FOREGROUND REMOVAL IN A MULTI-CAMERA SYSTEM

by

Daniel T. Mortensen

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

Jacob H. Gunther, Ph.D. Major Professor	Todd K. Moon, Ph.D. Committee Member
Thomas H. Fronk, Ph.D. Committee Member	Richard S. Inouye, Ph.D. Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2019

Copyright © Daniel T. Mortensen 2019

All Rights Reserved

ABSTRACT

Foreground Removal In A Multi-Camera System

by

Daniel T. Mortensen, Master of Science
Utah State University, 2019

Major Professor: Jacob H. Gunther, Ph.D.
Department: Electrical and Computer Engineering

Traditionally, whiteboards have been used to brainstorm, teach, and convey ideas with others. However whiteboard content is difficult to transmit and view remotely. To solve this problem, A multi-camera system is developed which can be scaled to broadcast an arbitrarily large writing surface while ignoring distractions. In layman's terms, the objective of this masters thesis is to develop a system which creates a single output image based on the contents of multiple inputs while at the same time removing foreground information. Related Research has been performed previously in image stitching, single image foreground/background object detection and removal, and perspective transforms but to the best of our knowledge this is the first time anyone has attempted to solve this problem using a multi-camera system.

The main components of this problem include bringing the input images into the observation space, identifying foreground material, and replacing the foreground information with its corresponding background estimation. Methods that were employed include homographic transformations and blending techniques for the first component, and statistical tests coupled with classification algorithms for the second.

(53 pages)

PUBLIC ABSTRACT

Foreground Removal In A Multi-Camera System

Daniel T. Mortensen

Traditionally, whiteboards have been used to brainstorm, teach, and convey ideas with others. However distributing whiteboard content remotely can be challenging. To solve this problem, A multi-camera system was developed which can be scaled to broadcast an arbitrarily large writing surface while removing objects not related to the whiteboard content. Related research has been performed previously to combine multiple images together, identify and remove unrelated objects, also referred to as foreground, in a single image and correct for warping differences in camera frames. However, this is the first time anyone has attempted to solve this problem using a multi-camera system.

The main components of this problem include stitching the input images together, identifying foreground material, and replacing the foreground information with the most recent background (desired) information. This problem can be subdivided into two main components: fusing multiple images into one cohesive frame, and detecting/removing foreground objects. for the first component, homographic transformations are used to create a mathematical mapping from the input image to the desired reference frame. Blending techniques are then applied to remove artifacts that remain after the perspective transform. For the second, statistical tests and modeling in conjunction with additional classification algorithms were used.

To those who have supported me through this and all prior endeavors, specifically my parents.

ACKNOWLEDGMENTS

I would be remiss if I failed to give credit to the ECE department at Utah State University, to my many professors, and to friends and family who have supported me as I have pursued higher education. In particular I desire to express my deepest gratitude and heartfelt thanks to Dr. Jacob Gunther who has patiently mentored and tutored me through both undergraduate and masters programs and who has always been a key player in my academic success.

Daniel T. Mortensen

CONTENTS

Page

ABSTRACT iii

PUBLIC ABSTRACT iv

ACKNOWLEDGMENTS vi

LIST OF TABLES ix

LIST OF FIGURES x

1 INTRODUCTION 1
- 1.1 Problem Statement 1
- 1.2 Literature Review 3
  - 1.2.1 Image Fusion 3
  - 1.2.2 Foreground Object Segmentation 4
- 1.3 Paper Overview 5

2 SYSTEM SETUP 6
- 2.1 Hardware 6
  - 2.1.1 Cameras 6
  - 2.1.2 Computer Setup 6
  - 2.1.3 Hardware Triggering 7
- 2.2 Camera Software Settings 7
  - 2.2.1 General Purpose 8
  - 2.2.2 Master 8
  - 2.2.3 Slave 9
- 2.3 Calibration 9
  - 2.3.1 Camera Statistics 10
  - 2.3.2 Perspective Transform 10
  - 2.3.3 Shading Normalization 13

3 SOFTWARE ARCHITECTURE 14
- 3.1 Camera System 15
  - 3.1.1 CameraSystem::Constructor 15
  - 3.1.2 CameraSystem::Run 16
  - 3.1.3 CameraSystem::Close 16
- 3.2 Camera Array 16
  - 3.2.1 CameraArray::Constructor 18
  - 3.2.2 CameraArray::GetImages 18
- 3.3 BLackFlyCamera 19
  - 3.3.1 BlackFlyCamera::Constructor 20
  - 3.3.2 BlackFlyCamera::BeginAcquisition 20

3.3.3 BlackFlyCamera::PopulateImages 20
3.4 Image Processor 22
3.4.1 ImageProcessor::Process 23
3.5 Shading Normalizer 23
3.5.1 ShadingNormalizer::Constructor 24
3.5.2 ShadingNormalizer::ComputeNormalizationMasks 24
3.5.3 ShadingNormalizer::ApplyShadingMasks 24
3.6 Statistical Separator 26
3.6.1 StatisticalSeparator::FuseImages 26

4 ALGORITHMS 27
4.1 Luminance Normalization 27
4.1.1 Theory and Implementation 27
4.1.2 Pseudocode 28
4.1.3 Examples and Results 28
4.2 Foreground Segmentation 29
4.2.1 Theory and Implementation 29
4.2.2 Pseudocode 30
4.2.3 Results 32

5 Discussion and Future Work 33
5.1 Discussion 33
5.2 Future Work 33

REFERENCES 35

6 CURRICULUM VITAE 37

LIST OF TABLES

Table Page

2.1 General Purpose Camera Settings 8
2.2 Master Camera Settings 9
2.3 Slave Camera Settings 9

LIST OF FIGURES

Figure Page

2.1 Camera Array Setup 7

2.2 Camera Hardware Setup 8

2.3 Homography Calibration 12

2.4 Comparison of pre and post-normalized images 13

3.1 CameraSystem Dependency Structure 15

3.2 CameraSystem::Constructor Functional Diagram 16

3.3 CameraSystem::Run Functional Diagram 17

3.4 CameraSystem::Close Functional Diagram 17

3.5 CameraArray Structural Diagram 18

3.6 CameraArray::Constructor Functional Diagram 19

3.7 CameraArray::GetImages 19

3.8 BlackFlyCamera::Constructor 20

3.9 BlackFlyCamera::BeginAcquisition 20

3.10 BlackFlyCamera::PopulateImages 21

3.11 ImageProcessor Dependency Structure 22

3.12 ImageProcessor::Process 23

3.13 ShadingNormalizer::Constructor 24

3.14 ShadingNormalizer::ComputeNormalizationMasks 25

3.15 ShadingNormalizer::ApplyShadingMasks 25

3.16 StatisticalSeparator::FuseImages 26

4.1 Comparison of pre and post-normalized images 29

4.2 Initial Foreground Estimate 32

4.3 Post-Processed Results 32

CHAPTER 1

INTRODUCTION

Image fusion has long been a topic of study in digital signal processing with applications ranging from panoramic photographs [1], [2] and Computer-Generated Imagery (CGI) to GPS denied navigation. Image fusion is a popular choice when developing methods for capturing large fields of view where alternatives would include expensive optical systems [3]. Foreground/background segmentation is a separate area of study in image processing which focuses on identifying, separating, and tracking objects in images and video feed [4]. Current work in foreground/background segmentation focuses primarily on single camera systems [5], [6], [7]. There are also a small number of published works which utilize multiple cameras to accomplish the same [4]. A third relevant area of research includes foreground replacement, also referred to as Video in-painting which has traditionally been applied to single camera systems [8].

There has been no work done on removing foreground objects using a multi-camera system. This would require that foreground objects be identified, and replaced with an estimated background canvas. This is an important topic of research because of its potential uses in multi-camera systems that are used for communication in both business and academic settings. This thesis aims to solve this problem while emphasizing one particular application: distance education.

1.1 Problem Statement

Traditionally, curriculum has been designed with one central purpose: to help students develop a desired skillset. One of the earliest accounts of public education can be dated back to 605 AD during the Sui Dynasty. In this time period, the Chinese government instituted the Imperial Examination which served as a medium through which candidates for government positions were identified [9].

Education and examinations were also instituted by England in 1853. During this time, William Gladstone commissioned Stafford Northcote and Charles Trevelyan to research the organization of the English Civil Service. Their report (The Northcot-Trevelyan Report of 1854) gave four areas of instruction: that recruitment should be based on the results of a standardized test, that new hires should have a well rounded education so as to be transferable between departments, that there would be a hierarchy in place, and that promotion would be awarded through achievement [10].

It is plain to see that the original motivations for education and exams remain in place today. Students attend school, study, and receive reports that detail their competency. These reports are then used in hiring and recruiting to determine readiness for employment. Because of the impact education has on one's quality of life, the demand for such has increased dramatically in recent years, especially in technical fields. The National Center for Educational Statistics mentions that Fall enrollment in degree-granting post-secondary institutions increased by 21 percent between 1994 and 2004 and later that Between 2004 and 2014, enrollment increased 17 percent, from 17.3 million to 20.2 million. [11] In order to meet this increased demand, many classes may need to be given through non-traditional methods. In the past, traditional classes provided sufficient resources for those enrolled. Unfortunately, contemporary situations vary much from those of the past and many students require flexibility that is best accommodated by distance education [12].

Fortunately, even though demand for online courses has increased, technology has become more affordable. In fact, many devices such as cameras, microphones, and websites are already used in remote classes. However, one item still remains missing from a remote classrooms —a whiteboard. When broadcasting lectures, cameras are placed either towards the back, or in the front. When in back, remote students are unable to see material written on whiteboards. When placed in front, they obstruct the view of students who are physically present, and many times fail to capture the entire board. Current research has solved many problems associated with this topic. There are systems that stream live video of single whiteboards and others that correct for angled images and would be ideal for ceiling mounts [13]. These systems contain many pieces of the solution, but fail to solve this problem, nor do they remove distracting foreground data from the observation space. In short, work has been done to fuse images together. The questions we desire to answer in this masters thesis are: How can we remove foreground information from a live video feed? How can this be effectively implemented on hardware? How should the software be implemented such that the system operates in real-time?

1.2 Literature Review

This chapter contains information regarding relevant work. There are two areas of research which will be examined: Image Fusion, and Foreground Object Segmentation. Each has been extensively explored in previous work and this review is not meant to be all inclusive, rather the purpose of this review is to examine pertinent work and help the reader understand how the proposed thesis will contribute to its field.

1.2.1 Image Fusion

Image fusion is the process by which two or more images are fitted together by some algorithm [3]. Fusion techniques can be placed in one of three categories: pixel fusion, feature fusion, and decision fusion [14]. For the purposes of this paper, we will consider primarily methods for pixel fusion. When examining techniques for pixel fusion, there are several requirements that define a successful algorithm. They are pattern preservation, external noise and artifact omission, or that features not included in the input images are not introduced during processing, and object shift and rotation invariance. This success criteria essentially means that the algorithms must not: exclude information given in the input images, insert artifacts not originally present, or depend on the orientation/location of any one particular object in the scene [14].

Pixel level techniques can further be broken down into spatial, and spectral algorithms. Spacial algorithms depend on information contained in the spacial domain such as pattern recognition, edge detection, or points of interest. Spectral fusion methods compute a 2D Fourier transform, process the image in the frequency domain, and then compute an inverse

Fourier transform to retrieve the final image [15]. Applications of Fourier based image fusion include medical imaging, remote sensing, computer vision, and robotics [16] [17]. Popular pixel level fusion techniques include pixel based laplacian and wavelet fusion [14], homographic mappings [18], [19], [20], [21], [22], [1], [2], and gradient based multiresolution algorithms [23].

1.2.2 Foreground Object Segmentation

Traditional Foreground Object segmentation begins with the segmentation of still images in a video sequence and can combine temporal information to relate objects between frames. When segmenting individual video frames, many popular methods include the use of superpixels. A superpixel is a number of adjacent pixels which are grouped according to a predefined criteria. Some examples include semantic segmentation [24], geometric context separation [6], and support vector machines [7]. In general however, the most commonly used methods are a variant of either graph cutting or watershed techniques, which result in segmentation cuts consistent with object fragments [25].

Graph cutting techniques focus on representing any given image as a graph where each pixel or superpixel is a node, and their codependences are vertices. When "cutting" a graph, we desire to find the path that minimizes the cost of cutting, such as Dijkstra's algorithm, or a minimum spanning tree. The difficulty with these types of algorithms, is the need to know marker locations, where markers represent vertex intersections. This continues to be an area of research although work has been done to identify markers, or points of interest with algorithms such as through edge detection on multiresolution images [26]. Alternative solutions include creating superpixels from individual frames and combining them to more accurately estimate marker locations [5] which can then be paired using homographic remapping [27].

Segmentation in videos is much the same as that of single images except that temporal information is also available for exploitation. Temporal information has been used in the past to clarify ambiguities left behind by traditional methods and allow observations to travel from frame to frame [4]. Additional cameras have also been used for the same purpose. These viewpoints keep foreground objects from drifting and being absorbed into the background over time.

1.3 Paper Overview

In this masters thesis, segmentation was achieved by first mapping input images into a viewing space through homographic transformations, identifying foreground material by way of statistics tests, and replacing the foreground information with its corresponding background estimation. The remainder of this masters thesis is organized as follows. Chapter 2 gives an explanation of calibration techniques and system setup. Chapter 3 shows software architecture. Chapter 4 explains the workings of new algorithms that were developed to solve this particular problem. Chapter 5 discusses results and future work.

CHAPTER 2

SYSTEM SETUP

2.1 Hardware

This section discusses the hardware used throughout the duration of the thesis as well as their setup, configuration, and use. The hardware component was necessary for the success of this research topic because it allows for real-time demonstration of the algorithms given in Chapter 4. It required a significant time investment as custom hardware was required to operate hardware triggers and various software settings required configuration.

2.1.1 Cameras

The cameras used were Blackfly USB 3 Computer Vision Cameras from Point Gray. These cameras have a frame rate of up to 40 frames per second with image dimensions 1920 x 1200. Additionally, each camera utilizes a global shutter and is equipped with a Tamron CCTV lens. In this series of experiments, four cameras were used, each mounted on top of a tripod (See Figure ??). Because the cameras operated using USB 3.1, special cabling needed to be purchased from the camera manufacturer which could operate over larger distances of up to 5 meters.

2.1.2 Computer Setup

In order to accommodate the large data output, a specialized PCIe card was installed which provided 4 additional USB 3.1 hardware interfaces on independent buses. Usually, USB interfaces will share data buses to conserve resources, however because each camera is capable of saturating a single bus it became necessary to equip each camera with an independent bus.

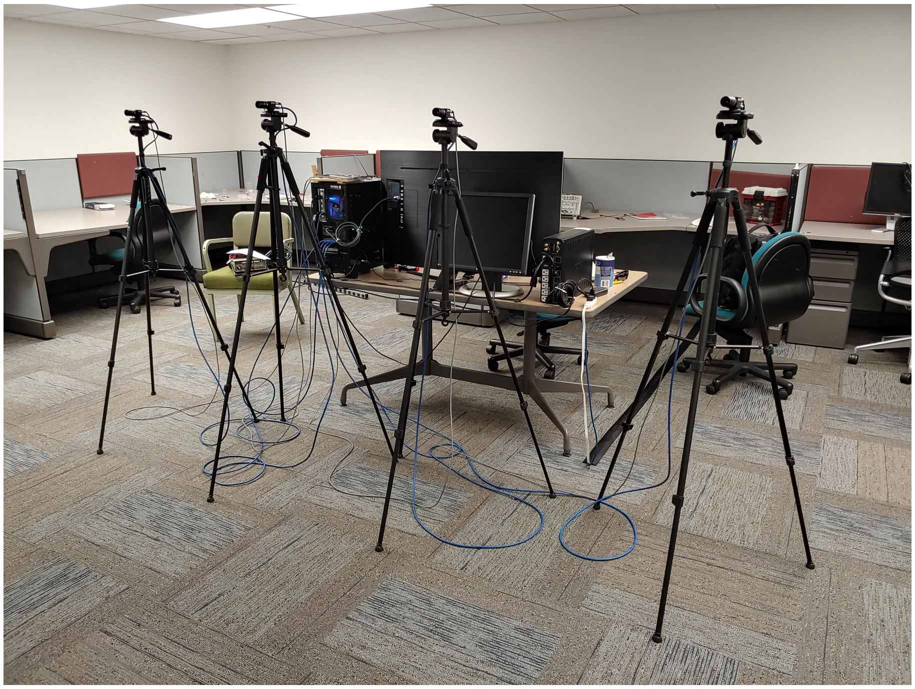


Fig. 2.1: Camera Array Setup

2.1.3 Hardware Triggering

The algorithms used in this masters thesis presuppose that each image be taken at the same time. In order to guarantee this condition, a master/slave configuration was used where a capture signal from the master was transmitted to each slave device, causing each camera to trigger simultaneously. The slave devices were configured to trigger on a falling edge. The master made us of a pull-up resistor to maintain the input line high, and brought the line low to signal a frame capture. A diagram of the hardware setup can be seen in Figure 2.2.

2.2 Camera Software Settings

There are three sets of software configuration settings used to initialize the cameras. They are the General Purpose, Master, and Slave settings. The General Purpose settings apply to all cameras, whereas master and slave settings are used to configure their respective device types.

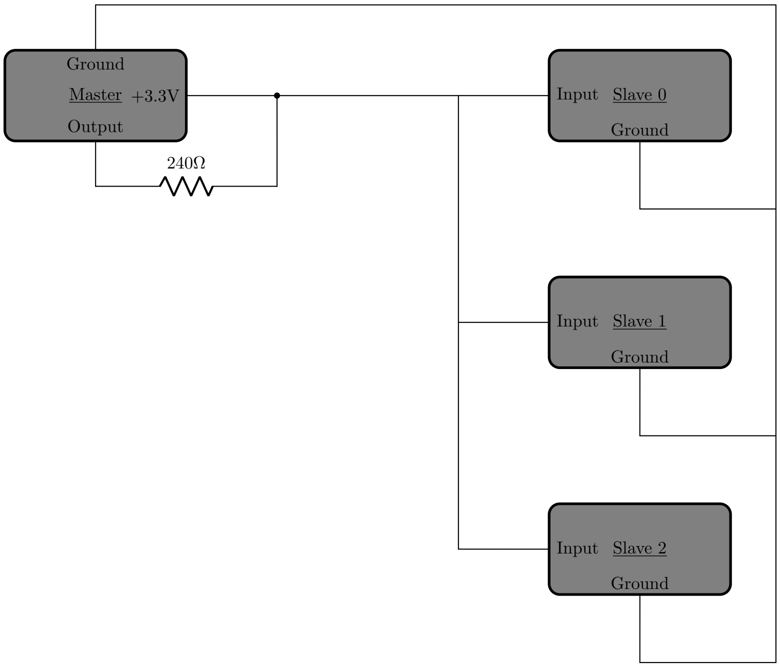


Fig. 2.2: Camera Hardware Setup

Table 2.1: General Purpose Camera Settings

General Purpose Camera Settings		
Camera Function	Chosen Setting	Additional Information
Pixel Format	RGB8	Will be converted to BGR before used in algorithms
Acquisition Mode	Continuous	Continue Capturing until program exits
Buffer Mode	Newest Only	Ensures that each frame set is paired correctly

2.2.1 General Purpose

The General Purpose settings are configurations that are contained by all cameras regardless of type. Additional settings are then applied based on if the camera is a master or slave. These general purpose settings are given in detail in Table 2.1.

2.2.2 Master

The master camera is configured differently then it's slave counterparts. Both include the General Purpose Camera Settings, but are then configured to be either a master or a slave. The master settings are shown in Table 2.2.

Table 2.2: Master Camera Settings

Master Camera Settings		
Camera Function	Chosen Setting	Additional Information
Line Selector	Zero	Determines the output line
V3	High	Pulls the line high when not triggered

Table 2.3: Slave Camera Settings

Slave Camera Settings		
Camera Function	Chosen Setting	Additional Information
Trigger Mode	Off	Trigger mode must be off before changes can be made
Trigger Source	Hardware	Cameras can trigger from either a software or hardware trigger
Trigger Activation	Falling Edge	Set camera to trigger when a falling edge is detected on the input line
Trigger Mode	On	Activate trigger mode now that settings are in place
Exposure Mode	Timed	Maintains a set time for the exposure on each image
Trigger Overlap	Readout	exports the previous image while the current is being captured

2.2.3 Slave

One primary difference between the slave and master cameras, is the use of a hardware trigger. Every time the master camera captures an image, it outputs a signal to one of the GPIO ports which causes the slave devices to capture as well. Each slave device must therefore be initialized with a hardware trigger. The settings given in table 2.3 shows the necessary Steps to configure a slave device.

2.3 Calibration

Before the camera array system can be utilized, it must first be calibrated. The purpose of these calibration steps is to gather information about the target scene which is used by system algorithms to generate desirable output. Three principle areas of calibration are needed before image processing can begin. They are camera statistics, perspective transforms, and shading normalization.

2.3.1 Camera Statistics

The camera statistics are used to determine variance or uncertainty in the given pixel values, as these are subject to defects that are introduced by imperfections in the sensor/optics and by the process by which data is quantized and converted from an analog signal to a digital representation.

In Algorithm 3, it is assumed that the pixel data can be modeled as

$$y_{i,j} = \nu_{i,j} + p_{i,j}, \tag{2.1}$$

where $\nu_{i,j}$ is Gaussian with mean μ and variance σ^2. The purpose of this calibration section is to estimate this mean and variance. We know that the maximum likelihood estimators for the mean and variance for Gaussian distributions are

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \text{and} \qquad \hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N} \left(x_i - \hat{\mu}\right)^2 . \tag{2.2}$$

For each camera, images were collected with the lens cap on. This gives a true pixel value of 0, or black. From this the mean and variance can be calculated. It should also be noted that changes in variance due to autocorrections made by the camera such as automatic gain control were not evaluated and that future improvements might include parameter calibration in front of a white sheet or something of the sort.

2.3.2 Perspective Transform

The perspective transform maps the images from their respective input viewing spaces into a common reference frame. In order to do this, a model must first be determined by which this mapping can occur. The perspective transform calibration creates this mapping.

This section focuses on calibrating mappings from each input camera perspective to a final viewing space. For this, a baseline camera is selected arbitrarily. One by one, points of interest are found for input images taken from each respective camera. Pairs are then found between the baseline and each additional image. The SIFT algorithm was used for the initial point generation and pairing. Unfortunately, this algorithm isn't fool proof and RANSAC was used to remove mismatched points. Once this was done, a homographic transformation can be found as long as at least 4 pairs of points were found. A homographic transformation (or homography) is an affine transformation between two viewing perspectives such that

$$\left\{A, x \middle| Ax_i \rightarrow d_i \quad \forall x \in X, d \in D\right\}, \tag{2.3}$$

where X represents the set of points belonging to the input perspective, D represents the set containing all points in the transformed viewing space, and A is a 3x3 matrix. For dimensions to work, the x,y coordinates are augmented s.t.

$$x_i = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{and} \quad d_i = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ c \end{bmatrix}. \tag{2.4}$$

The final operation is to normalize $\tilde{x}$ and $\tilde{y}$ by

$$x_{\text{final}} = \tilde{x}/c \quad \text{and} \quad y_{\text{final}} = \tilde{y}/c. \tag{2.5}$$

Since the A matrix can be described as

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \tag{2.6}$$

solving for the coefficients of A becomes a set of equations which are easily solvable given a sufficient number of input and output point pairs. A high level diagram describing the algorithmic flow for the homography calibration can be seen in Figure 2.3.

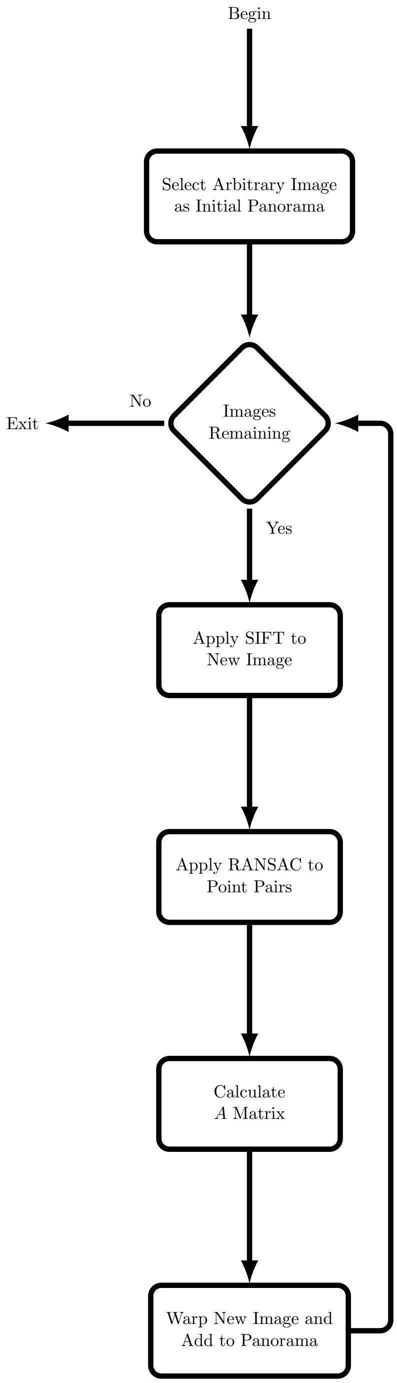
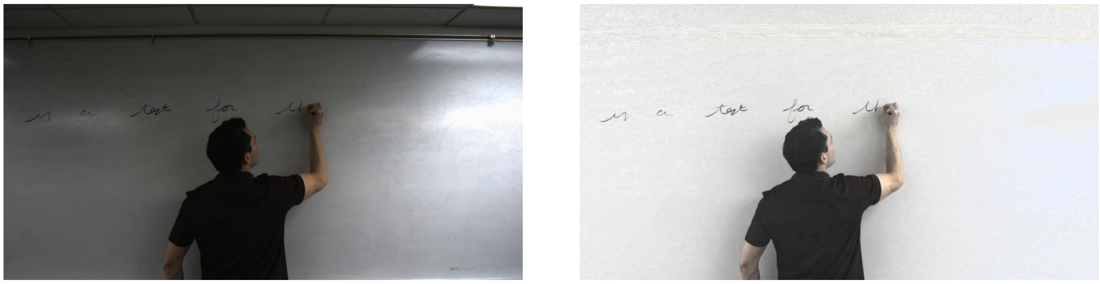

Begin
Select Arbitrary Image as Initial Panorama
Images Remaining
No
Exit
Yes
Apply SIFT to New Image
Apply RANSAC to Point Pairs
Calculate A Matrix
Warp New Image and Add to Panorama

Fig. 2.3: Homography Calibration



(a) Original Image (b) Postprocessed Image

Fig. 2.4: Comparison of pre and post-normalized images

2.3.3 Shading Normalization

Finally, because each image is taken from a different perspective, the lighting is inherently different. The calibration for shading normalization is used to gather data about the shading differences so that these effects can be mitigated. Figure 2.4 shows an example of an image where shading effects have been suppressed.

The shading normalization calibration method implements Algorithm 1 and stores the resulting normalization masks in memory for future use. This allows the system to be calibrated once and as long as lighting remains the same then no further calibration is needed. Further work would need to be done to determine continuous calibration for lighting if the system were to be used in more dynamic settings.

CHAPTER 3

SOFTWARE ARCHITECTURE

This chapter discusses overall software architecture. In this chapter there are several references to interfaces. This can mean one of two things: first, an abstract class which enforces predetermined functionality on all child classes. The second refers to objects that can be requested from objects that are used to access their processing results, these can also be referred to as external runtime parameters.

Software for this project was designed in an object oriented fashion and each class will be described in two ways: through a structure diagram which denotes dependencies and inheritance properties and through functional diagrams which show program flow and high level functionality. Functional diagrams will be given starting with public functions, followed by their corresponding subfunctions.

Dependencies and their use are denoted by green blocks, with an underlined section indicating from which dependency this functionality is derived. Subfunctions are colored white, and abstract classes are pink. Straight arrows are used to denote flow direction with double boxes indicating pipelined processes. Additionally, Boxes with a blue interior are used to indicate Cuda acceleration. In Structural diagrams, there are two types of arrows. Solid arrow heads indicate dependency and hollow arrow heads indicate inheritance.

This system is organized in an object oriented approach with special emphasis on maintainability and the ability to easily decouple and replace dependencies. Each main system component relies on a factory whose purpose is to provide objects upon which the parent class is dependent. The return types of these dependencies are abstract classes from which the original class will inherit. This abstracts the original object type away from the rest of the system and should the dependency need to be modified or changed, as long as it fulfills the requirements set by the abstract class from which it inherits, the rest of the system will function with no additional changes.

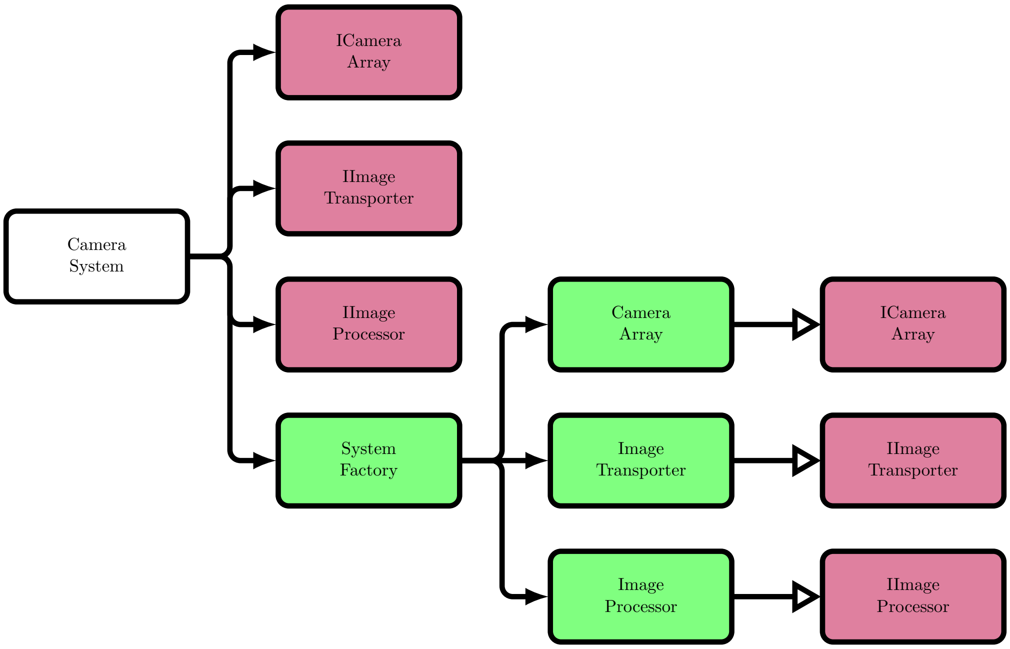


Fig. 3.1: CameraSystem Dependency Structure

3.1 Camera System

The Camera System class accomplished three principle objectives. These objectives are calibration and setup, implementation of runtime functions, and cleanup. The first is handled in the constructor, the second in run function, and the third in the close function. As seen in Figure 3.1, the camera system is dependent on three primary objects: the camera array, the image transporter, and the image processor. The majority of the work done in this masters thesis focuses on the image processor. A system factory was also used to allow for maintainability.

3.1.1 CameraSystem::Constructor

The camera system constructor is primarily focused on calibrating the system and defining parameters. The first object to be initialized is the camera array (more details on what this entails will be given later in the chapter). The image processor is then calibrated which includes defining homographies, receiving statistical information, and calculating shading normalization parameters. During this process, The user is required to adjust camera position and erase the whiteboard. A flowchart is given in Figure 3.2

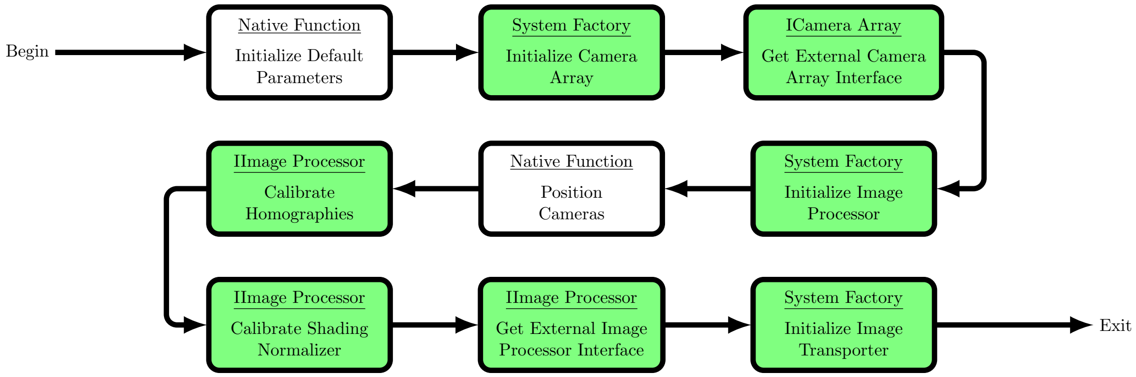


Fig. 3.2: CameraSystem::Constructor Functional Diagram

3.1.2 CameraSystem::Run

The purpose of this function is to retrieve incoming frames, process them using parameters obtained during the calibration phase of setup, and export them to an external source. Figure 3.3 shows the software layout.

3.1.3 CameraSystem::Close

This function is responsible for closing down the system, exiting threads, etc. Most of these functions are handled within dependencies and will be discussed later in the chapter. The only change that has to be made in the current scope is to set the ”terminate” variable to ”true” and wait for all camera threads to join, as see in Figure 3.4.

3.2 Camera Array

The CameraArray object is responsible for managing all single cameras, retrieving images in a timely manner, and providing a functional external interface that allows other objects to access camera data. These cameras are operated in separate threads so as not to block and utilize a buffering system so that external objects may access data while the next image set is being loaded into memory. This object depends on its own factory, which is used to allow for maintainable code and for the decoupling of dependencies. It is also dependent on the abstract ICamera object (which is generated by the CameraArrayFactory). The final dependency is the abstract ICameraArray class, from which this class inherits.

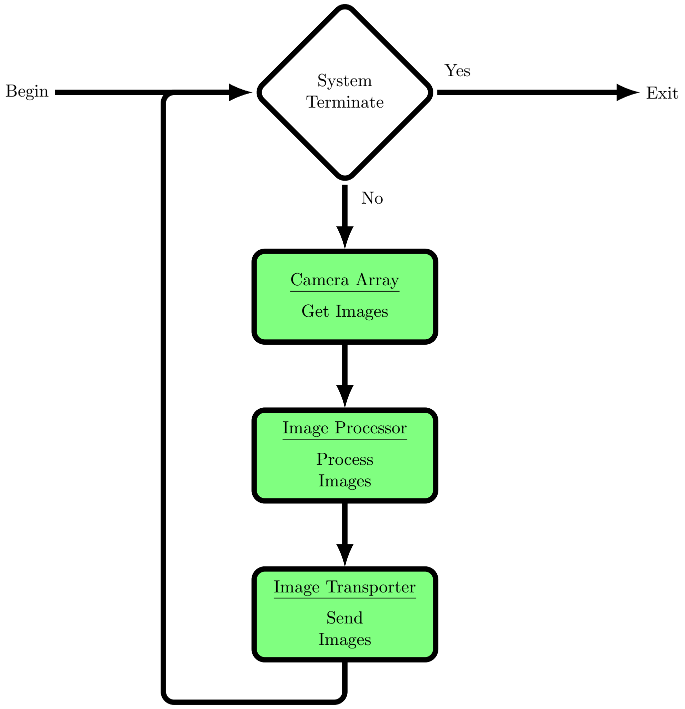


Fig. 3.3: CameraSystem::Run Functional Diagram

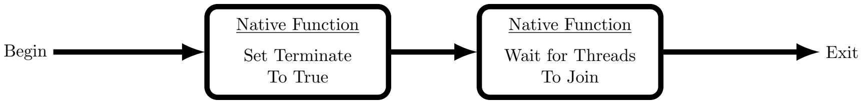


Fig. 3.4: CameraSystem::Close Functional Diagram

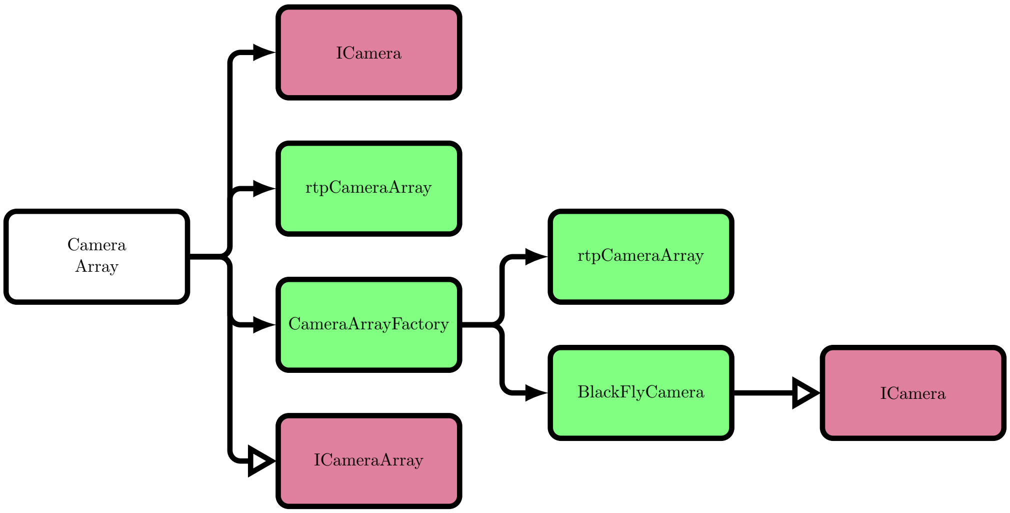


Fig. 3.5: CameraArray Structural Diagram

This inheritance ensures that the CameraArray class will function as planned in the overall system. A diagram outlining these dependency relationships can be found in Figure 3.5.

3.2.1 CameraArray::Constructor

The Primary focus of the CameraArray::Constructor is to initialize all camera objects, instantiate member variables, start all cameras in their own separate threads, and populate the image buffer with the first set of images in preparation for homography calibration. Dependencies are initialized by the CameraArrayFactory, and master/slave relationships are defined for each camera. Because of the hardware trigger configuration used to synchronize cameras (to be discussed later in the chapter), the master must be started first, followed by each slave. For more detail, see figure 3.6.

3.2.2 CameraArray::GetImages

The CameraArray contains an image buffer which contains both an outbound and inbound location. The outbound is available to a consumer while the inbound is being loaded with the next set of images. They are kept separate to avoid race conditions and the pointers rotated once CameraArray::GetImages is called provided that all images have been successfully loaded into the inbound location. Figure 3.7 shows the proper flow for

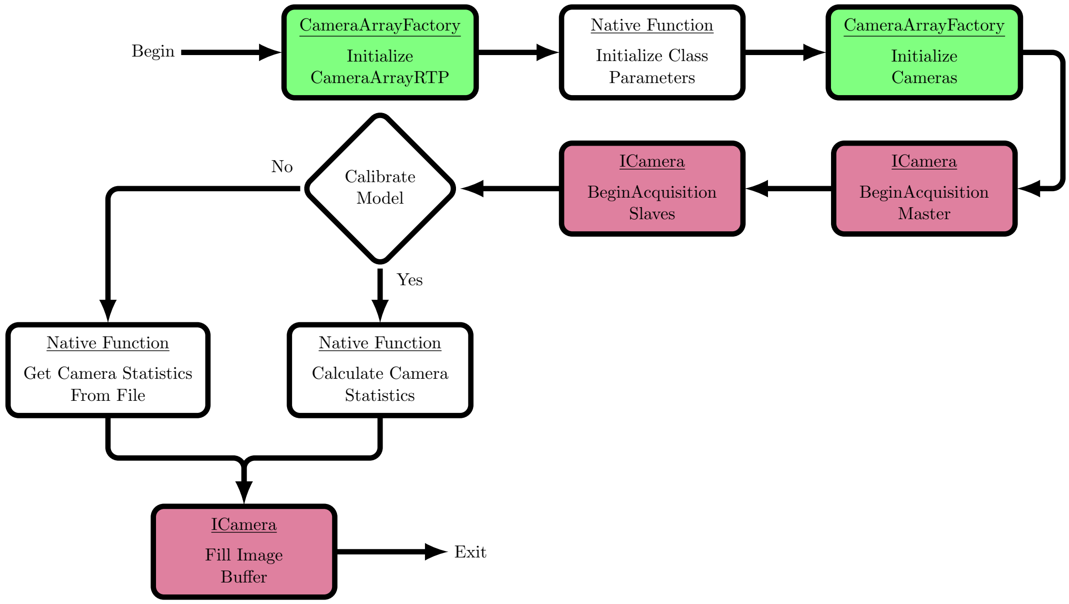


Fig. 3.6: CameraArray::Constructor Functional Diagram

this function.

3.3 BLackFlyCamera

The BlackFlyCamera object is responsible for retrieving images from a single camera, storing them in memory, and providing that memory to the CameraArray object when requested. The BlackFlyCamera operates in a separate thread and handles the image acquisition asynchronously. This allows the cameras to capture images for the next cycle

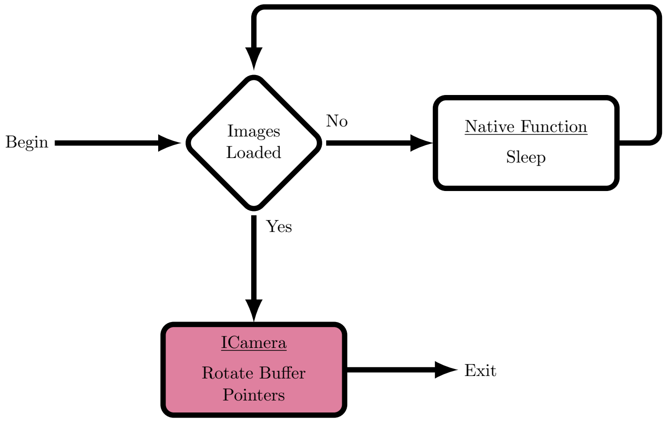


Fig. 3.7: CameraArray::GetImages

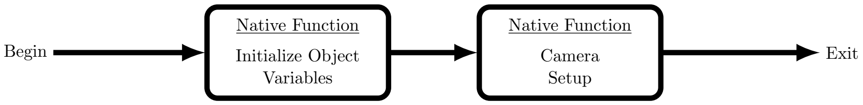


Fig. 3.8: BlackFlyCamera::Constructor

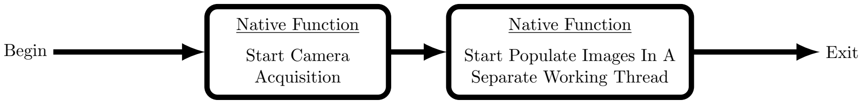


Fig. 3.9: BlackFlyCamera::BeginAcquisition

while the current images are being processed. No Structure diagram is provided for this model as it only depends on itself and inherits from it's interface, ICamera. The remainder of this section will focus on BlackFlyCamera::Constructor, BlackFlyCamera::PopulateImage, and BlackFlyCamera::BeginAcquisition functions.

3.3.1 BlackFlyCamera::Constructor

The constructor for the BlackFlyCamera consists of initializing object variables passed in from the CameraArray, and setting up the cameras themselves. Camera setup and calibration is covered in Chapter 2, and will not be discussed in detail at this time. Figure 3.8 shows the functional layout of the BlackFlyCamera Constructor.

3.3.2 BlackFlyCamera::BeginAcquisition

The BeginAcquisition function initializes the separate thread in which images will be retrieved. It also passes instructions through the camera API to begin capturing images. A functional diagram is provided in Figure 3.9.

3.3.3 BlackFlyCamera::PopulateImages

This function captures images from the camera hardware, stores them in the output buffer, and notifies the main thread that images are ready. This thread then sleeps until the image buffer is empty once again. This behavior continues until the program terminates where this thread is exited. Figure 3.10 shows the functional layout.

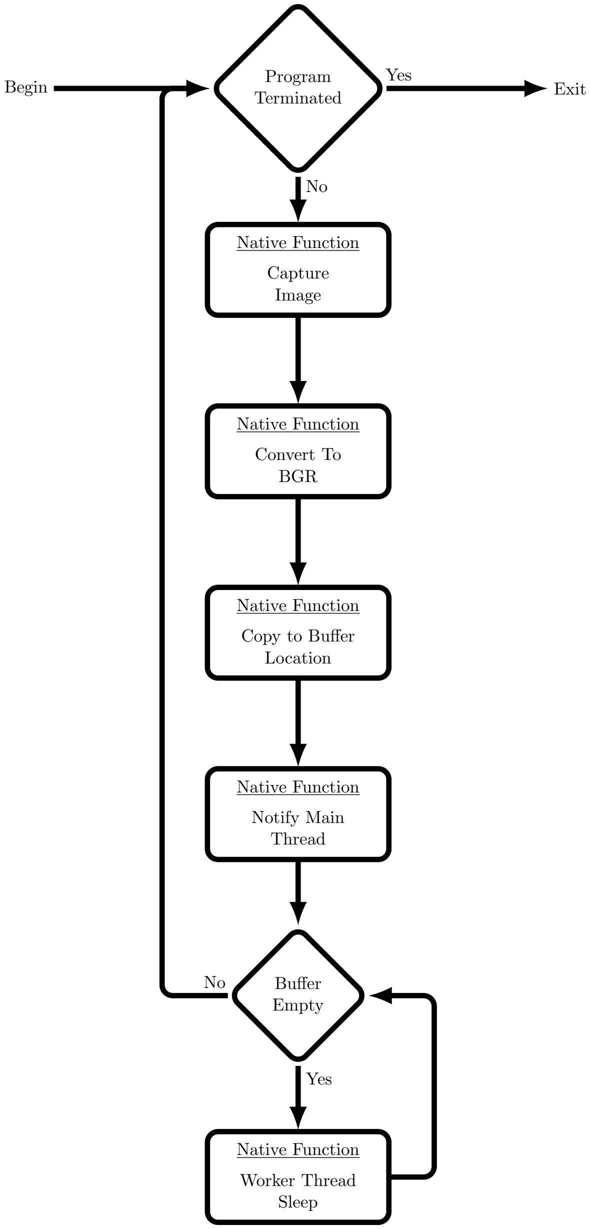


Fig. 3.10: BlackFlyCamera::PopulateImages

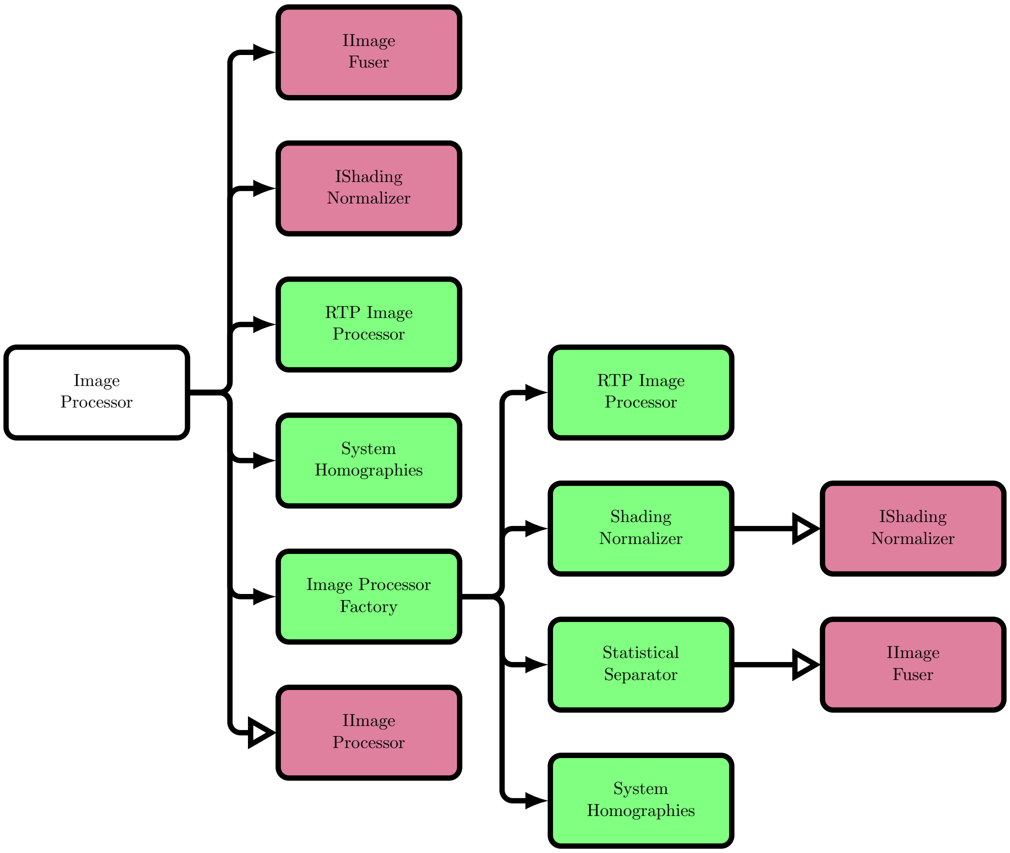


Fig. 3.11: ImageProcessor Dependency Structure

3.4 Image Processor

The image processor object receives images from an ICameraArray object, processes them, and provides a processed result to an IImageTransporter type object. The processing flow is as follows: Receive images, Normalize Shading, perspective transform, blend and teacher removal, export final image to IImageTransporter. The dependencies are shown in Figure 3.11. The constructor primarily allocates the appropriate memory for intermediary processing steps and initializes member variables where the destructor only deallocates the same memory. Because of their simplicity, they will not be discussed going forward. It is also worthy to note that there are functions for calibrating shading normalization parameters, but because this functionality is discussed in more detail in Chapter 4, it will also be excluded from the current discussion. The remainder of this section will discuss functional details related to the ImageProcessor::Process function.

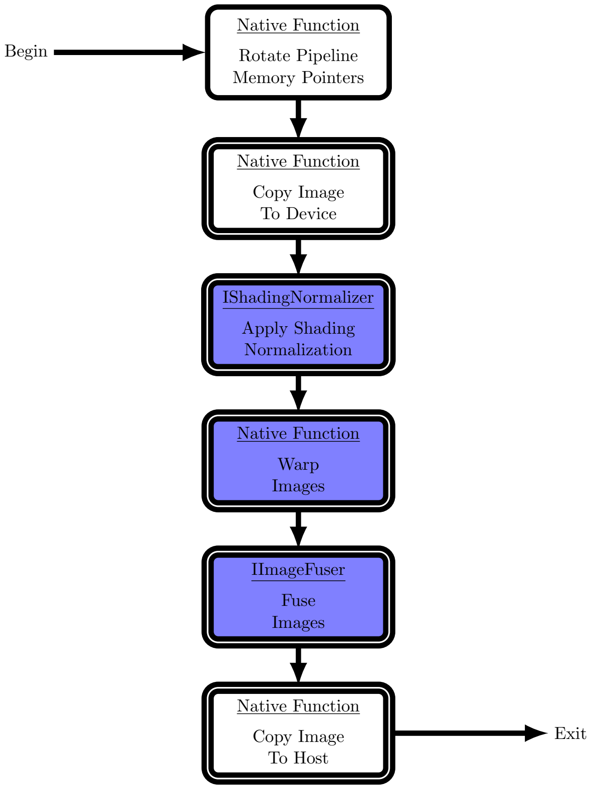


Fig. 3.12: ImageProcessor::Process

3.4.1 ImageProcessor::Process

The objective of the ImageProcessor:Process function is to take a set of input images, normalize the shading, warp them into the observation space, remove foreground information, and return a panorama of only background imagery. All image processing techniques were implemented in Cuda and were performed on a GPU device. Whenever Device is mentioned, it refers to the GPU and Host refers to the development platform. Furthermore, a pipeline was implemented such that each step could be performed in parallel with data from different time steps. The algorithm pipeline is shown in Figure 3.12.

3.5 Shading Normalizer

The Shading normalizer takes images and performs Algorithm 2 on each pixel. This normalizes the image luminance, which helps when comparing images from different cameras/perspectives. The Shading Normalizer has three primary functions: the construc-

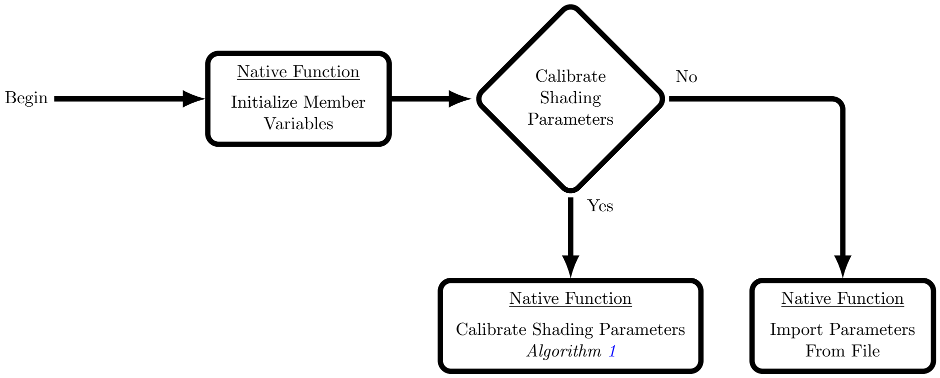

Begin
Native Function
Initialize Member Variables
Calibrate Shading Parameters
No
Yes
Native Function
Calibrate Shading Parameters
Algorithm 1
Native Function
Import Parameters From File

Fig. 3.13: ShadingNormalizer::Constructor

tor, calibration function (Algorithm 1), and ShadingNormalizer::ApplyShadingMasks (Algorithm 2). Because the shading normalizer has no dependencies, a structural diagram would be trivial and as such is not provided.

3.5.1 ShadingNormalizer::Constructor

The constructor initializes member variables, and then either initializes shading normalization parameters from values stored in a file, or calibrates new parameters. A functional diagram is provided in Figure 3.13.

3.5.2 ShadingNormalizer::ComputeNormalizationMasks

This function is used during calibration time to compute the normalization masks used in Algorithm 2 and follows the pseudocode given in Algorithm 1. First the optical data is copied from the host to the device, Algorithm 1 is applied, after which results are copied from the device back to the host (See Figure 3.14).

3.5.3 ShadingNormalizer::ApplyShadingMasks

The ShadingNormalizer::ApplyShadingMasks function follows the instructions given in Algorithm 2. The images are converted to the YCbCr color frame, the first channel (or luminance) is normalized, after which the image are converted back to BGR. A functional diagram is provided in Figure 3.15.

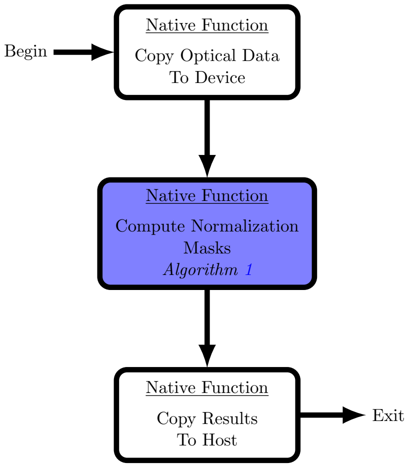


Fig. 3.14: ShadingNormalizer::ComputeNormalizationMasks

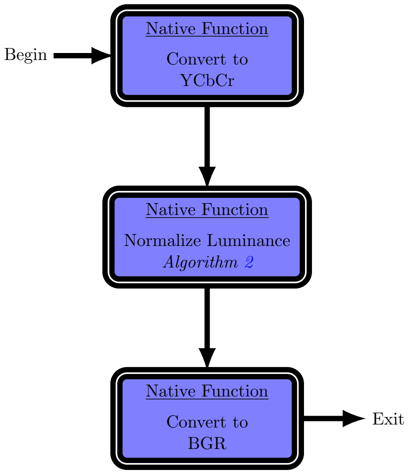


Fig. 3.15: ShadingNormalizer::ApplyShadingMasks

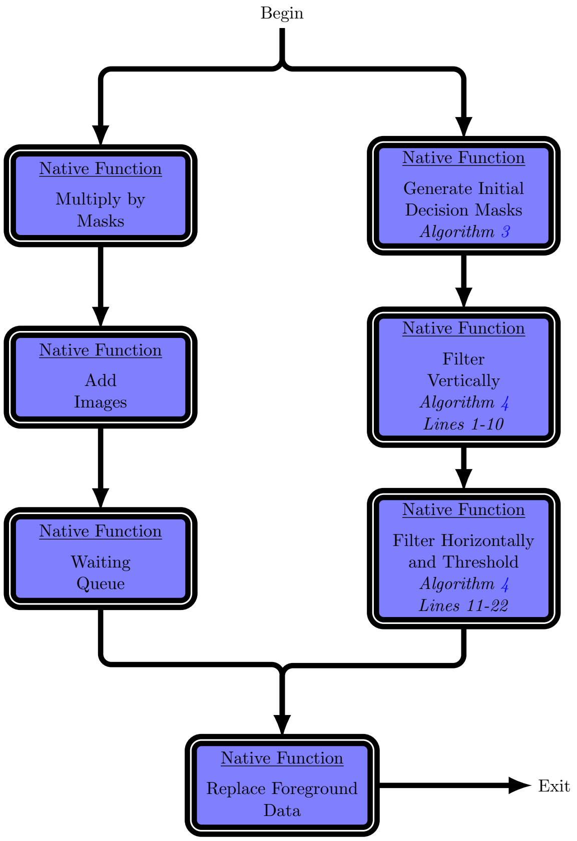


Fig. 3.16: StatisticalSeparator::FuseImages

3.6 Statistical Separator

The Statistical Separator has 2 tasks: fuse input data into one cohesive image, and remove foreground information from the output panorama. There is one principle function which implements this functionality: the StatisticalSeparator::FuseImages function.

3.6.1 StatisticalSeparator::FuseImages

The StatisticalSeparator::FuseImages function has three principle components: fuse and blend images, detect foreground regions, and replace foreground pixels with background data from prior time steps. These are pipelined in such a way as to allow for each component to be done in parallel on the GPU.

CHAPTER 4

ALGORITHMS

This chapter covers the implementation of custom algorithms used during the course of this masters thesis. Additional methods such as SIFT, RANSAC, and homography generative algorithms were used, but will not be discussed in this chapter as adequate information and descriptions are readily available.

4.1 Luminance Normalization

The purpose of this algorithm is to mitigate differences in image perspectives due to shading. The calibration portion, located in Algorithm 1, generates a mask that normalizes the luminance between input and template data, denoted as X, and D respectively. This occurs only once and is stored for use at runtime. The runtime portion, located in Algorithm 2, takes an image as expressed in the YCbCr color space, and applies the normalization coefficients to the luminance channel. This is done to mitigate the effects of shading when comparing the differences between orthorectified images in Algorithm 3.

4.1.1 Theory and Implementation

For the calibration portion of the algorithm, we start with a base or ”true” image which will be used as a standard to which the input image will be mapped. The objective is to find a set of coefficients a such that

$$\big\{a, x \big| a_i x_i \rightarrow d_i, x \in X_i, d_i \in D\big\} \tag{4.1}$$

where A and D represent the set of all points in the input and template image respectively. The solution is to point-wise divide each pixel from the template image by the value in the input image.

4.1.2 Pseudocode

The Pseudocode for the calibration algorithm can be found in Algorithm 1 and the runtime normalization portion is described in Algorithm 2 where X, D, A, Y, and Z represent the input data as expressed in the YCbCr colorspace, template data as expressed in the YCbCr colorspace, Luminance normalization coefficients, input data as expressed in the RGB colorspace, and output data as expressed in the RGB colorspace.

Algorithm 1 Luminance Normalization Calibration

```
Require: X, D
for i=1:number of rows do
    for j=1:number of columns do
        $a_{i,j} = d_{i,j}/x_{i,j}$
    end for
end for
return A
```

Algorithm 2 Luminance Normalization Runtime

```
Require: Y, A
Convert RGB image to YCbCr colorspace: $Y \rightarrow X$
for i=1:number of rows do
    for j=1:number of columns do
        $d_{i,j} = x_{i,j}a_{i,j}$
    end for
end for
Convert normalized image to RGB colorspace: $D \rightarrow Z$
return Z
```

4.1.3 Examples and Results

When applied, shading effects are greatly reduced as seen in Figure 4.1. For this demonstration, a white image was used to calibrate the normalization parameters which greatly reduced shading and glare artifacts.

(a) Original Image (b) Postprocessed Image

Fig. 4.1: Comparison of pre and post-normalized images

4.2 Foreground Segmentation

The main purpose of this research is to utilize a multi-camera system to remove foreground objects. Foreground objects are defined as objects which do not reside on the target observation plane. Prior to this algorithm, the image set will have been cleaned by suppressing glare and shadowing artifacts, and warped into the final observation space. A preliminary final image will have also been constructed by fusing the preliminary image set. What remains to be accomplished is to determine which pixels are foreground and substitute them with previous background data.

4.2.1 Theory and Implementation

Because the preliminary image set was warped from each image's respective input perspective to a common viewing space using a homography, objects not on the target observation plane will appear differently in each warped image. This algorithm exploits this difference and determines foreground pixels through a hypothesis test. This model assumes that the observed pixel can be expressed as

$$y_{i,j} = \nu_{i,j} + p_{i,j} \tag{4.2}$$

where $\nu_{i,j}$ is a Gaussian random variable with a mean and variance that were calculated using calibration data from the cameras, and $p_{i,j}$ is the true pixel value at point i,j. In this test, we desire to know the probability of pixels from different camera inputs originating from the same $p_{i,j}$. Let y_0 and y_1 be data from two different cameras in overlapping regions. Additionally, let μ_0, μ_1, σ_0, and σ_1 be the means and variance from the corresponding cameras as calculated during calibration. Under the null hypothesis, both data points would come from the same origin pixel, and $(y_0 - \mu_0) - (y_0 - \mu_1) \sim \mathcal{N}\left(0, \sigma_0^2 + \sigma_1^2\right)$. This allows us to calculate a z-score by

$$z = \frac{(y_0 - \mu_0) - (y_1 - \mu_1)}{\sqrt{\sigma_0^2 + \sigma_1^2}} \tag{4.3}$$

A threshold, η, can be set such that if $z > \eta$ we reject the null hypothesis and classify this pixel value as foreground.

During the initial detection phase, a preliminary decision image, denoted C, is formed. The next phase is to eliminate false detections and coalesce groups of detections into cohesive objects. To remove outliers and correct misclassified foreground data, a spatial filter was applied over a square region of area l^2 and the results thresholded at λ. This resulted in clustered foreground detections centered on objects of interest.

4.2.2 Pseudocode

The algorithm used to define an initial detection mask is shown in Algorithm 3. In Algorithm 3, an initial pixel value is received and converted into a z-score using camera statistics obtained during the calibration procedure. This preliminary value is then thresholded by determining an appropriate size for the statistics test.

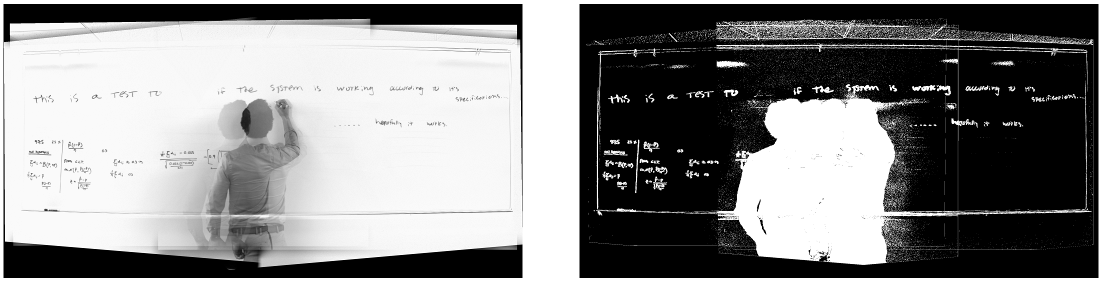
The second portion of the algorithm is found in Algorithm 4 where the results from Algorithm 3 are received as input and filtered. Algorithm 4 lines 1-16 compute the sum of the pixels in a given window. Because this operation can be described as 2-D convolution with a separable kernel, this can be broken down into two convolutional steps where first the rows are convolved and then the columns. The reduces the algorithms complexity to where it can run in real-time. The final part is where the sum is thresholded by a given value λ. If a pixel exceeds the thresholded value, it is classified as foreground in the final decision mask (See Figure 4.3).

Algorithm 3 Initial Foreground Detection Algorithm

```
Require: $\sigma_0^2$,$\sigma_1^2$,$\mu_0$,$\mu_1$,$y_0$,$y_1$,$\eta$
$z = \frac{(y_0-\mu_0)-(y_1-\mu_1)}{\sqrt{\sigma_0^2+\sigma_1^2}}$
decision $= z > \eta$
return decision
```

Algorithm 4 Detection Filtering Algorithm

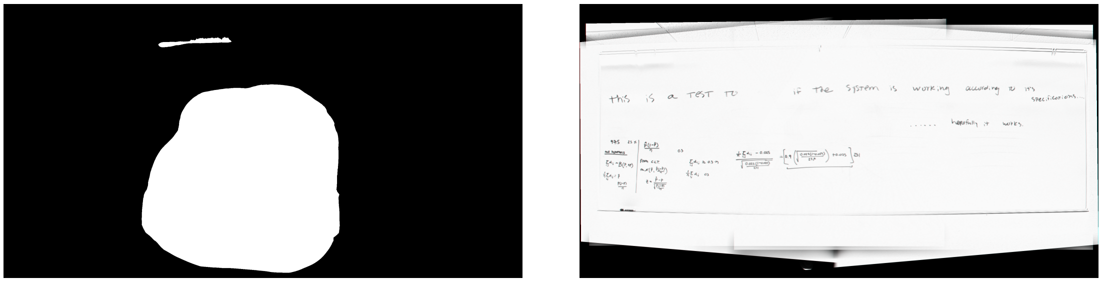
```
Require: $C$,Image Length, ,Image Height, $l$, $\lambda$
$tempImage$=zeros(size(Image))
for j=1:Image Length do
    for k=1:Image Height do
        sum=0
        for i=1:l do
            sum=sum+$C_{k-l/2+i,j}$
        end for
        $tempImage_{k,j}$=sum
    end for
end for
for j=1:Image Height do
    for k=1:Image Length do
        sum=0
        for i=1:l do
            sum=sum+$tempImage_{k-l/2+i,j}$
        end for
        if sum> $\lambda$ then $C_{j,k}$=1
        else $C_{j,k}$=0
        end if
    end for
end for
return $C$
```



(a) Preliminary Fused Image

(b) Preliminary Detection Mask

Fig. 4.2: Initial Foreground Estimate



(a) Final Decision Mask

(b) Final Output Image

Fig. 4.3: Post-Processed Results

4.2.3 Results

Results for the initial foreground estimate can be seen in Figure 4.2. Notice how pixels not found in the plane of interest contain a high detection density whereas other areas don't. Writing also tends to be misclassified which could be attributed to small errors in the homography model. These errors were removed in the next processing step which suppresses false detections (See Figure 4.3).

CHAPTER 5

Discussion and Future Work

5.1 Discussion

Towards the end of the introduction, three questions were posed which defined the work presented in this masters thesis. They were how can the teacher be removed? How can this be implemented on hardware? and How should the software be architected so that it will support real-time streaming? These questions were answered in previous chapters. The teacher was removed by identifying pixels with statistically inconsistent input values, filtering the results to produce the final decision mask shown in Chapter 4. The software architecture was presented in chapter 3, and resulted in an average framerate of 27 frames a second, showing that a real-time implementation is indeed possible.

There were unforeseen complications that had to be addressed. One problem was the shading differences between perspectives. This lead to additional algorithms and calibration steps that had to be taken in order to mitigate the effects caused by these differences. Additionally one problem that had not been anticipated prior was how to capture images at the same time in multiple cameras. Original expectations were that temporal differences in the input images would be negligible. Unfortunately that that was not the case and had to be remedied via hardware triggers and master/slave settings on the cameras.

5.2 Future Work

The system developed in fulfillment of this masters thesis is far from perfect and there are many opportunities for improvement. The calibration procedure continues to be difficult, requiring multiple steps and a fair amount of user interaction. It requires the user to position the cameras and provide sufficient drawings on the whiteboard in order to correctly calibrate the perspective transforms. Results obtained while calibrating the perspective transform, lack consistency because of the random nature in which points are selected. If points are improperly chosen, transforms tend towards numerical instability. Future work would include algorithms to create sets of points engineered to mitigate these effects.

The shading normalization also requires the user to erase and clean the whiteboard prior to calibration. This would be insufficient should the system be used in a setting where it was exposed to variational shading such as near a large window. To acheive performance the calibration would have to be performed several times a day which would not be feasible in many situations. Ideally, a method of calibration would be developed which did not require user interaction and which would allow the shading parameters to be continuously updated. A second change in shading normalization would be to simply normalize all RGB values instead of only changing the luminance. As seen in Chapter 4, shading effects are not entirely removed by this process and perhaps normalizing all RGB values would yield increased performance.

Chapter 4 also shows that the filtering used to convert the preliminary decision mask into a final foreground area loses much of the foreground definition. Improvements to this algorithm would include possible object tracking, as well as image segmentation. These results could provide additional information regarding the shape and position of the foreground object.

Machine learning could also be used to detect and define foreground areas. Neural networks have provided encouraging results in problems with a large number of input parameters and increased variability, both of which are inherenet to this problem.

There were also a large number of false positive around text on the whiteboard. One possible cause would be real-life deviations from the homography model. When computing homographies, it is assumed that images being transformed are all on the same plane. When that is not strictly true this would cause ghosting in the final results. One possible approach, for example, would be to create local mappings which would be more accurate then a general model.

REFERENCES

[1] Y. Lu, Z. Hua, K. Gao, and T. Xu, "Multiperspective image stitching and regularization via hybrid structure warping," *Computing in Science & Engineering,*, 2018.

[2] J. Gao, S. J. Kim, and M. S. Brown, "Constructing image panoramas using dual-homography warping," in *CVPR 2011*, 2011.

[3] Y. Zheng, E. A. Essock, and B. C. Hansen, "An advanced image fusion algorithm based on wavelettransform incorporation with pca and morphological processing," *Algorithms and Systems III*, May 28.

[4] A. Djelouah, "Multi-view object segmentation," 2015.

[5] C. Couprie, C. Farabet, Y. LeCun, and L. Najman, "Causal graph-based video segmentation," *IEEE International Conference on Image Processing*, 2013.

[6] D. Hoiem, A. Efros, and M. Hebert, "Geometric context from a single image," *IEEE International Conference on Computer Vision*, 2015.

[7] P. K. N. Silberman, D. Hoiem, D. H. R. Fergus P. K. Nathan Silberman, and R. Fergus., "Indoor segmentation and support inference from rgbd images," *Proc. of IEEE European Conference on Computer Vision*, 2012.

[8] A. Siddique and S. L. and, "Video inpainting for arbitrary foreground object removal," *IEEE Winter Conference on Applications of Computer Vision*, 2018.

[9] H. Gan, "Chinese education tradition the imperial examination system in federal china," *Journal of Management and Social Sciences*, 2008.

[10] M. Kazin, R. Edwards, and A. Rothman, *The Princeton Encyclopedia of American Political History.*, 2010.

[11] N. C. f. E. S. U.S. Department of Education. (2017) Fast facts. [Online]. Available: https://nces.ed.gov/fastfacts/display.asp?id=98

[12] N. S. University. (2015) Pros and cons of online education. [Online]. Available: https://www.ies.ncsu.edu/resources/white-papers/pros-and-cons-of-online-education/

[13] Z. Zhang and L. wei He, "Note-taking with a camera: whiteboard scanning and image enhancement," in *IEEE International Conference on Acoustics, Speech, and Signal rocessing*, 2004.

[14] R. Bhangale and P. M. Mahajan, "Image fusion techniques," *International Journal on Recent and Innovation Trends in Computing and Communication*, 2015.

[15] D. K. Sahu and M. P. Parsai, "Different image fusion techniques a critical review," *International Journal of Modern Engineering Research*, 2012.

[16] C. K. Solanki and N. M. Patel, "Pixel based and wavelet based image fusion methods with their comparative study," in *National Conference on Recent Trends in Engineering & Technology*, 2011.

[17] G. Pajares and J. M. de la Cruz, "A a wavelet-based image fusion tutorial," *Pattern Recognition Society*, 2004.

[18] T. Xiang, G.-S. Xia, L. Zhang, and N. Huang, "Locally warping-based image stitching by imposing line constraints," in *Pattern Recognition (ICPR)*, 2016.

[19] D. Lee, J. Yoon, and S. Lim, "Image stitching using multiple homographies estimated by segmented regions for different parallaxes," in *Vision Image and Signal Processing (ICVISP)*, 2017.

[20] C.-C. Lin, S. U. Pankanti, K. N. Ramamurthy, and A. Y. Aravkin, "Adaptive as-natural-as-possible image stitching," in *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[21] X. Pan and G. Wang, "Parallax-tolerant image stitching based on mesh optimization," in *dvanced Information Technology Electronic and Automation Control Conference (IAEAC)*, 2017.

[22] S. Li, L. Yuan, J. Sun, and L. Quan, "Dual-feature warping-based motion model estimation," in *Computer Vision (ICCV)*, 2015.

[23] V. S. Petrovic, "Gradient-based multiresolution image fusion," *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 2004.

[24] C. Farabet, C. Couperie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling." *IEEE Trans. on Pattern Analysis and Machine Intelligence,*, 2013.

[25] P. F. Felzenszwalb and D. P. Huttenlocher, "efficient graph-based image segmentation." *International Journal of Computer Vision*, 2004.

[26] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Internatinoal Journal of Computer Vision*, 2004.

[27] M. A. Fischler and R. C. Rolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, 1981.

CHAPTER 6

CURRICULUM VITAE

Education

2019 Utah State University

Masters of Science, Electrical Engineering

2019 Utah State University

Bachelors of Science, Electrical Engineering

2019 Utah State University

Minor in Mathematics

2019 Utah State University

Minor in Computer Science

Research

Current Foreground/Background Separation — Utah State University

This area of research focuses on separating foreground and background objects in order to display only intransient background information. To accomplish this, the images are segmented and each segment is tracked over time. The segments that are classified as moving are not included in the image update, whereas stationary segments are incorporated into the display image.

Current Noise Minimization in the VLF Band — Sandia National Laboratories

The VLF project centers around characterizing potential noise sources in the VLF band, and mitigating their effects in communication systems. Various methods were tested including MMSE predictors, phase limiters, and low-pass filtering, with the phase limiter in conjunction with low pass filtering achieving the highest results. The current noise rejection technique adds 6 dB to the noise floor tolerance.

2017-2018 Image Orthorectification and Fusion — Utah State University

Using algorithms such as SIFT and RANSAC to identify points of interest between overlapping images, a homographic mapping is calculated which transforms pixels from one image space to the next. This mapping is used to fuse images together, using bi-linear interpolation to compensate for non-integer mappings and estimation techniques to regularize image transition imbalances in the resulting image.

2016-2017 Infant Biometrics Estimation — Photorithm Inc.

An algorithm for detecting infant breath rate biometrics was developed using real-time eigenvector approximation techniques. In disturbance free environments, infant location was successfully detected and a biometric waveform generated in real-time. Peak detection algorithms were utilized to quantify breaths/second

Industrial Experience

2017 Product Engineer Intern — Micron Technology, Inc.

The objective of a product engineer is to monitor the health of a product and coordinate efforts to produce and improve it throughout its life cycle. During my time as a product engineer, I designed and programmed 5 additional bin tests to verify product functionality as well as recommended fixes for 2 race conditions during the course of the internship

2016 Software Maintenance Developer — Hill Air Force Base

Was 1 of 2 interns selected to work for the A-10 Aircraft software maintenance group (SMXG). SMXG's mission was to support A-10 pilots by improving software performance and user interfaces. During the course of the internship, I developed a light weight aircraft simulator in under 3 months with the purpose of allowing pilots to verify changes to the user interface remotely.

Presentations

2018 Optimal Estimation for Aerospace Systems — Utah State University

This two part lecture series was given in relation to an aerospace estimation course at Utah State University were the derivation for the equations used in the Kalman Filter was presented. Lecture material included the derivation of the propagate and update step using prior distributions in conjunction with Gaussian assumptions to compute a posterior estimator that is optimal in both the minimum mean squared and Bayesian sense.

2018 Radar and Signal Processing Division — Sandia National Laboratories

Research on noise characterization and mitigation in the VLF band was presented at Sandia National Laboratories following an internship in the Radar and Signal Processing Division at Sandia National Laboratories. Discussion topics included potential noise sources as well as mitigation techniques that allowed communications systems to function in high noise environments.

2017 Product Engineering — Micron Technology, Inc.

This presentation was given to management employees at Micron Technology Inc. to update them on test development for a multilevel DRAM architecture. Traditional DRAM has been developed on a single level because of fabrication constraints. Recently, advances in process technology have made possible multilevel wafer fabrication which requires alternative testing methods. This lecture presented fixes and updates to said architecture as well as novel testing methods designed to increase product quality

Additional Projects

Speech Diarization using Neural Networks.

Linear classifier using Perceptron Algorithm.

Implementation of Linear Logistic and Quadratic Regression, K-nearest Neighbor, and Naive Bayes methods in a binary classifier on mixed-model data sets.

Classification of MNIST data set Using Neural Network.

LLL Algorithm implementation to demonstrate vulnerabilities in various cryptographic algorithms.

Emulator of Speak and Spell using a Yule-Walker model and linear predictive coding.

Implementation of Discrete and Continuous Time Kalman Filter, Extended Kalman Filter & Particle Filter.

Estimated parameters of Gaussian mixture model and Hidden Markov model using the Expectation-Maximization algorithm.

Used blind source separation to statistically disentangle mixed audio feed.

Implemented real time Sobel Edge Detection in CUDA

Successfully demonstrated parallel process image fusion using CUDA

Removed White Noise from speech signal using adaptive FIR filter

Coursework

Detection and Estimation — Dr. Todd Moon

Course materials included detection theory, including Neyman-Pearson, Bayes, and Minimax Bayes detection. Maximum likelihood and Bayes estimation theory. Recursive estimation and Kalman filtering and smoothing. Expectation maximization and hidden Markov models. Lectures on detection using Neyman-Pearson, Bayes, and Maximum A Posteriori were also presented.

Mathematical Methods for Signal Processing — Dr. Todd Moon

Signal representation using vector spaces. Linear algebraic techniques for signal modeling and estimation. Optimal detection and estimation algorithms, with applications.

Stochastic Processes — Dr. Todd Moon

Introduction to stochastic processes in communications, signal processing, digital and computer systems, and control. Topics include continuous and discrete random processes, correlation and power spectral density, optimal filtering, Markov chains, and queuing theory

Neural Nets and Machine Learning — Dr. Todd Moon

Advanced course in theories and techniques of machine intelligence, using neural networks. Information on traditional detection techniques was also presented.

Probability and Statistics — Dr. Kady Schneiter

Discrete and continuous probability, random variables, distribution and density function, joint distributions, conditional probabilities and expectations, Bayes' theorem, moments, moment generating functions, inequalities, convergence in probability and distribution, and central limit theorem. Basic theory of point and interval estimation and hypothesis testing. Topics include: sufficiency and completeness; method-of-moments, best unbiased, maximum likelihood, Bayes', and empirical Bayes' estimators; Neyman-Pearson lemma; and likelihood ratio tests

Optimal Estimation for Aerospace Systems — Dr. Randall Christensen

Probability theory, stochastic system models, optimal estimation for linear systems, optimal smoothing, and optimal estimation for nonlinear systems. Applications include orbit determination, real-time position, velocity, and attitude determination for rockets, aircraft, and spacecraft.

Convex Optimization — Dr. Jacob Gunther

The theory of convex optimization and applications, as applied to engineering. Numerical methods for solving convex optimization problems are presented. Computational work required.

Real Time Processing — Dr. Scott Budge

Real-time processor architectures and methods used for digital signal processing. Includes C and assembly language programming, modern DSP architectures, tools for real-time system development, and finite word-length effects. Real-time system design and implementation of basic concepts, including modeling, scheduling, resource access control, synchronization, and communication. Emphasis placed on both theory and practice. Exploration of open topics and current challenges in designing real-time systems. Includes hands on implementation.

Computer Science — Dr. Kenneth Sunberg

Basic instruction given on C++, C, Python, and Matlab. More advanced courses included material on data structures and algorithms as well as software analysis using Big O notation and cyclomatic complexity. Coursework on operating systems, parallel computing, and image processing was also received.

Continuous and Discrete Time Systems and Signals — Dr. Jacob Gunther

Time domain analysis of higher-order systems: impulse response and convolution. Laplace transform analysis of circuits and systems. Frequency domain analysis including discrete Fourier series, Fourier transforms, and analog filter design. State-space representations and analysis of systems. Sampling of continuous-time signals. Time and z-transform domain analysis of discrete-time systems. Frequency domain analysis using the discrete-time Fourier transform, DFT and FFT. Frequency response and digital filter design.