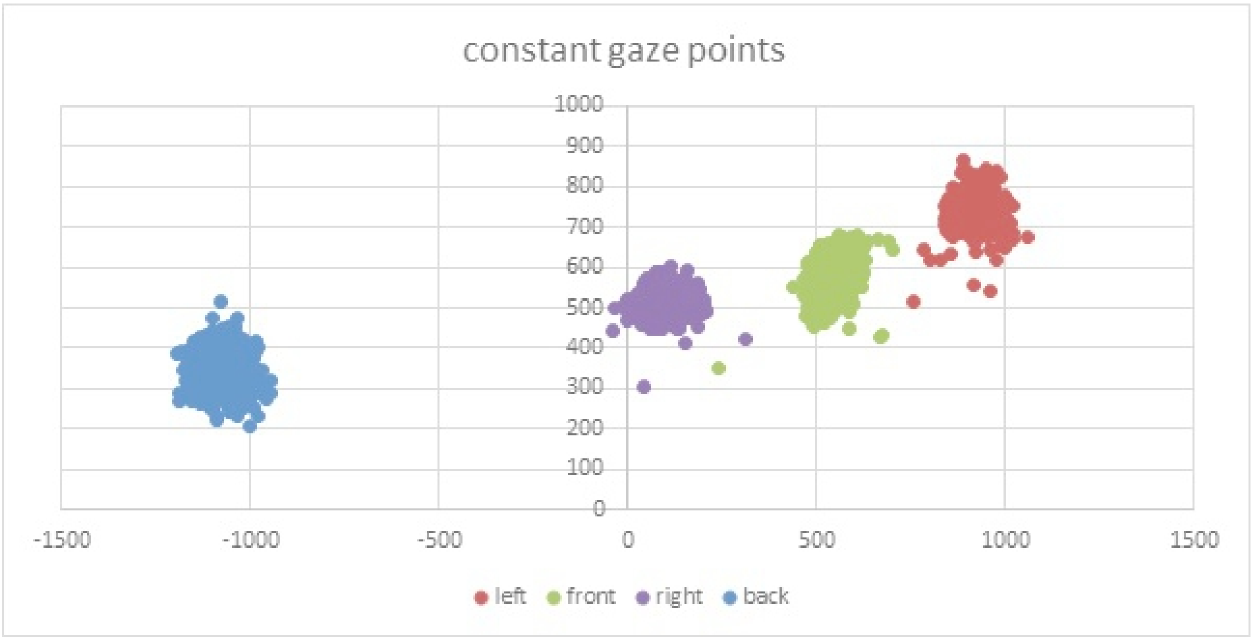


Figure 4.13: Top view of the tracking system as deployed in the skidder cab.

Three dots were marked on the front windshield of the skidder at about eye level along a horizontal line (Figure 4.14a). The points were placed 500 mm apart. The operator was asked to stare at each point for a period of about 10 seconds while video from the rear-mounted and hard hat-mounted cameras was recorded. Video from the two cameras was again synchronized using a flash of light. The distance from the operators eye to the windshield was also measured (1.5 m). The same process was repeated for the rear window and using the door-mounted camera to capture head orientation video, but only a single point was used in that case (Figure 4.14b).


(a) Calibration dots on the front window


(b) Calibration dot on the rear window



(c) Gaze points on both windows

Figure 4.14: Gaze points calibration in the skidder cab

After calculating gaze points for the forward looking video data, it was found the mean locations were very nearly 500 mm apart, which confirmed the choice of algorithm parameters. They were skewed, however, relative to the horizontal (Figure 4.14c) and subsequent positions were rotated by 11.3° to compensate for this effect. Since only a single point was taken on the rear window, this rotational effect could not be measured and was therefore ignored.

4.3 Tracking system field test

A field trial of gaze tracking was conducted using the procedures as outlined previously, namely the camera setup as in Figure 4.13 and using the light flash technique to align video data in time. An experienced skidder operator drove a Cat 525D around a track laid out in the Mary Olive Thomas tract in Auburn, AL. It took him about 10 minutes to complete a lap of the test path and he made a total of five laps. On two of the laps, the operator was asked to pick up and drop two different bundles of trees.

Example results are shown in Figures 4.15 to 4.17 and illustrate the ability of the tracking system to pick out pupil location. Figure 4.15 shows an example of the system working well despite dark ambient light conditions. The pupil was located, singulated, and a nice clear center was identified. Figure 4.16 shows an example where the area filter was necessary to remove multiple artifacts, only one of which was the pupil. To be consistent, the algorithm used a single eye for gaze tracking, normally the left, but, on occasion, detection of the left eye was not successful. In those cases, the right eye was sought as a backup and Figure 4.17 is an example of that occurrence. Overall, about 80% of frames over the entire recording were successfully processed and a valid gaze point determined. Processing time per frame on the computer system used in these tests was about 10 ms.

(a)

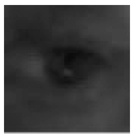
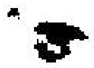
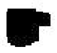
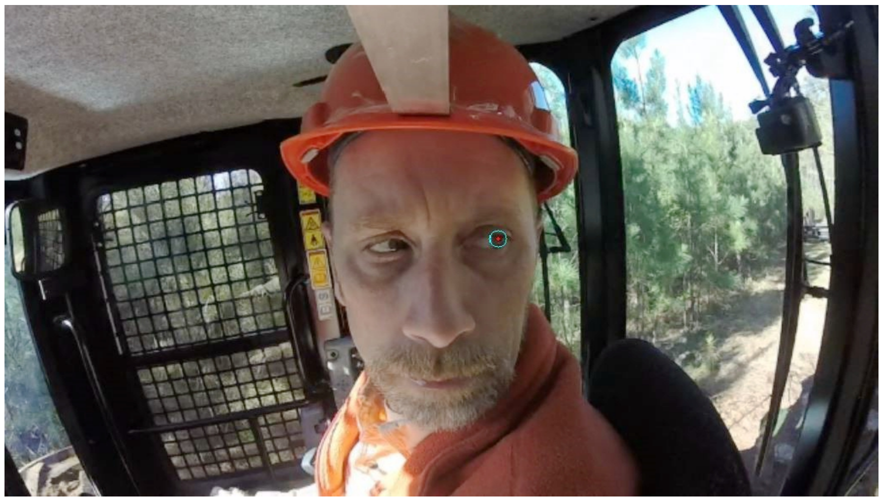



(b) (c) (d)

Figure 4.15: Example tracking frame


(a)

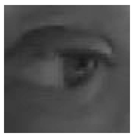
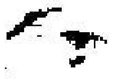
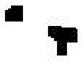
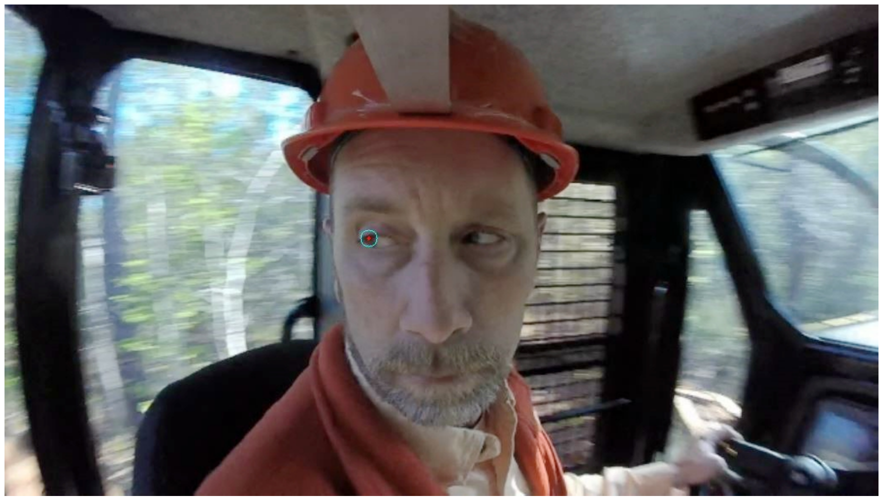
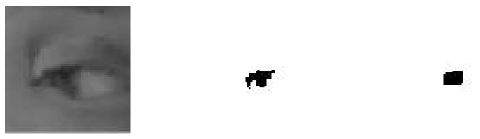



(b) (c) (d)

Figure 4.16: Example tracking frame: noise filter



(a)



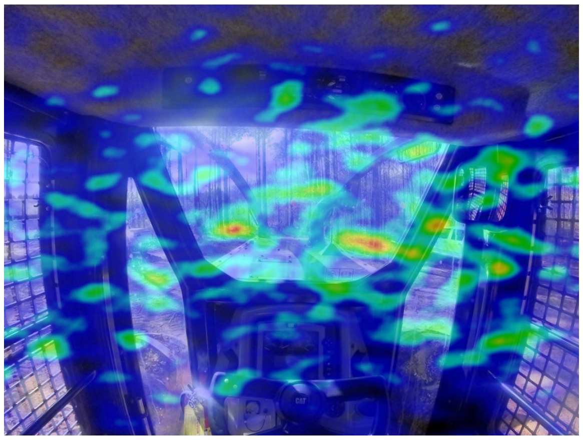
(b) (c) (d)

Figure 4.17: Example tracking frame: eye switched

4.4 Results from heat maps

The ‘heat map’ generated from the previous field test was shown in Figure 4.18. The darker, cooler zones were areas of low interest to the driver, while brighter, warmer (yellow and red) areas indicated high interest. For the front view, attention was mainly directed out the front window and most often towards the top corners of the hood. This view was likely the best indication to the operator of the overall direction of the skidder and was also, perhaps, where the greatest concern might be for some form of trouble to arrive. The small front-facing corner windows were also used quite often, probably to view the status of the front wheels. The right-side small window received a large amount of focus and this probably had something to do with the operator wishing to smooth his ride over the course of the track.

When looking to the rear, the operator’s attention was most often focused on the grapple itself, which was as expected. There was also, however, some attention paid to the side window, and other areas of the rear window, which may have indicated the operator backing the machine into position to grapple the load. For these tests, the front view was used to determine the gaze point 76% of the time, with the rear camera setup filling the remainder.


(a) Front

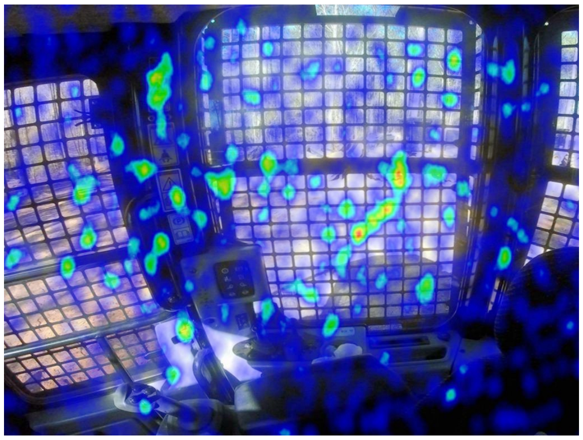
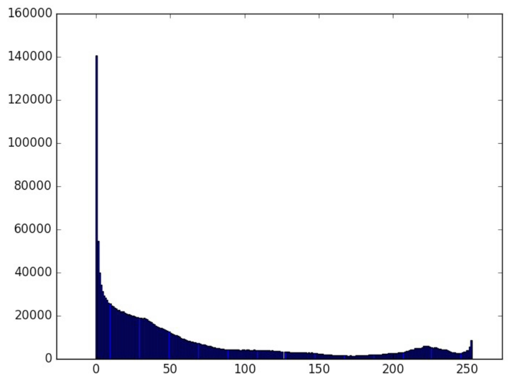
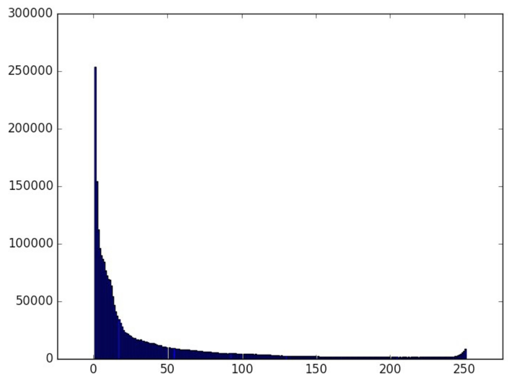

(b) Back

Figure 4.18: Views in the cab with heat map added



(a) Histogram of front heat map



(b) Histogram of back heat map

Figure 4.19: Relative histogram

In most instances, however, the operator's attention was not necessarily focused on any particular area. Figures 4.19a and 4.19b show histograms of the frequency of occurrence of gray values in the attention heat maps, to the front and rear, respectively. The cooler, darker (blue/green) colors indicating low rates of attention were mapped to the lower gray values, while greater attention was indicated by higher values. In both maps, more than 90%

of the gaze points were not directed towards any specific area. This indicates it is probably important to an operator to have stay visually aware of the totality of his surroundings for most of his time. In other words, it should be important to provide to a remote operator as complete a picture as possible of the skidder surroundings in or to maintain a complete situational awareness.

4.5 Conclusions

In this chapter, the algorithm of calculating 3D gaze vectors with head motion was introduced. The eye vector coordinate system was calibrated in order to determine the relative vector with the head motion vector. Then the overall vectors were later used to generate the heat map of the gaze points. The principle of drawing the 3D gaze vectors on the 2D heat map was to calculate the projection gaze points on the surface in front of the operator with a certain distance. With the shape and color information in the heat map, the interested area which the operators looked frequently can be found. Also, the percentages of time for each color areas during their operation can be determined by the histogram of the gray scaled heat map.

Chapter 5

Video-based driving test

With rapid development, remote control technology has been applied to many fields, for instance, unmanned aerial vehicle. Since forestry work is on of the most dangerous events, the accident rate in timber harvesting cannot be ignored. Usually the forestry workers are required sufficient safety awareness training in order to reduce the potential risk during work. Though machinery provides enough protection for operators, incidents caused by distraction, falling objects, and severe weather will still lead to harm or even death. However, because that the remote control technology has not yet been utilized in this field, workers have to operate machine in the forest personally.

The idea of video-based just driving came from the problem statement above. To test the feasibility of video-based driving on both the TurboForest mini skidder and Caterpillar 555D skidder, the live video system was firstly built. Then the test path was mapped with GPS coordinates as well as qualitative determination of ups and downs of the path. In this study, the ups and downs of the path was expressed by the vertical acceleration of the skidder. At last, the windows of the skidder were covered before the video-based test was executed.

5.1 Video system setup

The live video cameras used in this study was a 180° fisheye lens 1080p wide angle web USB camera (Fig 5.1). The resolution was set to 1024×768 along with 30 fps. A 19 inch TV was set as the monitor. Considering the severe vibration in forest, that the TV along with the bracket totally weighed 5 pounds was held by a ram mount which can support up to 20 pounds. A HP laptop was in charge of processing the video to the monitor and acquiring

GPS signal including GGA as well as VTG. A free software called OBS (Open Broadcaster Software) was a perfect solution for this problem. This software primarily was utilized in live streaming and recording. However, with this software, multiple cameras or other video, audio, image sources can be added to generate nice scenes without device conflicts. All thr devices were powered using an inverter that converted 12V DC from the battery of the skidder to 110V AC.

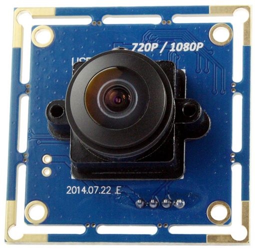


Figure 5.1: Fisheye cam: ELP-USBFHD01M-L180

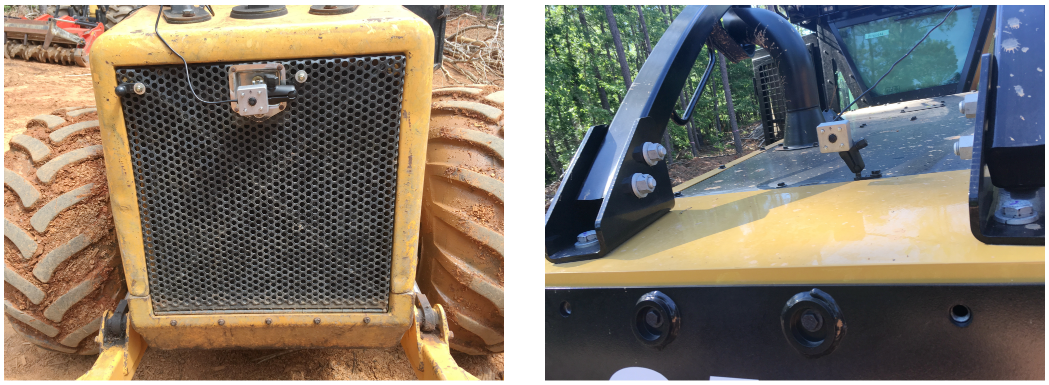


Figure 5.2: Fisheye camera set on the front of both skidders

Since the driver mostly focused on the areas around the hoods of the skidder while driving forward, then the fisheye camera that protected by a case was set in the head of the skidder (See Fig 5.2). The view from the fisheye camera which was set in the front was wider compared with driver's vision inside the cab. Unlike the driver's vision that was partly blocked by the hood, the live video from the monitor showed a clear image of the upcoming path (Fig 5.3). In a certain degree, it may help the driver get a better vision while driving the skidder.

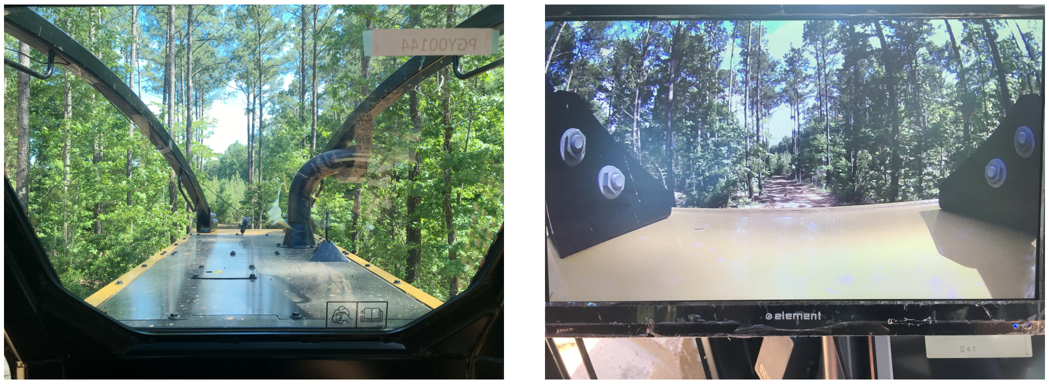


Figure 5.3: Vision comparison between driver's vision and live video

5.2 Path mapping

To test how good the driver can handle the skidder only based on the video from the cameras, the path has to be firstly mapped.

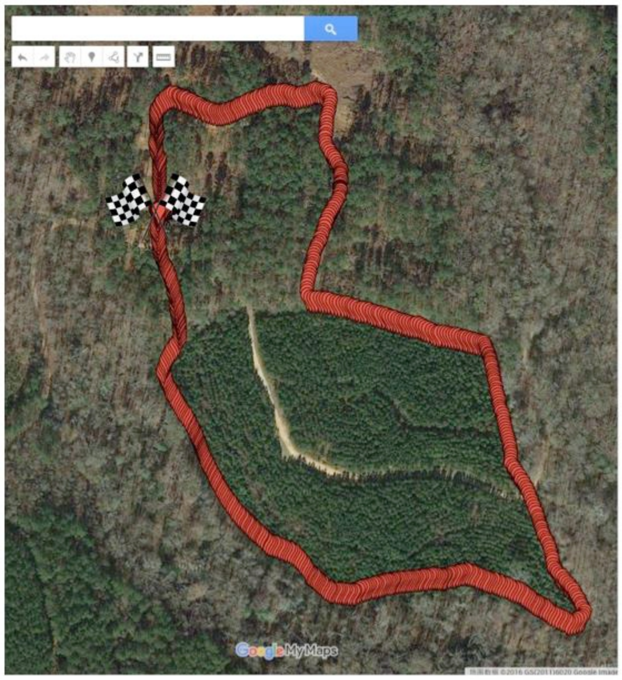


Figure 5.4: Skidder path in MOT Demonstration Forest

The skidder path shown as figure 5.4 was inside the Mary Olive Thomas (MOT) Demonstration Forest. It took the skidder 15 minutes to finish a lap on average. Considering the ups and downs on the path, a GoPro camera (Fig 5.5a) was installed in the front of the skidder to record the road condition. Bad road conditions (Fig 5.5b) which resulted to the severe vertical vibrations were also recorded in order to have the overall view of the whole path condition. Generally speaking, it's easy to drive on the plain and straight road.

Therefore, the sections of the road or corners that the driver may feel difficult to drive through were the standard that how good the driving can be only with video-based driving system.

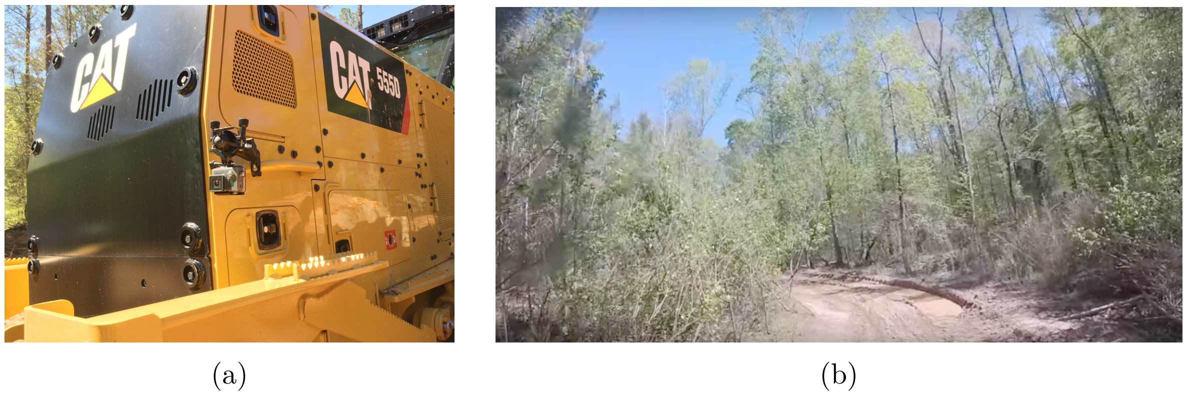


Figure 5.5: Road condition recording: (a) GoPro camera setup; (b) Example frame of bad road condition

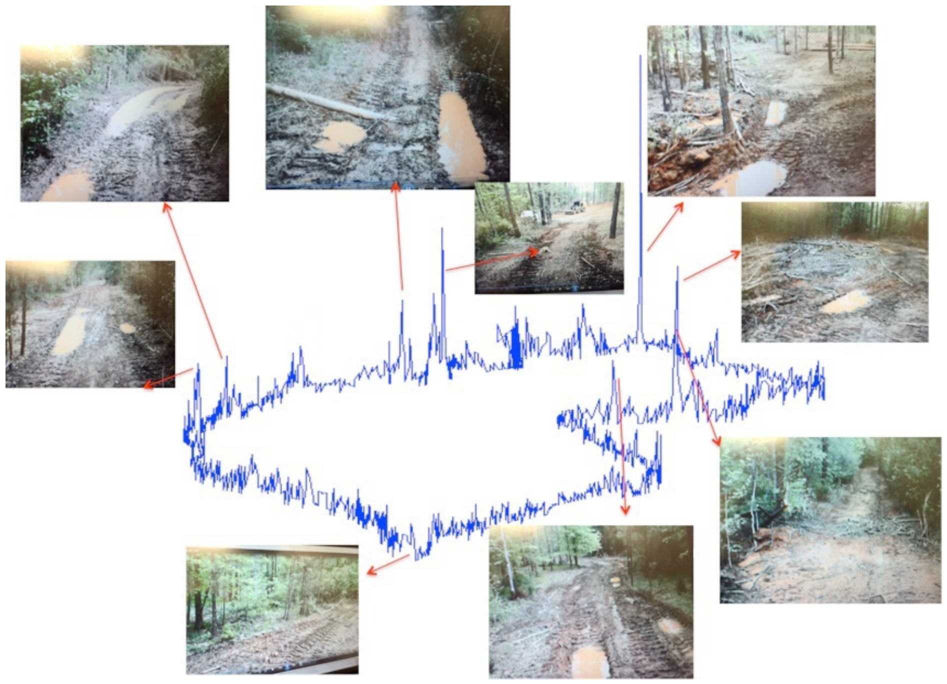


Figure 5.6: Path mapping by vertical acceleration with severe road conditions

However, to tell how bad the road condition was not enough to quantize the road condition. The terrain was determined by the vertical acceleration of the skidder measured from ADXL362 accelerometer (A cooperation study of skidder vibration by Dr. Pengmin Pan, Biosystems Engineering, Auburn University, 2016). Figure 5.6 indicated the road condition through vertical vibration and pinned several most severe road conditions with visible big changes of the acceleration.

5.3 Results and discussions

In this study, the tests evaluated the feasibility of video-based driving through comparing the overall driving time and average speed. The commercial change of the speed was also considered as a factor of evaluation. Two types of driving test were included. One was mormally driving while the other one was to drive the skidder with windows covered by paper. Thus, the driver was able to handle the machine based on the views provided by the monitor (Fig 5.7). A TurboForest mini skidder was used for the tests.

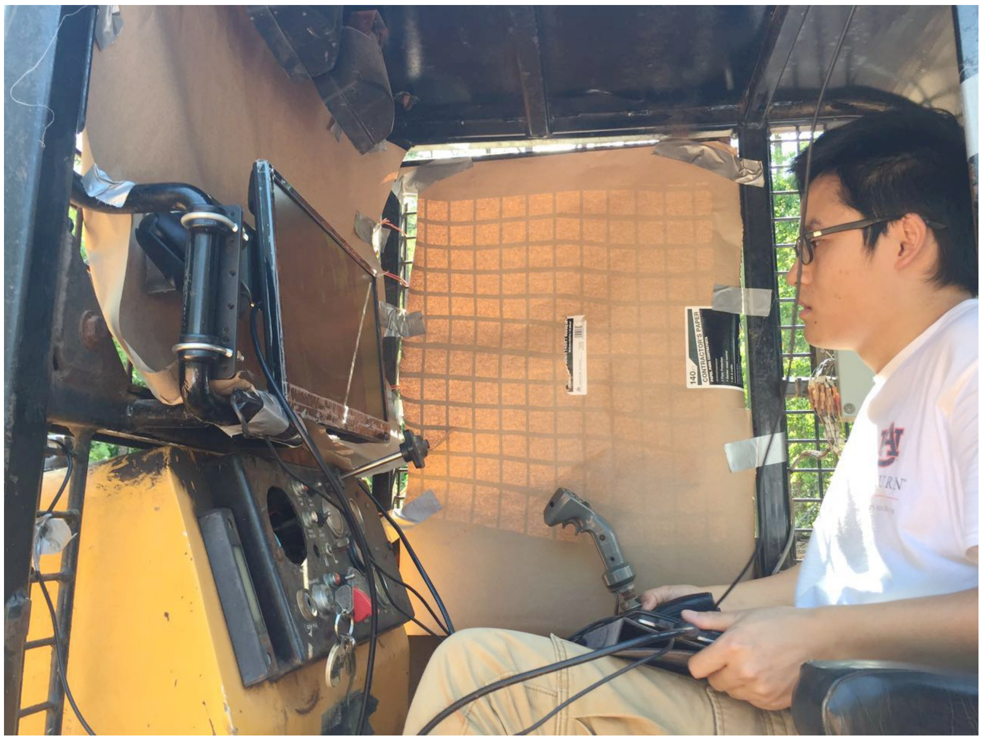


Figure 5.7: A view of the skidder cab covered by paper

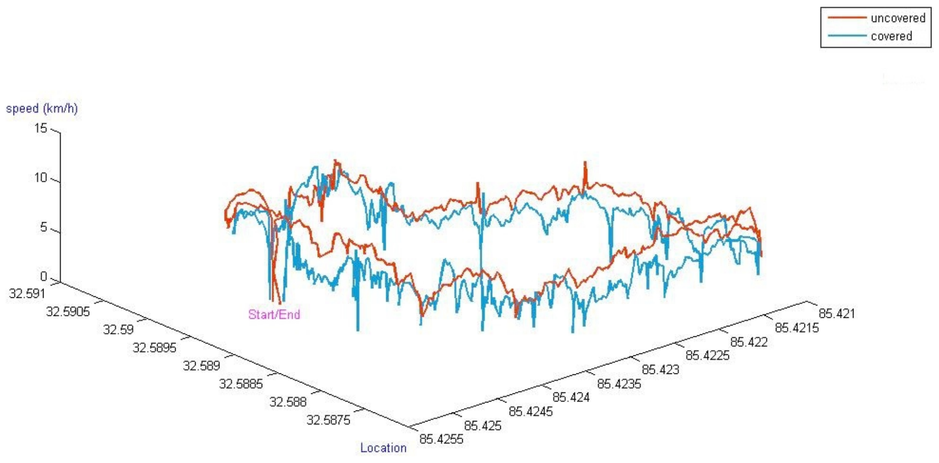
The video-based driving test showed that the idea of remote driving skidder can be accessed. For a single lap driving test ran on mini skidder by Chennan Xue with windows covered, the overall time was 648 seconds compared with 552 seconds with windows uncovered. However, the average speed with windows covered was $5.15km/h$ compared with uncovered which was $5.90km/h$. For multiple laps driving tests (5 laps without windows covered and 4 laps with windows covered) by Rees Bridges, the average overall time of windows covered was 572.1 seconds while it was normally 422.4 seconds. The average speed was $8.50km/h$ with windows covered compared with $6.66km/h$. From the 3D speed-location plots (Fig 5.8), it can be found that the skidder frequently stopped with windows covered where the speed dropped to 0. The driver had to stop to observe the environment as well as the direction.

However, results on the Caterpillar 555D skidder (Table 5.1 and Fig 5.9) showed the overall productivity of the remote operation can be increased by getting familiar with the video-based driving. With the windows coverd, the second lap was faster than the first lap. The overall operation time with windows covered probably can match up with normal driving after practicing.

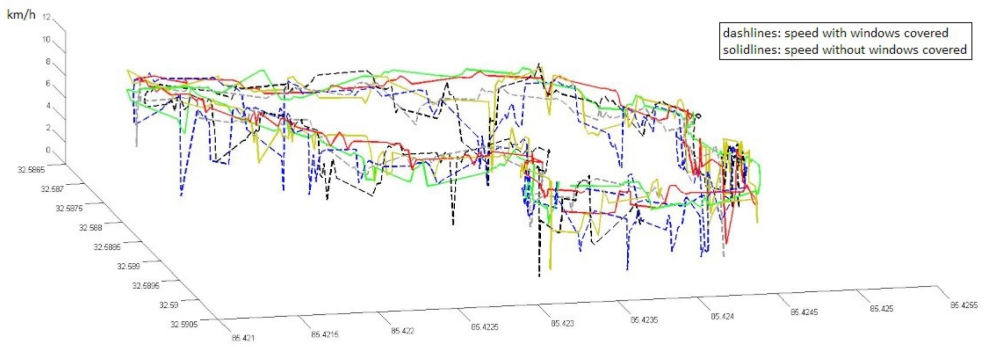
Type	**Ave. speed (km/h)**	**Overall time (s)**
Uncovered	6.718	304
Covered 1	4.418	471
Covered 2	4.819	436

Table 5.1: Average speed and overall time comparison

The feedback from the drivers pointed that the main problem was caused by the lag of the camera. Although the fisheye lens can provide better vision for the driver, the distortion of the video may result in the fatigue of the eyes.



(a)



(b)

Figure 5.8: Speed-location comparisons on mini skidder: (a) Single lap; (b) Multiple laps

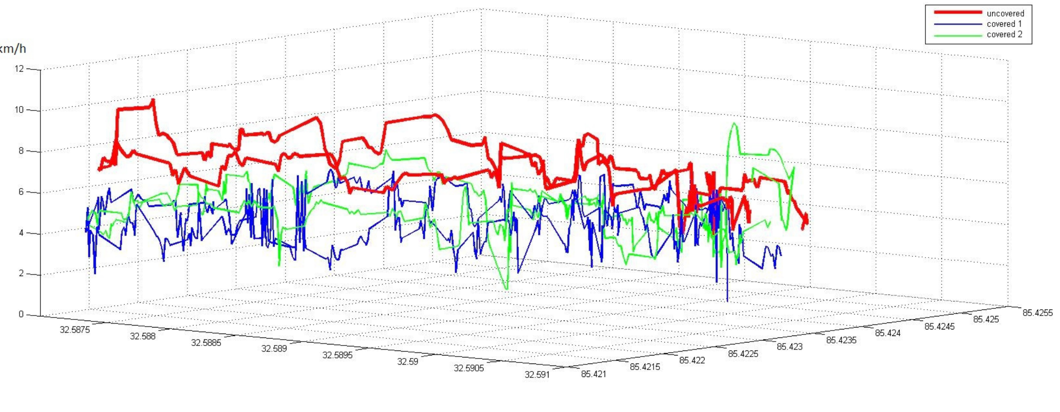


Figure 5.9: Speed-location comparison on Caterpillar 555D

Chapter 6

Conclusions and future work

6.1 Conclusions

6.1.1 Hardware and software

Creation of the head-mounted gaze and head motion tracking was based on a simple hard hat. A GoPro camera was mounted in the front of the hard hat for tracking the real-time gaze direction. A 5×4 chessboard was attached on the back of the hard hat. With a GoPro camera installed right behind, the head motion can be recorded. Besides, another GoPro camera was set on the left of the driver to capture the frames when they looked back.

Image processing including gray scale, binary conversion, and morphological transformation was used in order to reduce noises and get a clear image of the pupil. According to the coordinates from the center of the pupil, plus parameters of the eye ball, the gaze vectors were calculated. For the head motion estimation, the camera was firstly calibrated for the sake of removing the distortion. Then, a 2D-3D conversion was executed through RANSAC function. Therefore, the rotation and transition of the head motion can be tracked so that the vectors of head motion were solved.

6.1.2 Tracking system and results

The principle of gaze tracking with head motion was tested. Under lab condition, the detection rate of gaze tracking achieved over 90% while in field test it reached 80% above. The overall accuracy can be explained as that the gaze points dropped within a circle of $40mm$ when the user starred at a point $1000mm$ in the front.

From the heat map, the color areas indicated the most interested and less interested zones that the skidder driver looked at during forestry work. For the front view, the driver primarily focused on the road condition to handle the skidder. However, they were more concentrated on the grapple, rear tire and control panel when they looked back.

6.1.3 Video-based driving test

The results showed that it was possible to handle the skidder based on the live video. The overall time can be reduced by more practice. Once the driver becomes more familiar with the driving system and path, shorter time it will be. But the first problem was to solve the lag of the video. A faster laoptop or tablet with better processor may be the answer. Also, additional side views may be required in order to get a larger range of vision. The fisheye camera worked great with super wide angle but it came with distortion of the image which may make the driver dizzy.

6.2 Future work

6.2.1 Hardware and software

In this study, the custom designed helmet was a bit heavy. For long time tracking test, the helmet has to be light and comfortable. Python may not the best solution for fast processing in developing gaze tracking system. C or C++ language can be used to code the programs. The process speed may be faster than that in this study. Also, the small processing unit such as Respberry Pi was initially considered for tracking, but found it was too slow. However, a portable tracking unit is always a great product that can be used in research or daily life.

6.2.2 Methods of tracking

The gaze detection method in this study was not able to detect pupil with user wearing glasses. Also, it can hardly detect pupil under bad lighting condition. Therefore, image processing and algorithm for eliminating reflection is the big step towards a robust tracking system. Besides, infrared lighting source can be added in order to get better results in pupil tracking. Feature-based head pose estimation has great potential in modifying current tracking system. Even though the feature-based algorithm has relative high false positive rate, with the well trained classifiers, it can be used for head pose estimation eventually based on video. However, head-mounted head pose estimation also can be achieved by using several light sources. Through tracking the lighting points and calculate their corresponding positions, the pose can be estimated. This technology is now widely applied on the virtual-reality gaming or simulation. But the problem that the light source is easily corrupted by other light source needs to be solved.

6.2.3 Accuracy of the overall tracking system

The precision of the gaze estimation system requires the accuracy of the gaze tracking system, head pose estimation system, and the algorithm that combine the previous two results together to generate the final heat map. Each part is essential to the whole system. However, among these three parts, the accuracy of gaze tracking dose really matter. New algorithm or methods of training classifiers are the key factors of improving the tracking accuracy. Currently, Haar-like feature-based detection is the fastest and efficient way to detect object. Thus, the classifier as the fundamental unit in the tracking system plays a big role. So for specific tracking tasks, it is better to train the classifiers with explicit positive and negative images from the target source.

6.2.4 Video-based driving

First of all, the lag of the video should be reduced. With the feasibility of the remote driving, the next step may be testing operating the grapple to grab woods based on video. The camera set on the skidder needs to cover the current dead zones of the driver.

Bibliography

P Albizu-Urionabarrenetxea, E Tolosana-Esteban, and E Roman-Jordan. Safety and health in forest harvesting operations. diagnosis and preventive actions. a review. *Forest Systems*, 22(3):392–400, 2013. ISSN 2171-9845.

David Beymer and Myron Flickner. Eye gaze tracking using an active stereo head. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 18, 2003 - June 20, 2003*, volume 2 of *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages II/451–II/458. Institute of Electrical and Electronics Engineers Computer Society. ISBN 10636919.

Gabriele Fanelli, Juergen Gall, and Luc Van Gool. Real time 3d head pose estimation: Recent achievements and future challenges. In *Communications Control and Signal Processing (ISCCSP), 2012 5th International Symposium on*, pages 1–4. IEEE. ISBN 1467302740.

Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010. ISSN 0162-8828.

Di Gao, Guisheng Yin, Weijie Cheng, and Xiaoning Feng. Non-invasive eye tracking technology based on corneal reflex. *Procedia Engineering*, 29:3608–3612, 2012. ISSN 1877-7058.

Carola Häggström, Martin Englund, and Ola Lindroos. Examining the gaze behaviors of harvester operators: an eye-tracking study. *International Journal of Forest Engineering*, 26(2):96–113, 2015. ISSN 1494-2119.

Huy Tho Ho and Rama Chellappa. Automatic head pose estimation using randomly projected dense sift descriptors. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 153–156. IEEE. ISBN 1467325341.

Q. Ji. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5):357–377, 2002. ISSN 10772014. doi: 10.1006/rtim.2002.0279.

Qiang Ji and Zhiwei Zhu. Eye and gaze tracking for interactive graphic display. In *2nd International Symposium on Smart Graphics, SMARTGRAPH '02, June 11, 2002 - June 13, 2002*, volume 22 of *ACM International Conference Proceeding Series*, pages 79–85. Association for Computing Machinery. doi: 10.1145/569005.569017. URL `http://dx.doi.org/10.1145/569005.569017http://dl.acm.org/citation.cfm?doid=569005.569017`.

Chris L Kleinke. Gaze and eye contact: a research review. *Psychological bulletin*, 100(1):78, 1986a. ISSN 1939-1455.

Chris L Kleinke. Gaze and eye contact: a research review. *Psychological bulletin*, 100(1):78, 1986b. ISSN 1939-1455.

Dongheng Li, David Winfield, and Derrick J Parkhurst. Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 79–79. IEEE. ISBN 0769523722.

Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900–I–903 vol. 1. IEEE. ISBN 0780376226.

Päivi Majaranta and Andreas Bulling. *Eye tracking and eye-based human–computer interaction*, pages 39–65. Springer, 2014. ISBN 1447163915.

Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. Example-based object detection in images by components. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(4):349–361, 2001. ISSN 0162-8828.

Carlos H. Morimoto and Marcio R. M. Mimica. Eye gaze tracking techniques for interactive applications. *Computer Vision and Image Understanding*, 98(1):4–24, 2005. ISSN 10773142. doi: 10.1016/j.cviu.2004.07.010.

Carlos H. Morimoto, Arnon Amir, and Myron Flickner. Free head motion eye gaze tracking without calibration. In *Conference on Human Factors in Computing Systems, April 20, 2002 - April 25, 2002*, Conference on Human Factors in Computing Systems - Proceedings, pages 586–587. Association for Computing Machinery.

Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000. ISSN 0920-5691.

C Papazov, TK Marks, and MJ Jones. Real-time head pose estimation and facial feature localization using a depth sensor and triangular surface patch features. 2015.

Francisco J Parada, Dean Wyatte, Chen Yu, Ruj Akavipat, Brandi Emerick, and Thomas Busey. Experteyes: Open-source, high-definition eyetracking. *Behavior research methods*, 47(1):73–84, 2015. ISSN 1554-3528.

Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 746–751. IEEE. ISBN 0769506623.

Roberto Valenti, Nicu Sebe, and Theo Gevers. Combining head pose and eye location information for gaze estimation. *Image Processing, IEEE Transactions on*, 21(2):802–815, 2012. ISSN 1057-7149.

Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 606–613. IEEE. ISBN 1424444209.

Francisco Vicente, Zehua Huang, Xuehan Xiong, Fernando De La Torre, Wende Zhang, and Dan Levi. Driver gaze tracking and eyes off the road detection system. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):2014–2027, 2015. ISSN 15249050. doi: 10.1109/TITS.2015.2396031. URL `http://dx.doi.org/10.1109/TITS.2015.2396031`.

Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol. 1. IEEE. ISBN 0769512720.

Jing Zhang, Li Zhuo, Zhenwei Li, and Yingdi Zhao. An approach of region of interest detection based on visual attention and gaze tracking. pages 228–233, 2012. doi: 10.1109/icspcc.2012.6335613.

Xiangxin Zhu and Deva Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2879–2886. IEEE. ISBN 1467312266.

Appendices

Appendix A

Python code

A.1 Gaze tracking

```
import numpy as np
import cv2

faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eyeCascade = cv2.CascadeClassifier('haarcascade_eye.xml')
cap = cv2.VideoCapture('*.avi')
n=0

while(cap.isOpened()):
    cx=0
    cy=0
    n=n+1

    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(
         gray,
         scaleFactor=1.1,
         minNeighbors=5,
         minSize=(30, 30),
         flags=cv2.CASCADE_SCALE_IMAGE
         )
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

    eyes = eyeCascade.detectMultiScale(
            gray,
            scaleFactor=3,
            minNeighbors=8,
            minSize=(40, 40),
            flags=cv2.CASCADE_SCALE_IMAGE
            )
##    print eyes
    for (ex,ey,ew,eh) in eyes:
##            cv2.rectangle(gray, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)

            str1 = ''.join(map(str,eyes))
            str2 = str1.replace("[","")
            str3 = str2.replace("]"," ")
            a = [int(s) for s in str3.split() if s.isdigit()]
            b = np.array(a)

            biggest_b = np.argmax(b)
            ex = format(np.amax(b))
            if biggest_b+1 < len(b):
                ey = format(b[biggest_b+1])

            ex = int(ex)
            ey = int(ey)
            ew = int(ew)
            eh = int(eh)

            crop = frame[ey+20:ey+eh-10,ex:ex+ew]
            crop_gray = cv2.cvtColor(crop,cv2.COLOR_BGR2GRAY)
            cv2.imshow("crop_gray",crop_gray)

            A = cv2.normalize(crop_gray,crop_gray,0,255,cv2.NORM_MINMAX)

            _ ,thresh = cv2.threshold(A,50,255,cv2.THRESH_TRUNC)
            cv2.imshow("TRUNC",thresh)
##            blur = cv2.blur(thresh,(3,3))
            _ ,thresh_1 = cv2.threshold(thresh,35,255,cv2.THRESH_BINARY)
            cv2.imshow("thresh",thresh_1)
            kernel = np.ones((4, 4), np.uint8)
            kernel_1 = np.ones((5, 5), np.uint8)
            closing = cv2.morphologyEx(thresh_1, cv2.MORPH_CLOSE, kernel_1, iterations=1)
            closing_1 = cv2.morphologyEx(closing, cv2.MORPH_OPEN, kernel_1, iterations=1)
            cv2.imshow("closing",closing_1)

            _,contours,hierarchy = cv2.findContours(closing_1, 1, 2)

            for cnt in contours:
                area = cv2.contourArea(cnt)
                if area > 50 and area < 300:
                    max_index = np.argmax(area)
                    cnt_max = contours[max_index]
                    (cx,cy),radius = cv2.minEnclosingCircle(cnt_max)
                    cx = int(cx)
                    cy = int(cy)
                    center = (cx,cy)
##                    radius = int(radius)
                    radius = 9
                    cv2.circle(crop,center,radius,(255,255,0),1)
                    cv2.circle(crop,center,0,(0,0,255),3)
    print n,cx,cy

    cv2.imshow('frame',frame)
    if cv2.waitKey(2) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

A.2 Head pose estimation

```python
"""
@author: xkevin
2/2/2016 15:40
"""

import numpy as np
import cv2
import glob

'--*load test pictures to get camera parameters*--'
# termination criteria
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((6*7,3), np.float32)
objp[:,:2] = np.mgrid[0:7,0:6].T.reshape(-1,2)

# Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.

images = glob.glob('*.jpg')

for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    # Find the chess board corners
    ret, corners = cv2.findChessboardCorners(gray, (7,6),None)

    # If found, add object points, image points (after refining them)
    if ret == True:
        objpoints.append(objp)

        corners2 = cv2.cornerSubPix(gray,corners,(11,11),(-1,-1),criteria)
        imgpoints.append(corners2)

        # Draw and display the corners
        img = cv2.drawChessboardCorners(img, (7,6), corners2,ret)
        cv2.imshow('img',img)
        cv2.waitKey(5)

cv2.destroyAllWindows()

'--*start real time estimation*--'

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],None,None)
print 'camera matrix & distortion coefficients'
print mtx, dist

def draw(img, corners, imgpts):
    corner = tuple(corners[0].ravel())
    img = cv2.line(img, corner, tuple(imgpts[0].ravel()), (255,0,0), 5)
    img = cv2.line(img, corner, tuple(imgpts[1].ravel()), (0,255,0), 5)
    img = cv2.line(img, corner, tuple(imgpts[2].ravel()), (0,0,255), 5)
    return img
w = 4
h = 5
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)
objp = np.zeros((w*h,1,3), np.float32)
objp[:,:,:2] = np.mgrid[0:h,0:w].T.reshape(-1,1,2)

axis = np.float32([[3,0,0], [0,3,0], [0,0,-3]]).reshape(-1,3)

n = 0

video_capture = cv2.VideoCapture(0)

while True:
    pyr=0
    x=0
    y=0
    z=0
    n = n + 1
    _, frame = video_capture.read()
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    ret, corners = cv2.findChessboardCorners(
                    gray,
                    (h,w),
                    cv2.CALIB_CB_ADAPTIVE_THRESH+cv2.CALIB_CB_NORMALIZE_IMAGE+cv2.CALIB_CB_FAST_CHECK)

    if ret == True:
        corners2 = cv2.cornerSubPix(gray,corners,(12,12),(-1,-1),criteria)

        # Find the rotation and translation vectors.
        _,rvecs, tvecs, inliers = cv2.solvePnPRansac(objp, corners2, mtx, dist)
##        print rvecs, tvecs
        rmtx = cv2.Rodrigues(rvecs)[0]
        z = np.zeros((3,1))
        rmtx1 = np.append(rmtx,z,axis=1)
##        print rmtx1
        _,_,tvec,rmtxx,rmtxy,rmtxz,eulerangles = cv2.decomposeProjectionMatrix(rmtx1)
        pitch  = eulerangles[0]
        yaw = eulerangles[1]
        roll = eulerangles[2]
        if roll > 0:
            roll = 180 - roll
        else:
            roll = -roll - 180
        pitch = str(pitch)
        yaw = str(yaw)
        roll = str(roll)
##        order = '{0}'.format(n)

        # output with comments
##        p = 'The pitch angle(U+D-) is: ' + pitch
##        y = 'The yaw angle(L+R-) is: ' + yaw
##        r = 'The roll angle(CW+CCW-) is: ' + roll
##        print order
##        print p   # up+down-
##        print y     # left+right-
##        print r    # cw+ccw-
##        ts = 'The transition vector[x,y,z] is:'
##        print ts
##        print tvecs     # x:left+right-
##                        # y:up-down+
##                        # z:away from cam(inch)
##        print '\n'

        # technical output: #,pitch,yaw,roll,x,y,z
        pitch = pitch.replace('[','')
        pitch = pitch.replace(']','')
        yaw = yaw.replace('[','')
        yaw = yaw.replace(']','')
        roll = roll.replace('[','')
        roll = roll.replace(']','')
        pyr = ' '+pitch+' '+yaw+' '+roll
        x = tvecs[0]
        y = tvecs[1]
        z = tvecs[2]
        x = str(x)
        y = str(y)
        z = str(z)
        x = x.replace('[','')
        x = x.replace(']','')
        y = y.replace('[','')
        y = y.replace(']','')
        z = z.replace('[','')
        z = z.replace(']','')

        # project 3D points to image plane
        imgpts, jac = cv2.projectPoints(axis, rvecs, tvecs, mtx, dist)

        img = draw(frame,corners2,imgpts)
    print n,pyr,x,y,z
    cv2.imshow('img',frame)

    if cv2.waitKey(10) & 0xff == 27:
        break
video_capture.release()
cv2.destroyAllWindows()
```

Appendix B

Equipment

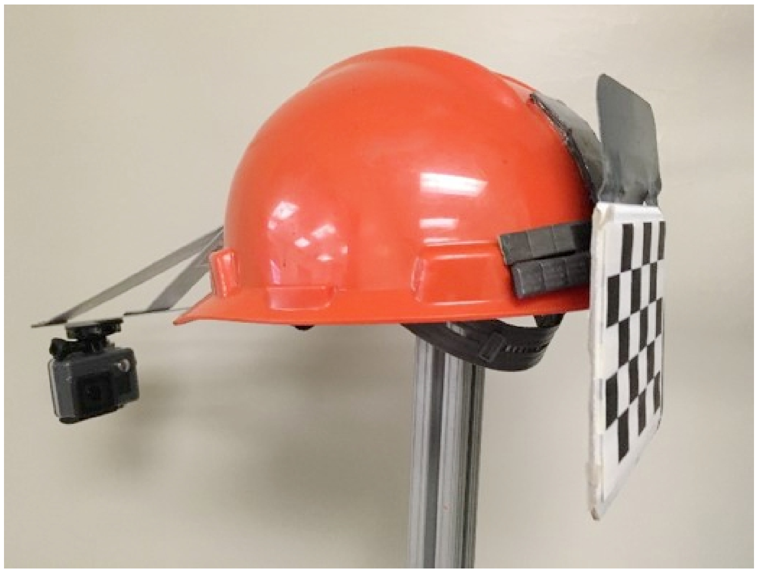


Figure B.1: Hard hat designed for tracking gaze and head motion

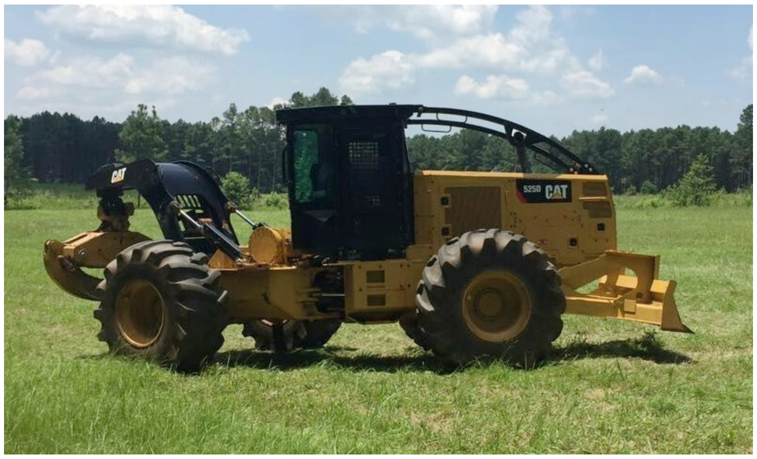


Figure B.2: Caterpillar skidder used for gaze and head motion tracking tests

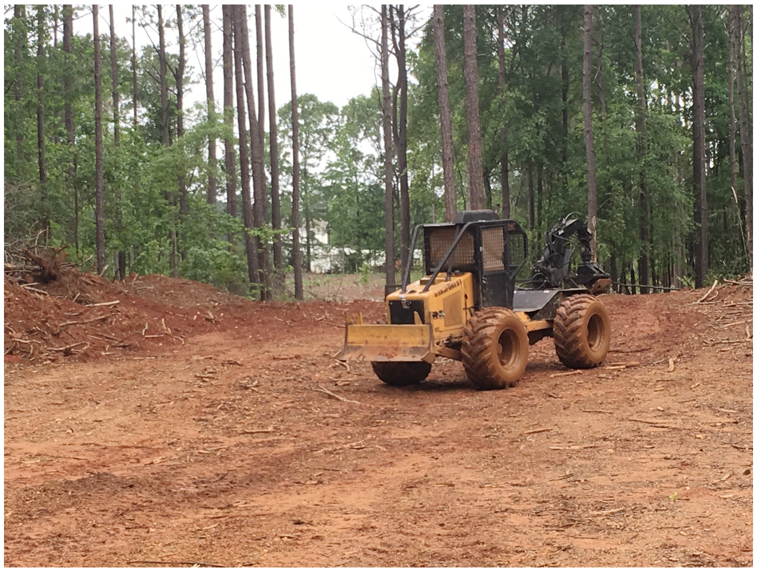


Figure B.3: TurboForest mini skidder with windows covered during driving tests