STEREO CAMERA CALIBRATIONS WITH OPTICAL FLOW

THESIS

Joshua D Larson, 2d Lt, USAF

AFIT-ENG-MS-21-M-056

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-21-M-056

Stereo Camera Calibrations with Optical Flow

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Computer Science

Joshua D Larson, B.S.C.S.

2d Lt, USAF

March 25, 2021

AFIT-ENG-MS-21-M-056

Stereo Camera Calibrations with Optical Flow

THESIS

Joshua D Larson, B.S.C.S.
2d Lt, USAF

Committee Membership:

Scott L Nykl, Ph.D
Chair

Clark N Taylor, Ph.D
Member

Brett J Borghetti, Ph.D
Member

AFIT-ENG-MS-21-M-056

Abstract

Remotely Piloted Aircraft (RPA) are currently unable to refuel mid-air due to the large communication delays between their operators and the aircraft. Automated Aerial Refueling (AAR) seeks to address this problem by reducing the communication delay to a fast line-of-sight signal between the tanker and the RPA. Current proposals for AAR utilize stereo cameras to estimate where the receiving aircraft is relative to the tanker, but require accurate calibrations for accurate location estimates of the receiver. This paper improves the accuracy of this calibration by improving three components of it: increasing the quantity of intrinsic calibration data with Convolutional Neural Network (CNN) preprocessing, improving the quality of the intrinsic calibration data through a novel linear regression filter, and reducing the epipolar error of the stereo calibration with optical flow for feature matching and alignment. A combination of all three approaches resulted in significant epipolar error improvements over OpenCV's stereo calibration while also providing significant precision improvements.

Acknowledgements

First and foremost, I would like to thank my friends and family who consistently supported me throughout this program, and without whom this research would not have been possible. I would also like to thank my committee members: Dr. Nykl, Dr. Taylor, and Dr. Borghetti; who not only supported my initial aimless wandering through research topics, but also taught me everything I needed to know to be successful. Finally, a special thank you to my friends at AFIT, who consistently made the long hours of studying and researching enjoyable.

Joshua D Larson

Table of Contents

Page

List of Figures

List of Tables

Stereo Camera Calibrations with Optical Flow

I. Introduction

The typical control latency of Remotely Piloted Aircraft (RPA) is too high to safely refuel mid-air[4], resulting in limited time on-mission. Line-of-sight communication reduces the control latency to safe levels for aerial refueling[4]. The tanker is the only object that is guaranteed to be within line-of-sight of the RPA, so to perform aerial refueling on RPAs, the tanker needs to (1) fly itself, (2) control its refueling boom, and (3) control the receiving aircraft (RPA). Automated Aerial Refueling (AAR) aims to do these three things. AAR is the process of aerial refueling without any human interaction, from the tanker nor receiver. AAR requires an accurate location estimate for both the receiving aircraft and the refueling boom, and improving these location estimates is the primary goal of this thesis. AAR is important for the United States Air Force (USAF) because it enables nearly unlimited time on-mission.

AAR still has several challenges to overcome before it can be deployed. Estimating the relative location of the receiving aircraft with very high accuracy is one of the primary challenges of AAR. This has to be done without the use of Global Positioning System (GPS), because GPS is susceptible to jamming and spoofing. Vision-based systems may provide a viable alternative to GPS, and is the primary research focus at the Air Force Institute of Technology (AFIT) for AAR.

One vision-based approach for AAR involves using two cameras, also called stereo cameras, to estimate the distance to the receiver. Stereo vision works similar to how humans use their eyes to estimate distance. One of the problems with using stereo

cameras for AAR is that it requires a very accurate stereo calibration, especially as the resolution of the cameras increases. Improving these stereo calibrations is the core purpose of this research.

1.1 Problem Statement

AAR needs high quality stereo calibrations to be able to accurately determine the receiver's location, and accomplishing that requires two components: (1) improving the intrinsic calibrations, which are eventually fed into the second component (2) improving the stereo (extrinsic) calibration. This research aims to improve both of these components in the following ways:

1. Increase the quantity and quality of data available to the intrinsic calibration
2. Incorporate optical flow into the stereo calibration process to reduce distance estimation errors

Optical flow is the measurement of motion between two images. Typically, optical flow is measured between different frames of a video to see how each object is moving, but optical flow can be applied to any pair of images; for example, when applied to stereo images, the optical flow corresponds to the distance to every pixel in the left image. The connection between optical flow of stereo images and distance estimation enables optical flow to be used for stereo calibrations.

The next chapter of this thesis lays out the background information for this work. Starting with a more in-depth overview of AAR, then moving into the relevant elements of the pinhole camera model and how that relates to calibrations, and then wrapping up with how deep learning and optical flow have previously been used for similar work.

The third chapter lays out how this work was conducted, starting with the core

functionality of the optical flow estimation, then how the intrinsic calibration quality can be increased through image adjustments and filters, and then finally how the optical flow estimation is combined with the intrinsic calibration to generate high-quality stereo calibrations.

The fourth chapter discusses the results of applying this methodology to this problem, with analysis on what went well and what can be improved. The overall thesis is then wrapped up in chapter five with recommendations for future work in this area.

II. Background and Literature Review

2.1 Aerial Refueling

Aerial Refueling is the process of using tankers to refuel receiving aircraft while mid-air. There are two primary ways that this is accomplished: using a boom and using a probe and drogue. The probe-and-drogue method is accomplished by having a refueling line hanging out the back of the tanker that is connected with the probe on the receiver. Since the drogue is not controlled, the responsibility for successful refueling lies with the receiver. The boom method utilizes a controllable arm on the tanker that is maneuvered into the receiver's refueling port. This is the primary method employed by the United States Air Force (USAF) as seen in Figure 1, and is also the focus of this research. Refueling with a boom occurs within the *refueling envelope*, which is approximated in this research as 30m between the vision system and the refueling port on the receiver.

Remotely Piloted Aircraft (RPA) are currently unable to refuel safely with the boom method due to the latency between the Ground Control Station (GCS) and the RPA[4]. To solve this, De Vries [4] proposes using methods with lower latency, such as direct line of sight communication. In the case of Automated Aerial Refueling (AAR), this means having the tanker control both the boom and the receiver. This introduces some challenges:

- The tanker needs an accurate pose estimate (less than 10cm of error) of the receiver to ensure neither the tanker nor receiver are damaged
- The pose estimation cannot use Global Positioning System (GPS) because it is susceptable to jamming and spoofing

Figure 1: A tanker refueling a receiver with a boom [1]

- The receiving aircraft cannot be modified to add visual markers–AAR must work on receiving aircraft as-is
- The AAR pipeline must run in real-time to keep the boom and receiver in the correct position and orientation

There have been several papers and theses published from the Air Force Institute of Technology (AFIT) on AAR that are elaborated on in the next section.

2.2 Previous AFIT Work

Some of the earliest work at AFIT established the current AAR vision pipeline, and was done with simulated imagery [5]. Parsons et. al. set up a virtual environment with stereo cameras, then used Iterative Closest Point (ICP) to calculate the pose of

the receiver. The simulated vision pipeline is directly built on by [6], which determines how much more accurate pose estimations are in simulated imagery compared to real imagery. Then [7] improved the pipeline further by adding in a Convolutional Neural Network (CNN) to filter out the irrelevant parts of the stereo images, speeding up the pipeline. The other important contribution from [7] was establishing that higher resolution cameras dramatically improve the pose estimation of the receiver. The reason that the higher resolution cameras improved the pose estimation accuracy is in part due to the limitations of the pinhole camera model on smaller resolutions.

2.3 Pinhole Camera Model

The pinhole camera model describes how a 3D environment is captured by a typical camera: light passes through the aperture and onto the image sensor. This is modeled as a perspective projection, as shown in Figure 2. Every 3D point p in the scene is projected onto the 2D plane with the coordinate (u, v). The principal point is at the location (c_x, c_y), is typically very close to the center of the image, and is primarily useful for distortion correction.

There are a few assumptions in the pinhole camera model: (1) there is no lens distortion, (2) the aperture is a single point (F_c), and (3) pixels are continuous. These assumptions typically do not hold, but can be compensated for to some extent with the proper calibration. To compensate for the lens distortion, the computer vision library OpenCV uses radial and tangential distortion expanding from the principal point[2]. Another approach is to have a per-pixel undistortion, where each pixel has a unique shift according to its placement in the image [8]. To correct for the size of the aperture and the size of the image (in pixels), a camera calibration K is introduced that transforms the projected points into pixel values. Finally, as illustrated in Figure 2, each pixel is a discrete bin rather than a continuous value. This can be compensated

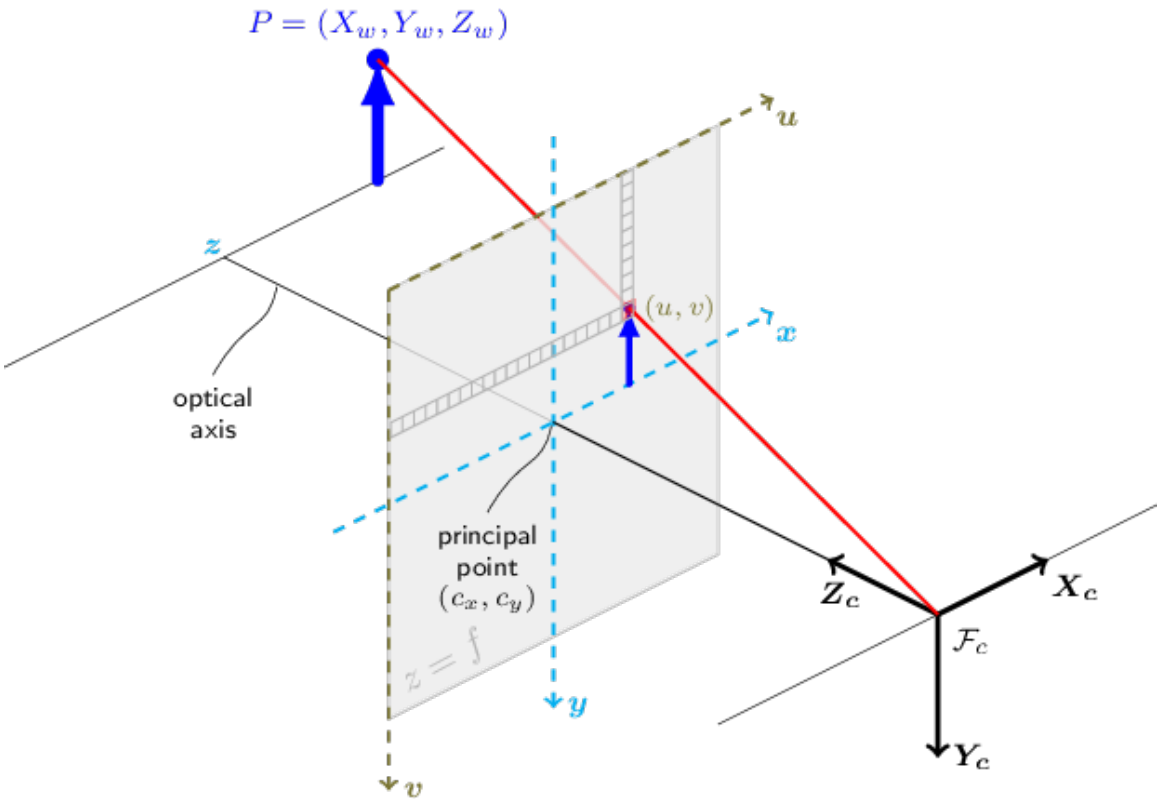

Figure 2: Pinhole Camera Model [2]

for by estimating where the point would lie between the bins or decreasing the size of each pixel.

After the perspective projection is completed, the 3D point now lies on the 2D image plane–losing the depth component. In order to recover the depth, more information is required about the image. One key element to recovering the depth component is to calibrate the camera, and is described in the next section.

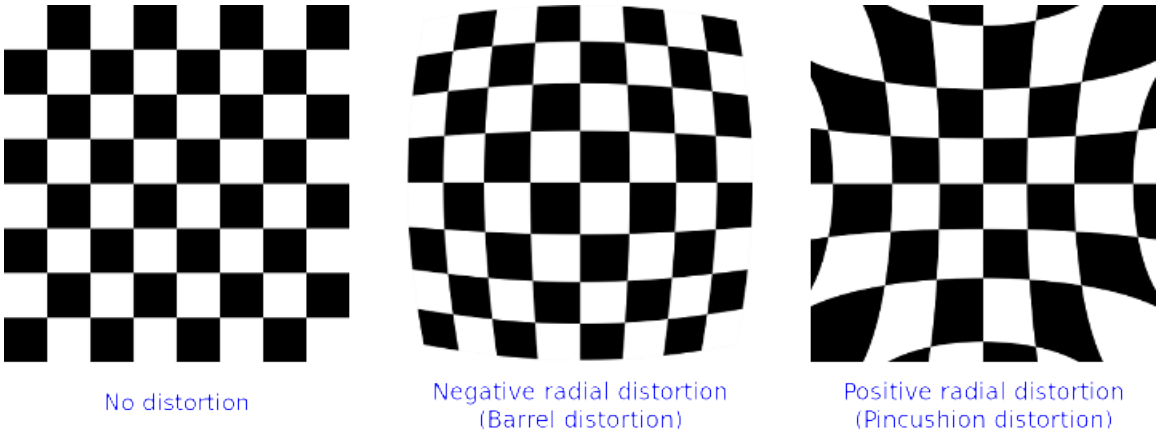

Figure 3: Radial Distortion Examples [2]

2.4 Camera Calibrations

Calibrating a camera is the process of understanding how the real world is captured onto a set of pixels according to the pinhole camera model. Described in the last section were a few shortcomings of the pinhole camera model and how they can be compensated for, and once they are compensated for the next step is to determine the intrinsic calibration matrix. The intrinsic calibration matrix transforms a 3D point (x, y, z) into a 2D pixel coordinate (u, v), and has the following structure:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = K \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad (1)$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{x'}{z'} \\ \frac{y'}{z'} \end{bmatrix}$$

The K matrix is multiplied with the column vector $(x,\ y,\ z)^T$, and then divided by its last dimension to get the two-dimensional $(u,\ v)^T$ coordinate. This operation is shown in Equation (1). In this matrix, f_x and f_y convert physical x and y coordinates into pixel coordinates, which is then shifted by c_x and c_y for the optical axis–typically near the center of the image.

In order to determine the correct intrinsic calibration matrix, frequently calibration patterns are used–such as chessboards or a grid of stars[8]. The purpose of the calibration pattern is to have well-defined image coordinates as well as the relative distances between each of the *key points* on the objects. The key points used depend on the pattern. For chessboards, each corner of the chessboard is used as a key point. An example of these key points is shown in Figure 4.

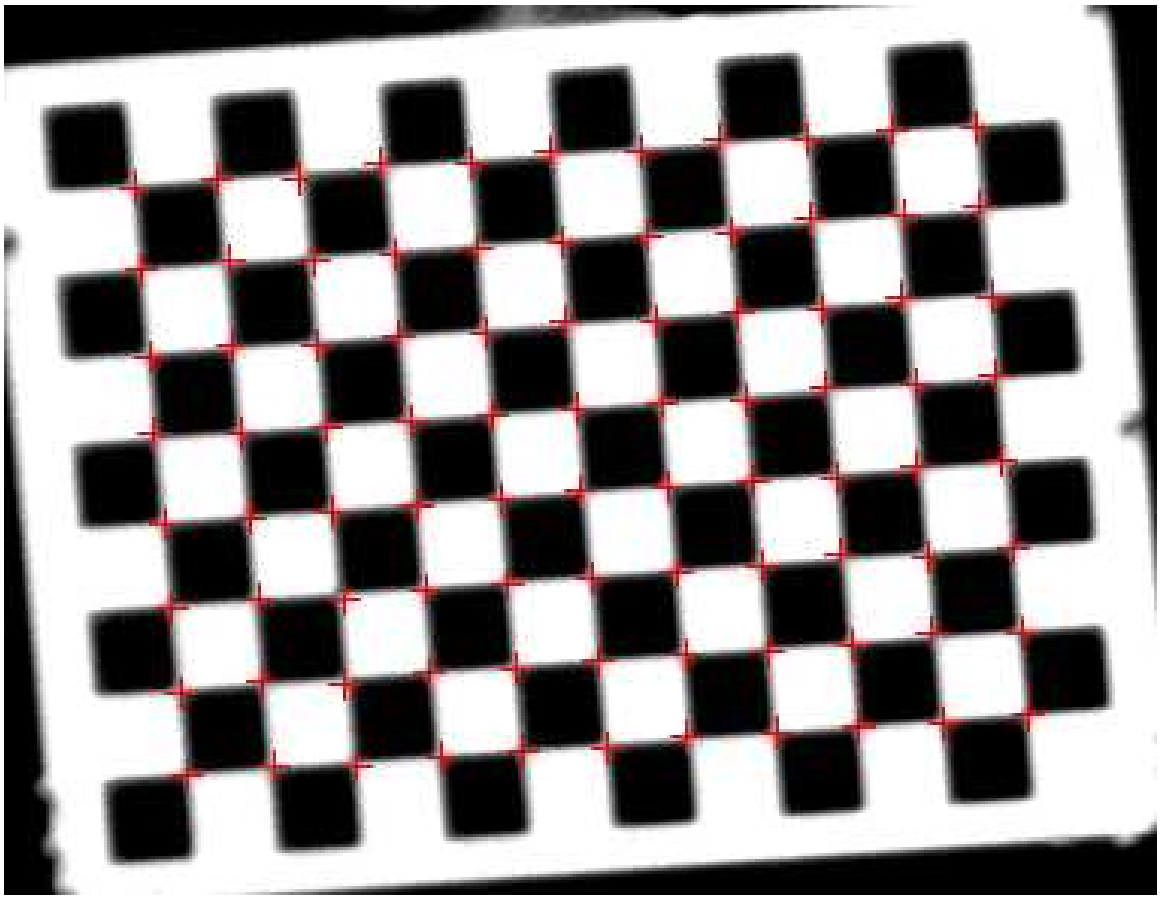

Figure 4: Example calibration pattern with corner detection

To use the camera calibration, specific key points are identified on an object and then the object's position can be solved for. Since it is not always practical to find the same key points on an object, and in the case of AAR no markers can be added to a receiving aircraft, another method for localizing objects involves using a second camera to recover the depth information with epipolar geometry.

2.5 Epipolar Geometry and Stereo Cameras

Epipolar geometry describes how a feature in one camera can be found in a second camera. For a point in one camera at (u, v) with an unknown depth, there is a line in the second camera where that point could be. This line is called the epipolar line, and every point in the first camera has one. An example of an epipolar line is shown in Figure 5. Where the point actually falls along the line directly corresponds to how far away it is, and is the basis for stereo vision. Since the distance along the epipolar line corresponds to how far away the point is, the next core problem is finding matching features along these epipolar lines, and one common way of doing that is with a process called Stereo Block Matching (SBM).

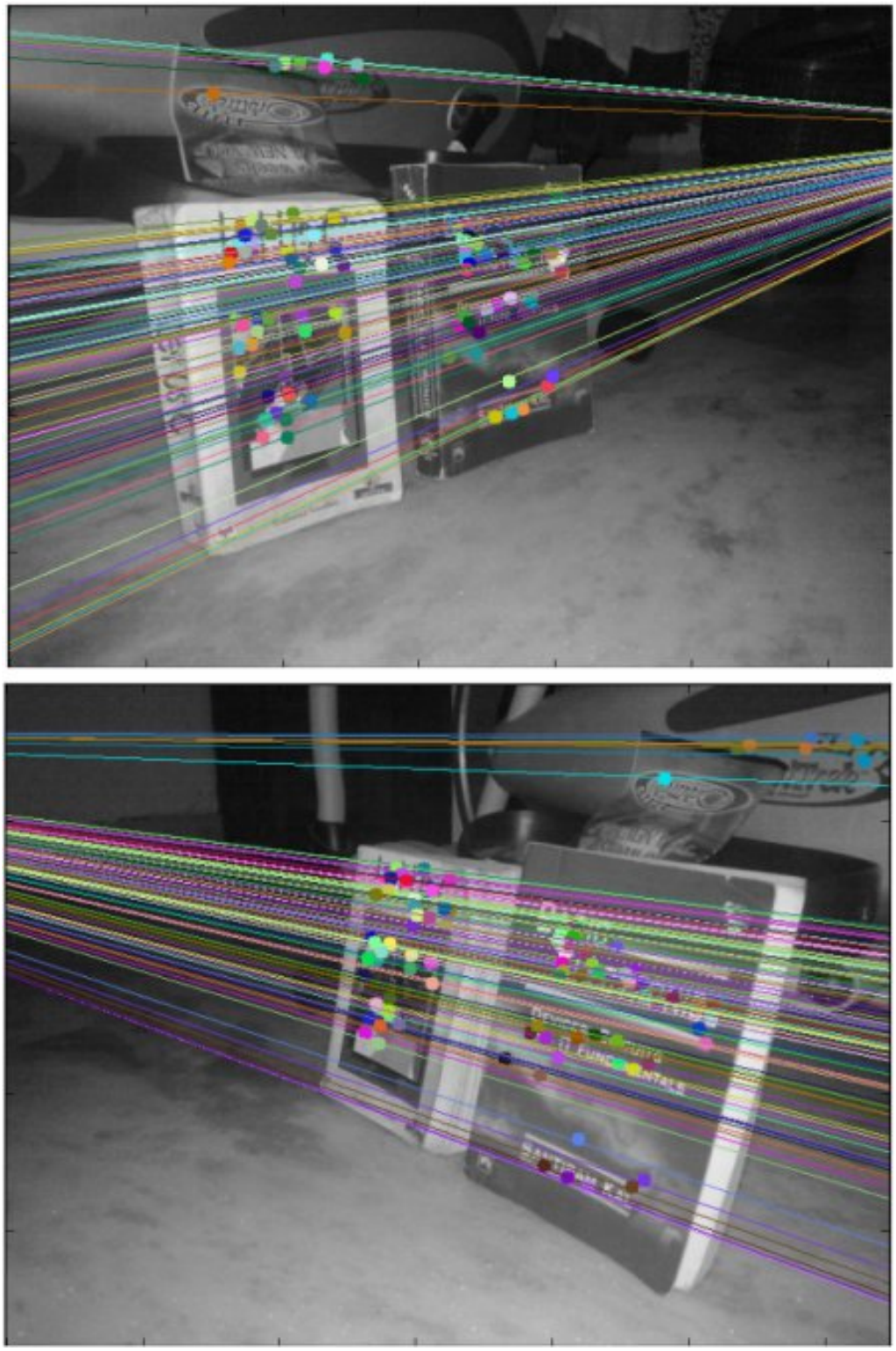

Figure 5: Matching features and their epipolar lines [3]

2.6 Stereo Block Matching

SBM is the process of finding matching features by scanning along the epipolar lines of an image. The epipolar line scan is also referred to as a walk, and entails moving from pixel to pixel on the epipolar line in the second camera. For every pixel in the first camera, its epipolar line in the second camera is walked until a matching feature is found. A matching feature, in the case of OpenCV, is determined by a metric called the Sum of Absolute Differences (SAD). As the epipolar line is walked, a region around each candidate feature in the second camera is taken and subtracted from a region surrounding the original feature in the first camera, and the sum of the absolute value of each of these differences is the final value. The size of this region is called the SAD window size, and the candidate feature with the smallest SAD wins. The distance from the start of the walk to the matched feature is called a disparity, measured in pixels. Since epipolar lines are typically sloped (i.e. not perfectly horizontal nor vertical), and walking along a sloped line in an image is inefficient on most CPUs and GPUs, the pair of images are typically rotated such that each epipolar line is horizontal. This rotation is called image rectification, and the process for finding the epipolar lines necessary for rectification is called stereo calibration.

2.7 Stereo Calibrations

A stereo calibration has two primary components: image rectification and solving for the reprojection matrix. Image rectification is the process of modifying the images such that the epipolar lines are horizontal, but for the remainder of this thesis it is used synonymously with the process of finding the epipolar lines in the context of stereo calibrations. Image rectification can be split up into two properties:

- Ensuring all matching features lie on the same row of both images (vertical alignment or v-based rectification)
- If a matching feature is infinitely far away, then it will be located at exactly the same pixel in both images (horizontal alignment or u-based rectification)

The reason for the latter has to do with the reprojection matrix, which is called the Q matrix in OpenCV, and is shown in Equation (2). Using the Q matrix, a pixel at coordinates (u, v) with a disparity of d (measured in pixels) is multiplied with Q, then divided by its 4th dimensional homogenous coordinate, w, to compute the 3D coordinate of that feature–as shown in Equations (3) and (4).

$$Q = \begin{bmatrix} 1 & 0 & 0 & c_x \\ 0 & 1 & 0 & c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{b} & 0 \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ w \end{bmatrix} = Q \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{3}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x'/w \\ y'/w \\ z'/w \end{bmatrix} \tag{4}$$

According to the pinhole camera model, as the disparity d approaches zero, w also approaches zero, and when x, y, and z are divided by w, they approach infinity. Since cameras have a finite resolution, points appear infinitely far away at some non-zero fraction of a pixel. As the disparity increases, the distance from the camera

decreases. In order to calculate the correct values for the Q matrix, the disparity d, image coordinate (u, v), and actual position (x, y, z) of several features must be known. Determining the actual position of several features is typically accomplished using the same calibration patterns from the intrinsic calibration. An alternative approach is proposed by [9] which takes the intrinsic calibrations of both cameras as well as the Q matrix and solves for the rectification without using any calibration patterns. After reprojection with the Q matrix, the next natural step is to use those reprojected points to find the locations of objects in the environment.

2.8 Pose Estimation

Pose estimation is the process of generating a 6 Degree-of-Freedom (6DoF) estimate for the position and orientation (pose) of an object, with 3 variables for translation (x, y, z) and 3 variables for rotation (roll, pitch, yaw). In the computational approach, this is split into two parts: nearest neighbor search, then a pose estimation update[10]. These two steps are repeated until the pose estimation converges to a single solution or the number of iterations hits an uppper limit. Deep learning has also been applied to this problem in a number of ways: through optical flow[11], augmenting the nearest neighbor algorithm[12], and by finding key points on the object to directly calculate pose[13, 14, 15]. Deep learning can be applied to many other elements of computer vision as well, especially through processing the images directly to extract additional information–as described in the next section.

2.9 Deep Learning in Computer Vision

Two of many goals for deep learning in computer vision is to transform numbers (pixels) into meaningful information or useful representations. This transformation is typically done through a Convolutional Neural Network (CNN), which is composed

of a variety of layers: convolutional layers, which are based on convolutions from linear algebra; pooling layers, which aggregate information across an image; and non-linear activation layers, which allow a CNN to approximate non-linear functions[16]. Chaining multiple convolutions together allows a CNN to learn complex relationships across the image. This section will detail some of the methods used in a CNN to achieve a variety of tasks, starting with principles of CNNs, then going into image segmentation networks, and then wrapping up with optical flow networks.

2.9.1 Convolutional Neural Network Principles

At a high level, a neural network is a function approximator. There are a variety of layers that make up a neural network, and some contain parameters for the approximated function. The process for learning the values for these parameters is called optimization, and a popular method for doing so is called Stochastic Gradient Descent (SGD)[17]. SGD minimizes a cost function that represents how far away the neural network's output was from what it should be (the truth values). SGD uses the first-order partial derivative in each layer of the neural network to approximate the shape of the cost function with respect to each of the network's parameters. Then, SGD iteratively updates each parameter in the direction that appears to minimize the cost function. While SGD works very well for small-parameter networks, as the network size grows, so do possible issues. One of the most common issues is vanishing and exploding gradients, which is where the loss function either shrinks or grows exponentially as it is propagated through the network. One solution to this is adding normalization, such as Batch Normalization[18], which normalizes all of the activations of the network to have approximately zero mean and a standard deviation of one. Another approach to solving this problem is called Self-Normalizing Networks[19], which uses the Scaled Exponential Linear Unit (SELU) activation function to con-

verge towards a zero mean and a standard deviation of one–allowing arbitrarily large networks.

While normalization allows the size of the neural network to grow to any size without vanishing or exploding gradients, there is still information loss as the length of the network increases. Two ways of addressing the information loss problem, in addition to the vanishing and exploding gradient problems, are Residual Networks and Highway Networks[20, 21]. The purpose of both is to create skip connections that have little effect on the actual performance of the network, but allow for the gradient to propagate back easier and for information to flow forward easier. Another notable neural network design is the Fully Convolutional Network, which is designed to take in an arbitrary-sized input and create an arbitrary-size output[22, 23]. In both papers, the original image is contracted to a smaller size to allow for image context to be learned, and then expanded to generate per-pixel regressions. This expansion is sometimes achieved using transposed convolutions, which work similar to standard convolutions, but will upscale the input[22]. Finally, Inception networks were created to expand the *receptive field*, which is the image-space range (in pixels) of each pixel regression, of the contraction (and optimize for GPU resources)[24].

2.9.2 Image Segmentation

Image segmentation in deep learning is the process of generating per-pixel proposals for what object(s) exist in each part of an image[25]. This type of network frequently employs many of the aforementioned techniques to achieve this: fully connected networks, inception networks, highway networks, and sometimes residual networks[26]. While image segmentation is frequently performed to estimate what object is contained by each pixel, the same architecture designs can be run on any image to determine its contents. One example of similar architectures being run for

different purposes is Optical Flow, which is described more in the next section.

2.9.3 Optical Flow

Optical flow is the measurement of motion between two images, measured in pixels. For every point in the first image of the pair, its matching point is found in the second image. Typically optical flow is run on pairs of frames in videos, since the smaller motion between video frames tends to be easier to estimate than larger movement between two arbitrary images; although, technically optical flow algorithms can run on any pair of images that are reasonably similar to each other. The effectiveness of optical flow algorithms, in particular with neural networks, is largely dependent on the dataset used to train it.

2.9.3.1 Datasets

This research uses two optical flow datasets for training and evaluation: Flying Chairs and MPI-Sintel, respectively[27, 28]. Flying Chairs is a collection of 22,872 512x384 image pairs, along with ground truth optical flow between the pair. Each image is composed of a random image from Flickr, along with one or more chairs placed randomly in the image, then both the background and chairs are randomly moved and rotated in the second image. While clearly an unlikely scenario in the real world, this dataset was found to be effective in teaching a network to generalize to many different types of motion[29]. The second dataset is MPI-Sintel, which is a series of short video clips from the open source movie Sintel. Since each image input is a frame of a video, the optical flow is more representative of a real world scenario, which makes it a popular benchmark for optical flow algorithms. Since Sintel has just 1065 flow images, it's common to train on Flying Chairs first, then fine tune on Sintel prior to evaluation[27, 30]. The next section goes over some neural network

architectures for optical flow.

2.9.3.2 Neural Networks

There are three main components to almost every optical flow network: feature extraction, correlation, and flow refinement. The first step of feature extraction is common to almost every CNN - it is the process of turning each pixel into a description of its surrounding area. Some networks are designed specifically to generate feature descriptors such that the l_2 distance between two feature descriptors corresponds to the similarity between those two features[31]. Once the feature extraction is completed on both images, for every point (u, v) in the first image, a region surrounding (u, v) in the second image is scanned for matches. These are called correlation layers, and are elaborated on in the next section. After the correlation layer, every network has some amount of smoothing (typically using more convolutional layers) to even out the noise in the final flow output. Flow refinement is also primarily where differences in architectures appear.

One of the standard methods for flow refinement is adding in convolutional layers that learn the correct way to smooth and correct a flow estimate[27, 32, 30]. Some other work uses 3D convolutions to refine flow, which has the effect of evaluating every flow possibility as likely or unlikely prior to reduction[33]. In order to assist the network in determining bad flow, some architectures employ image or feature warping to test a flow estimate and re-evaluate[32, 34, 35, 36]. Since CNNs are designed for pattern recognition, they struggle with data comparison. In other words, CNNs struggle to perform the matching step required for optical flow networks. Correlation layers were built to address this limitation.

2.9.3.3 Correlation Layers

Correlation layers are effectively search algorithms for neural networks, encapsulated into the base unit of neural networks: a layer. First proposed by [27], it works by comparing a feature descriptor at (u, v) in the first image to a nxn region around (u, v) in the second image to create an n^2 channel output representing the similarity of each feature in the second image to the original feature in the first. As the search window is increased, the memory requirements of the network increase quadratically. The n^2 channel output is called a correlation volume (or cost volume), and is typically what is refined by the network into the final 2-channel flow estimation output. Some alternatives to the traditional correlation operation have been proposed, especially for stereo networks where the network creates a custom cost volume that is then reduced through a center-of-mass operation (soft-argmin)[37].

2.10 Background Overview

In the next chapter, all of these concepts will be combined together to accomplish the three primary components of this research: increasing the quantity of intrinsic calibration data, increasing the quality of the intrinsic calibration data, and creating an optical flow network to perform the vertical rectification of the stereo calibration. To increase the quality of intrinsic calibration data, an image segmentation network was designed to locate the chessboard inside of an image, and then an algorithm modifies the image to make it easier for OpenCV to detect. To increase the quality of intrinsic calibration data, bad calibration patterns were detected and removed using a novel linear regression filter. Finally, the optical flow network uses a custom correlation layer and architecture to scale up to high-resolution images, and is then used to keep all matching features on the same row of both images–an important component of stereo calibrations.

III. Methodology

There are three primary ways this thesis seeks to improve camera calibrations: improve the quantity of intrinsic calibration data, filter out poor intrinsic calibration data, and augment stereo camera calibrations with optical flow. The improvements to intrinsic calibrations also leads to improvements in stereo calibrations, because the intrinsic calibration data is used for both intrinsic and stereo (extrinsic) calibrations. In order to use a neural network-based optical flow estimation on high-resolution imagery, the correlation layer needs to be changed to use less memory, which is described in detail in Section 3.1. Section 3.2 describes two methods: one for improving the quantity of intrinsic calibration data, and another for improving the quantity of intrinsic calibration data. Section 3.3 ties everything together: the novel correlation layer for high-resolution imagery, the improved intrinsic calibration data, and the optical flow neural network to generate a novel stereo camera calibration using optical flow. Finally, Section 3.4 describes how the new stereo calibration will be evaluated in Chapter IV.

3.1 Center-of-Mass Correlation Layers

One of the core problems with scaling up optical flow neural networks to larger inputs is memory constraints due to the correlation volume. For example, a search window of 33x33 results in a 1089-dimension cost volume. 1089 dimensions at a resolution of 1080p (1920x1080) results in 2GB of memory, just for the correlation volume. Further expanding the search window at that resolution increases the memory requirements quadratically. Additionally, based on the observation from [37], feature descriptors can be learned that allow for a center-of-mass calculation directly into a final flow estimate. Combining the center-of-mass observation with the concepts in

[31], where the l_2 distance is used to compare feature descriptors, a custom correlation layer can be created that compares feature descriptors and reduces them to a flow value within the same operation–skipping the memory expansion of a correlation volume. This paper abbreviates center-of-mass correlation as comcorr.

A generic center-of-mass calculation is shown in Equation (5), with *com* being the final center of mass, v being the 1-dimensional distance from the coordinate system origin, and w_v being the mass at a particular v.

$$com = \frac{\Sigma_v v w_v}{\Sigma_v w_v} \tag{5}$$

The l_2 distance d between two feature descriptors (t_1, t_2) of length c is given by equation Equation (6).

$$d = \sqrt{\Sigma_{i=0}^{c}(t_1[i] - t_2[i])^2} \tag{6}$$

The l_2 distance needs to be converted into a value that lends itself to a center-of-mass, where larger values indicate a stronger match, and smaller (non-zero) values indicate a weaker match. In other words, as the l_2 distance increases, it should be decreasing and approach zero. To satisfy this property, the inverse exponent is used, as shown in Equation (7), because for an l_2 distance of zero the output is 1, and for a large l_2 distance the output approaches zero. In this paper, α is referred to as the activation because higher values indicate a better feature match. The center-of-mass method is functionally equivalent to [37], which uses a negated cost and softmax. From there, α is multiplied by the distance away in both the u and v directions and summed up according to Equation (8), and each element's activation in the search window is summed into σ according to Equation (9). The sum of activations (σ) roughly corresponds to the confidence in each flow estimate, assuming that the

distribution of activations is unimodal, where values closer to 1 indicate a higher confidence.

$$\alpha = e^{-d} \tag{7}$$

$$\begin{aligned} f'_u &= \Sigma_u u * \alpha \\ f'_v &= \Sigma_v v * \alpha \end{aligned} \tag{8}$$

$$\sigma = \Sigma_u u \alpha \tag{9}$$

The proposed flow f'_u and f'_v is divided by the total activation σ in Equation (10), to get the final flow f_u and f_v–completing the center of mass calculation.

$$\begin{aligned} f_u &= \frac{f'_u}{\sigma} \\ f_v &= \frac{f'_v}{\sigma} \end{aligned} \tag{10}$$

The center of mass calculation is useful for several reasons:

1. The resulting flow will be more tolerant to noise due to the local information around each match
2. The correlation operation also effectively generates a confidence value with σ
3. Each operation is continuously differentiable

As the distance (d) between two feature descriptors approaches zero, the activation (α) approaches one ($e^{-0} = 1$). When looking for a high-confidence match, the overall activation (σ) should be close to or exceeding one–since it is a sum of all

activations. σ can be incorporated into a network architecture for more intelligent noise filtering and error correction. While this process was described specifically for optical flow estimation, it generalizes to any number of dimensions and any type of feature descriptor (e.g. audio, image, other signals of any dimensionality). With the comcorr process defined, it can now be incorporated into a variety of neural network layers for use in deep learning. For efficient processing, each operation is performed on the GPU with CUDA.

3.1.1 COMCorr Base Search

The comcorr base search starts off without any prior flow estimates, and searches image 2 for any match to image 1 given a user-defined search window. It has two outputs: a 2-channel flow image and a 1-channel activation image, representing the calculated optical flow (f_u, f_v) and the summed activation (σ). Since backpropagation through the activation gradient frequently led to poorly-converging models, it was disabled in favor of only optimizing the optical flow. The summed activation can still be used by a neural network, but the gradient is locked at zero. This is also true for the next comcorr-based layer: refinement.

3.1.2 COMCorr Refine

After the flow estimate is initialized, either using the aforementioned base layer or using another method, it is sometimes desired to refine that estimate to move towards the nearest local maximum[30]. For that purpose, the comcorr refine layer was created. It takes as input two c-channel images representing feature descriptors, the current optical flow estimate, a window size to search, and will output an updated flow estimate based on the search window around the current estimate. In order for this local search to work as intended, the estimate must already be close to the

correct value–which is not always true. To help make that true more often, a novel error correction layer is proposed.

3.1.3 COMCorr Error Correction

Since individual locations in an image (u, v) are highly correlated with their neighboring locations $(u+1, v+1)$, the optical flow tends to be very similar as well. With a notable exception being along object boundaries. The high correlation allows for a layer to take the flow estimates from each (u, v), as well as its local neighborhood defined by a window size, and determine which neighboring flow estimate works best given both images' feature descriptors. The neighboring flow estimate that minimizes the l_2 distance between the first and second images' feature descriptors is treated as the correct flow. The flow estimate at (u, v) is replaced with this newly-determined flow estimate. This error correction approach can also be used around object boundaries to fill in unknown pixels during upscaling[38], which is described in the next section.

3.1.4 COMCorr Upscaling

Due to the aforementioned spatial correlation of feature descriptors, new pixels can be filled in by evaluating how well the higher-resolution feature descriptor matches the flow estimates of its lower-resolution neighbors. A new pixel inserted on the border between two objects (Object A and Object B) has two possible flow values: one that is similar to Object A, and one that is similar to Object B, because optical flow tends to be fairly similar from pixel to pixel on the same object. When applying the same error correction procedure as Section 3.1.3 to this new pixel, both possible flow values are evaluated based on the new pixel's higher-resolution feature descriptor, and the flow value for the new pixel becomes the one with the smallest l_2 distance

between the higher-resolution feature descriptors. In other words, the new pixel is determined to belong with either Object A or Object B. The core purpose of this upscaling layer is to provide crisp object boundaries, and is used in preference to transposed convolutions or traditional upsampling because of its low memory usage and specific use of higher-resolution feature descriptors.

3.2 Improving Intrinsic Calibrations

Good intrinsic calibrations are the product of two important components: quantity of data and quality of data. In the case of OpenCV-based camera calibrations, the data consists of images containing the chessboard calibration pattern–collectively called the calibration images. The quantity of the data is the number of calibration images, and the quality of the data comes from accurate sub-pixel location estimations for each of the internal chessboard corners (the keypoints). With a set of calibration images, this section describes a method for increasing the number of calibration patterns found and detecting poor sub-pixel calibration pattern corners, starting with using image segmentation for increasing the quantity of chessboards found.

3.2.1 CNN for Chessboard Color Correction

A key property of calibration patterns, in particular the chessboards used for OpenCV calibrations, is the high contrast between the black checkers and the white border; however, in a standard lighting environment, the images will not show either of these as perfectly black (value = 0) or perfect white (value = 255)–complicating the pattern discovery. To aide OpenCV in finding them, a color correction Convolutional Neural Network (CNN) is proposed to locate the chessboard and ensure the checkers have a value close to 0 and the border has a value close to 255. The first step in this approach is to train a CNN to find the chessboard.

3.2.1.1 Data

The data used to train this CNN is composed of a randomly selected background and a randomly placed chessboard. The background is a random (512x512) grayscale crop of a high-resolution photo taken both indoors and outdoors, with a full-white and full-black background mixed in to encourage the network to look for the pattern rather than specific colors. The chessboard is randomly warped, randomly scaled, and contains a random number of rows and columns. Since this level of variety in the training data would result in a very large collection of images, the background is cropped in-memory and synthetically altered to include the chessboard by a custom image generator. The custom image generator also allows for accurate per-pixel truth data. An example of these images is shown in Figure 6. To augment the training data, the brightness is randomly scaled in the range (0.25, 1.25) and Gaussian noise with mean 0.0 and a random standard deviation between (0.0, 0.15) is added to the input images. The output from this network is a one-channel image with values of either 0 (background) or 1 (chessboard).

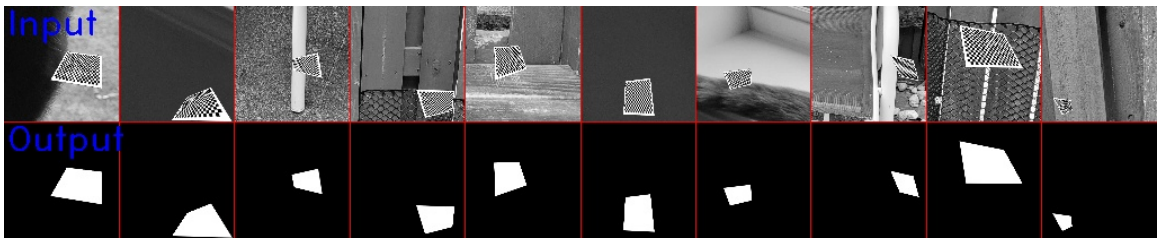

Figure 6: Synthetically generated training and test data for chessboard segmentation

3.2.1.2 Model and Training

The model is designed as a fully-connected U-Net architecture, with 5 downsampling bundles and 5 upsampling bundles to return to the original resolution. At each upsample, the previously calculated downsample is reused to refine the segmentation. The activation function for each downsample is Scaled Exponential Linear Unit

(SELU) (for self-normalizing properties) and at each segmentation regression the sigmoid activation function is used to constrain the output from 0 to 1. After each non-segmentation sigmoid convolutional layer, a batch normalization layer is added to ensure the network stays normalized.

The loss function of the network is Mean Squared Error (MSE), which takes the squared difference between the predicted output and the actual output. When squaring an error between 0 and 1, the squared value starts out very small and increases quadratically–resulting in much smaller parameter updates when the error is small, and slightly smaller parameter updates when the error is large. MSE was specifically chosen because of the use of the sigmoid activation function, where the output value will only approach 0 and 1, but never hit it exactly. If a network attempts to reach 0 or 1 exactly, it could result in exploding parameters as the output approaches the target.

The metric used to evaluate the model during training is Intersection over Union (IoU), which is the number of correctly estimated chessboard pixels (intersection) divided by the combination of actual chessboard pixels plus predicted chessboard pixels (union). IoU is used to focus in on the chessboard pixels, regardless of the overall size of the chessboard or the size of the image. For the optimizer, Adam is used with a learning rate of 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e - 7$ (all are defaults in Keras). After each iteration, the learning rate decays according to the function: $lr_i = 0.995 * lr_{i-1}$.

The computer used for training and evaluation is a AMD Ryzen 7 2700X with 16GB of system memory and a NVIDIA RTX 3080 with 10GB of device memory. Training takes approximately 30 minutes (10k iterations), with a batch size of 16, and a parameter count of 46,722.

3.2.1.3 Testing

The performance of this model is evaluated on 5 additional and unique photos, using the same 512x512 crop size but with a chessboard of size (rows=17, cols=24) and no data augmentation. Additionally, this model is used to perform the color correction of calibration images and is thus evaluated on its ability to make calibration images usable.

3.2.1.4 Color Correction

With a segmentation of the calibration image showing where the chessboard should be, the actual pixel values of the black and white regions can be calculated. The implementation used in this research is to first take the average pixel intensity of the chessboard, and call it *center*. Then, take the average pixel intensity below *center* and above *center*, and store it in *black* and *white* respectively. From there, the original image is scaled such that *black* becomes 0, *white* becomes 1, and any value outside that range is clipped to either 0 or 1 (to not have out-of-bound pixel values). An example of this equation is shown in Equation (11), and an example of the correction is shown in Figure 7.

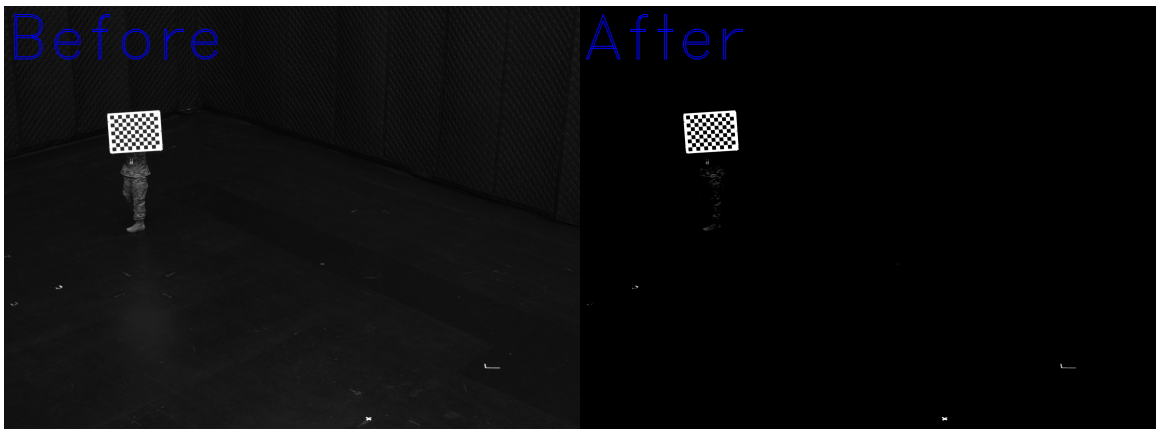

Figure 7: Example of color scaling and clipping for color correction

After color correction, the corners of the chessboard are found with the OpenCV

function “findChessboardCorners”. These corners are refined by the function “cornerSubPix”, which looks at the image gradient to get a sub-pixel estimate for where the corner is. Scaling the image with Equation (11) preserves these gradients because by definition they are between the black and white pixel values for the chessboard.

$$\begin{aligned} photo\_corrected = max(0., min(1., \\ (photo - black)/ \\ (white - black))) \end{aligned} \tag{11}$$

3.2.2 Chessboard Filtering Through Linear Regression

One common failure case when finding chessboard corners (the keypoints) is a bad match for a corner due to poor lighting conditions. When the corner is washed out, the estimated corner ends up at some location inside the checker, resulting in bad data for the intrinsic calibration. To detect this failure case, a novel chessboard filtration is proposed using a linear regression of each row and column of the estimated corners. A well-estimated chessboard will have each corner in a straight line, and any deviation from that line is a likely failure case. An example of this slight offset is shown in Figure 8 with several corners in the top row. The important exception to the straight-line intuition is when lens distortion is present, because the corners will follow a roughly parabolic shape. Depending on the calibration images, and the size/distance of the calibration pattern, it may be better to replace the linear regression with a quadratic regression.

After performing the linear regression for each row, the maximum and average distance from each corner to its corresponding line is recorded and saved along with the rest of the calibration data. When loading this data for use in an intrinsic calibration, the error values are used to filter out the bad chessboards. For this research,

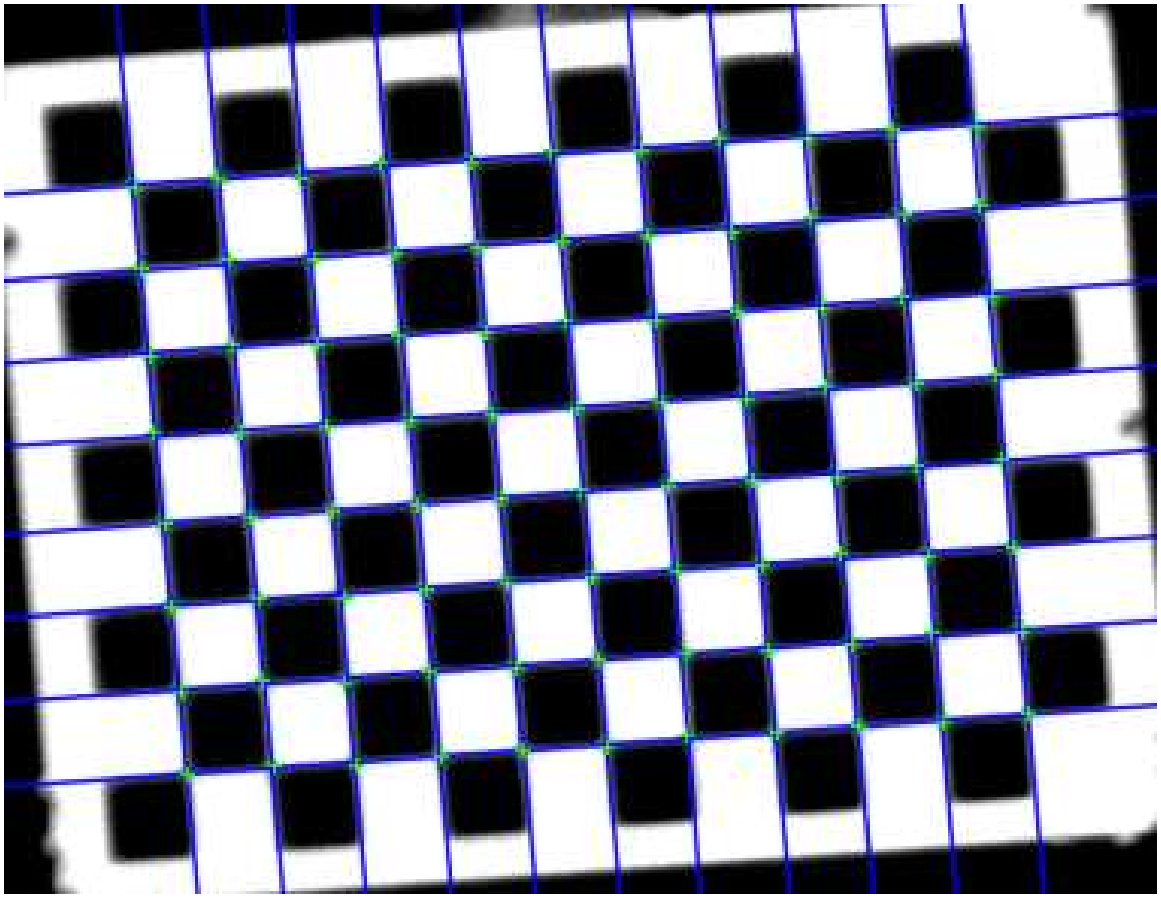

Figure 8: Linear regression on chessboard corners

a maximum error of 0.1 pixels is used and the average error is left unused.

3.3 Optical Flow for Extrinsic Calibrations

Stereo calibrations have two primary components: image rectification and reprojection. Image rectification is the process of putting all matching features on the same row of an image with the correct disparity. Reprojection with the Q matrix turns each (u, v, d) coordinate from Stereo Block Matching (SBM) into an (x, y, z). Optical flow is the process of finding matching features between two images, and for a properly-rectified stereo image pair the optical flow in the v-direction should be zero (indicating no vertical flow). Therefore, optical flow can be used to solve for the v-based rectification, which leaves the u-based (horizontal) rectification and the Q matrix. In order to solve for the horizontal rectification, the disparity between the left and right images must be known. There are two options for this:

1. Locate objects very far away, which have a disparity of zero
2. Use a calibration pattern to solve for both the chessboard position and the horizontal rectification

With the first option, the rectification can be accomplished entirely using optical flow–because matching features should have an optical flow of zero after rectification. For the experiment described in Section 3.4, there are no points "infinitely" far away–resulting in this research pursuing the second option. The first step of this process is to create an optical flow network that is capable of very accurately estimating flow.

3.3.1 Optical Flow Neural Network

A custom optical flow network was designed for this research to enable optical flow estimation on high-resolution (4K) images, which is particularly important for the experiment described in Section 3.4. To scale the network up to images of this size, the memory usage of the network was reduced through fewer convolutional filters

and the use of comcorr layers as described in Section 3.1. The data used to train the network is described in Section 3.3.1.1, the model itself is described and shown in Section 3.3.1.2, with the evaluation methodology described in Section 3.3.1.3.

3.3.1.1 Data

The data used for training this CNN is the Flying Chairs dataset, which as mentioned in Section 2.9.3.1, is composed of 22,872 512x384 image pairs, with ground truth optical flow between them. To enable learning high-resolution flow, the images are scaled up by 2x to a size of 1024x768 prior to augmentation and training. Both images are augmented in the following ways:

- Each color channel (RGB) is independently scaled by a random amount
- Gaussian noise is added with a mean of 0 and random standard deviation between 0 and 0.04
- Color channels are randomly swapped for both images (e.g. $B \leftrightarrow R$, $G \leftrightarrow B$, etc.)

Random noise and scaling are both augmentations that help the network learn brightness-invariant feature descriptors, while the color swapping is important for not overfitting to only plausible color combinations.

3.3.1.2 Model

The optical flow network is a combination of two fully-convolutional architectures. The first architecture is a U-Net architecture that is designed to create feature descriptors at a variety of image scales, and the second is responsible for creating and refining the flow estimate. The combination of both architectures is shown in Figure 9. The feature descriptor architecture has 6 downsampling bundles (orange) and

6 upsampling bundles (yellow), which create a receptive field of approximately 128 pixels at full resolution. The flow estimation architecture (green) has no parameters, and is composed entirely of comcorr layers–starting at the 6th layer of the feature descriptor output. The 6th layer has a resolution 32x smaller than the original images, and for training has a 9x9 search window to have a maximum search range of 128 pixels at full resolution. This search is accomplished by a base search comcorr layer, and is followed by an error correction layer to detect any clear mistakes in flow before upscaling. Then for each feature descriptor layer, starting with the 5th and working up to the 1st, there is a comcorr upscaling layer and a comcorr refine layer with a window size of 5x5. This combination of upscaling and refinement is designed to have better performance around object boundaries and to make any refinements as more information becomes available at higher resolutions. The final output after both networks are joined is a full-resolution flow estimation.

To keep the runtime down, especially for large images, each feature descriptor has just 12 channels. This number was chosen because it maximizes comcorr's use of shared GPU memory on most modern GPUs (48kB), while also still allowing for good feature matches. The value 1.7581 is added to each feature descriptor (generated with the SELU activation function), to keep each value positive. 1.7581 is approximately the minimum possible value of the SELU activation function. Additionally, since neural networks primarily encode feature descriptors as vectors where the angle is the most important distinguishing factor, each feature descriptor is normalized and multiplied by a network-learned constant prior to any comcorr layer. When the l_2 distance is later calculated between each feature descriptor, the normalization and scaling will effectively transform l_2 into an angular comparison. This component of the network is shown as the blue blocks in Figure 9.

The loss function is the l_1 distance between the predicted flow and the actual flow.

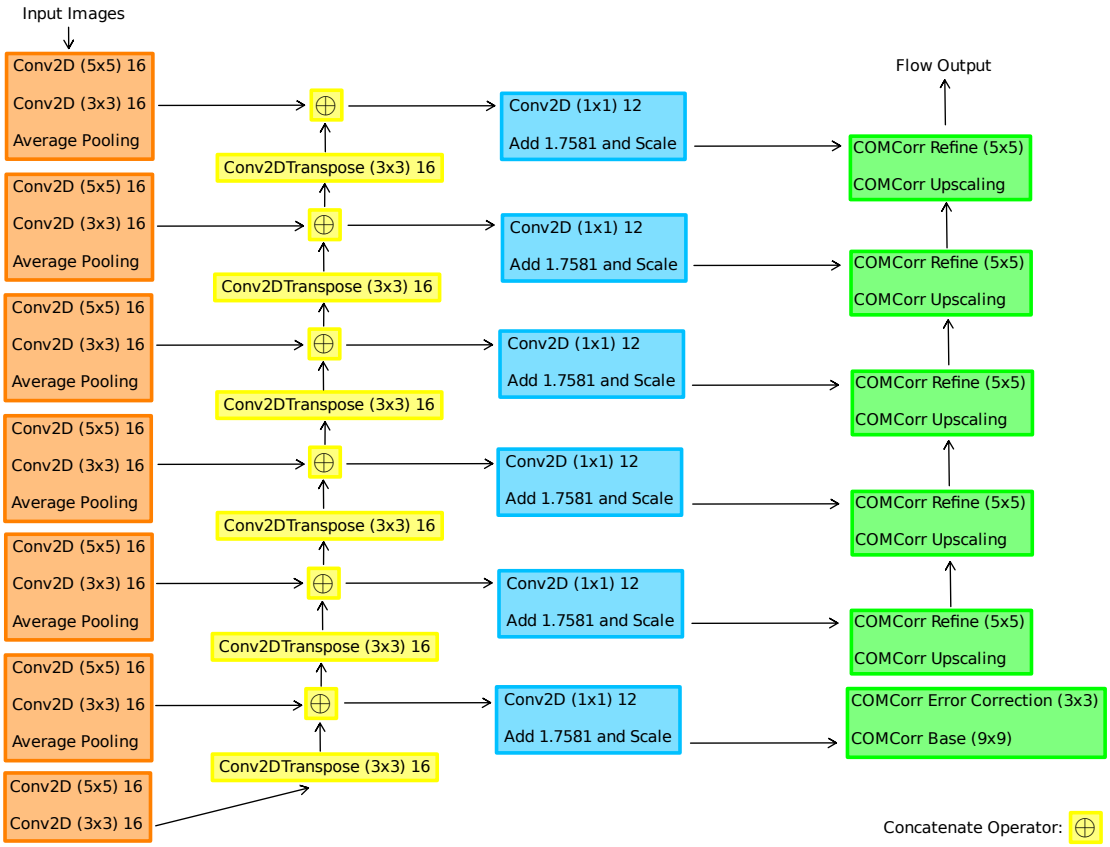

Figure 9: Proposed Optical Flow Architecture

The derivative of the l_1 loss is either 1 or -1, depending on the sign, which can help convergence with large flow values because the derivative is not scaled proportionally to the loss. The metric used to evaluate this network is called End Point Error (EPE). EPE is the l_2 distance between the actual flow and the predicted flow, and is the standard metric for evaluating optical flow. The Adam optimizer was used with a learning rate of $1e-4$, and default parameters for the remaining hyperparameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1e-7$.

The computer used for training and evaluation is a AMD Ryzen 7 2700X with 16GB of system memory and a NVIDIA RTX 3080 with 10GB of device memory. Training takes approximately 15 hours (114k iterations), with a batch size of 4, and a total parameter count of 83,774.

3.3.1.3 Testing

To evaluate the network, it will in part be evaluated on Sintel qualitatively, but its primary evaluation will be through its use in high-resolution stereo calibrations. A successful implementation should result in sub-pixel accuracy, allowing for SBM to generate a high-quality disparity map.

3.3.2 Stereo Calibration

The final stereo calibration is done by iteratively performing a least squares optimization using the optical flow as the vertical rectification error and the intrinsic calibration data is used to optimize the horizontal rectification as well as the Q matrix. Prior to each step, the intrinsic calibration for both cameras is used to undistort the calibration images as well as each of the (u, v) corner locations. After the undistortion, both images are transformed into a new unified camera matrix based off the left camera's intrinsic matrix. This step ensures that the input data follows the pinhole camera model as closely as possible prior to calibration. The final step is to determine the distance to each corner location (u, v) with Perspective-n-Point (PnP). PnP is a process that takes in the intrinsic calibration, the corner locations, and the relative location of each corner to solve for the overall calibration pattern's pose. It is available in OpenCV as the function solvePnP.

This calibration process optimizes two things: the rectification from the left image to the right image, and the baseline. Since the input contains the intrinsic calibrations, three of the four variables (c_x, c_y, and f) in the Q matrix are already known and don't need to be optimized. For the rectification, it's represented as a 3x3 matrix known as a homography. There are only 8 free variables, with the last variable being a scale factor. This results in 8 variables for the regression and 1 variable for the baseline, shown symbolically in Equation (12).

$$\begin{bmatrix} r_{0,0} & r_{0,1} & r_{0,2} & r_{1,0} & r_{1,1} & r_{1,2} & r_{2,0} & r_{2,1} & b \end{bmatrix} \tag{12}$$

Each step of the update algorithm is listed below:

1. Compute the left and right rectification from the full rectification (the homography H), which is performed as a $\frac{1}{2}$ interpolation forward (for the left image) and in reverse (for the right image)

2. Rectify the left and right images accordingly. This is done with OpenCV's function warpPerspective

3. Rotate the truth world points according to the rotation matrix in Equation (13), where H_l is the left rectification and M is the unified camera matrix

4. Rotate the (u, v) coordinates from the chessboard calibration for both the left and right camera with the rectification

5. Calculate the optical flow between the rectified images

6. Filter the optical flow output with three tunable parameters, depending on the environment:

    - Zero-out 480 pixels (in a 4K image with a disparity count of 512) on the left side of the flow image, since there can't be any matches there due to the disparity
    - Define a max error for this iteration - for the first iteration it is set to 1024 pixels, and then every two iterations it is divided by two until it hits 8. Any flow value larger than this is thrown out to help convergence
    - Filter out any matches with a comcorr activation (σ) less than 0.75

7. Append each remaining optical flow entry (u, v) to the least squares iteration with a Jacobian of Equation (14) and an error value of the v-flow

8. For each chessboard corner, take the original left (u, v) and calculate the Jacobian from Equation (15), with an error value defined by Equation (16) using the new rectified left (u_1) and right (u_2) coordinates, new world truth distance (z), and the current Q matrix's focal length (f) and baseline (b).

$$R = M^{-1} H_l M \tag{13}$$

$$\begin{aligned} num_y &= u * H[1,0] + v * H[1,1] + H[1,2] \\ denom &= u * H[2,0] + v * H[2,1] + H[2,2] \\ \left[0 \quad 0 \quad 0 \quad \frac{u}{denom} \quad \frac{v}{denom} \quad \frac{1}{denom} \quad \frac{-u*num_y}{denom^2} \quad \frac{-v*num_y}{denom^2} \quad 0\right] & \end{aligned} \tag{14}$$

$$\begin{aligned} num_x &= u * H[0,0] + v * H[0,1] + H[0,2] \\ denom &= u * H[2,0] + v * H[2,1] + H[2,2] \\ \left[\frac{u}{denom} \quad \frac{v}{denom} \quad \frac{1}{denom} \quad 0 \quad 0 \quad 0 \quad \frac{-u*num_x}{denom^2} \quad \frac{-v*num_x}{denom^2} \quad \frac{-f}{z}\right] & \end{aligned} \tag{15}$$

$$chess\_corner\_error = \frac{b * f}{z} - (u_1 - u_2) \tag{16}$$

This algorithm is run repeatedly until the rectification parameters (scale-based) $H[0,0]$, $H[0,1]$, $H[1,0]$, $H[1,1]$, $H[2,0]$, and $H[2,1]$ change by less than $1e-5$, the rectification parameters (shift-based) $H[0,2]$, and $H[1,2]$ change by less than $1e-1$, and the baseline changes by less than $1e-2$, or 30 iterations have elapsed, whichever

comes first. The result from this is a full rectification (homography from left to right) and a baseline, which needs to be converted into an OpenCV-compatible format using Equations (17) to (20). This calibration is evaluated according to the experiment defined in the next section.

$$left_r = M^{-1} H_l M \tag{17}$$

$$left_p = M \tag{18}$$

$$right_r = M^{-1} H_r M \tag{19}$$

$$right_p = M \tag{20}$$

3.4 Experiment

To evaluate the effectiveness of the proposed stereo calibration method, a simulated Automated Aerial Refueling (AAR) approach in a motion capture room was performed using both an OpenCV extrinsic calibration and the proposed optical flow-based calibration. Inside the motion capture room, a stereo camera is placed at a known position and rotation, facing a 0.148 scale receiver with motion tracking markers attached. The motion capture room provides the truth data for a simulated approach of the receiver to the stereo cameras, up to 75Hz and an accuracy within 1mm. The simulated approach consists of the receiver starting at approximately 19.8m away from the camera, coming in on approach to a distance of approximately 13.7m away, then pulling back to a distance of 19.5m from the camera. For evaluating the calibration, this method will be compared against OpenCV's stereoCalibrate method for pose estimation across the approach.

IV. Results and Analysis

4.1 Intrinsic Calibration Improvements

Intrinsic calibrations are improved in two ways: color correction is used to successfully identify more chessboards, and a linear regression is used for each row/column to throw out bad chessboards. The former is shown in Section 4.1.1, and the latter is shown in Section 4.1.2.

4.1.1 Color Correction

The final validation Intersection over Union (IoU) on 1000 test images without augmentation was 0.836 with a standard deviation of 0.189. An IoU of 0.5 means the network was just as correct as incorrect, and an IoU of 1.0 means the network was never wrong, so an IoU of 0.836 indicates the network was correct most of the time. Figure 10 shows the network performance qualitatively on the test data, where on most images the network very clearly identifies where the chessboard is in the image. The one notable exception is in the far right frame, where the bright background is illuminated in the predicted output. This artifact appears to be caused by the network correctly identifying the chessboard, then incorrectly predicting that the white background was part of the white border.

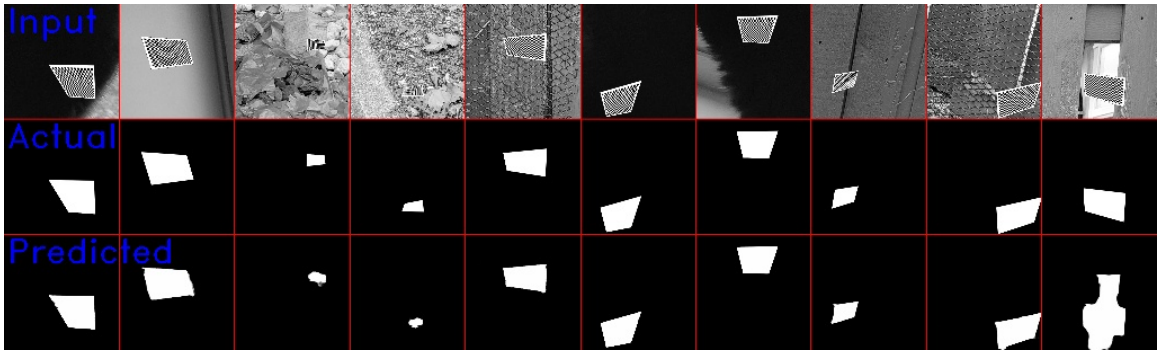

Figure 10: The color correction neural network on test data

The color correction process involves first finding the chessboard, then determining the black and white regions and scaling the brightness such that the black checkers have a pixel value of 0 and the white regions have a pixel value of 255. An example of that running on a calibration image is seen in Figure 11. After the adjustment, it is far more clear where the chessboard is.

Without color correction, OpenCV struggled to find the chessboards. After 24 hours of processing, only 60 images were processed–compared to just an hour to process over 1500 4K images with color correction enabled. Of the 60 images that were processed, OpenCV successfully found 34 of the 60 chessboards; with color correction, 58 of the 60 were found. Adding color correction results in a massive improvement, in both speed and quantity. There are two reasons for this: (1) significant portions of the image are clipped out, resulting in fewer pixels to search for the chessboard and (2) scaling the checkers between 0 and 255 makes it easier for OpenCV to correctly identify where each chessboard is because it directly fits the theoretical chessboard model.

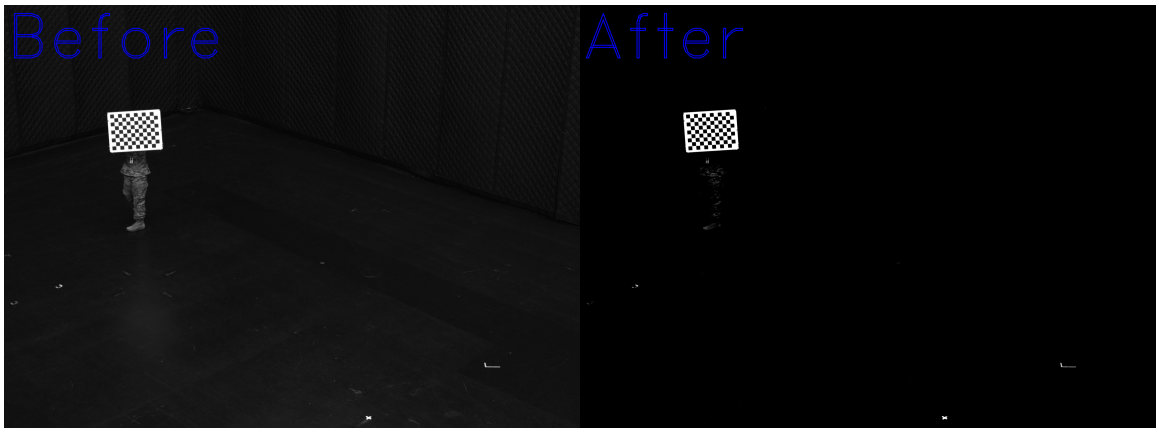

Figure 11: A calibration pattern before and after color correction

4.1.2 Linear Regression Filter

The linear regression filter creates a line of best fit between each row and column of the chessboard corners to find corner estimates that are "too far" off. A histogram of most of these max error values is shown in Figure 12, with the vast majority being under 0.2 pixels. There are several outliers up to 1 pixel, and then several more past that which are omitted for graph clarity. A max error of 0.1 tended to provide a nice balance between quality and quantity; although, given the number of images available for calibration in this case, it is probably more useful to select the smallest 100 rather than an arbitrary max error.

Figure 13 shows a comparison between a variety of maximum error values. An intrinsic and extrinsic calibration was run 30 times with 50 images randomly selected that satisfied the linear regression max error filter. Running the calibration 30 times created a range of reprojection errors, with some unfiltered calibrations resulting in very low reprojection errors, while others had very high reprojection errors. An interesting observation from Figure 13 was how decreasing the linear regression maximum error resulted in lower reprojection errors, and in the case of the intrinsic calibrations (left and right), also resulted in a lower standard deviation. With extrinsic calibrations, the trend slowed down at a reprojection error of one pixel. The reason that the extrinsic calibration didn't follow the same trend as the intrinsic calibrations is because the exact shape of the chessboard isn't as important as the locations of the chessboard in 3D-space. A standard OpenCV extrinsic calibration needs a large variety of locations and distances to properly calibrate the two cameras, which the linear regression filter does not help solve.

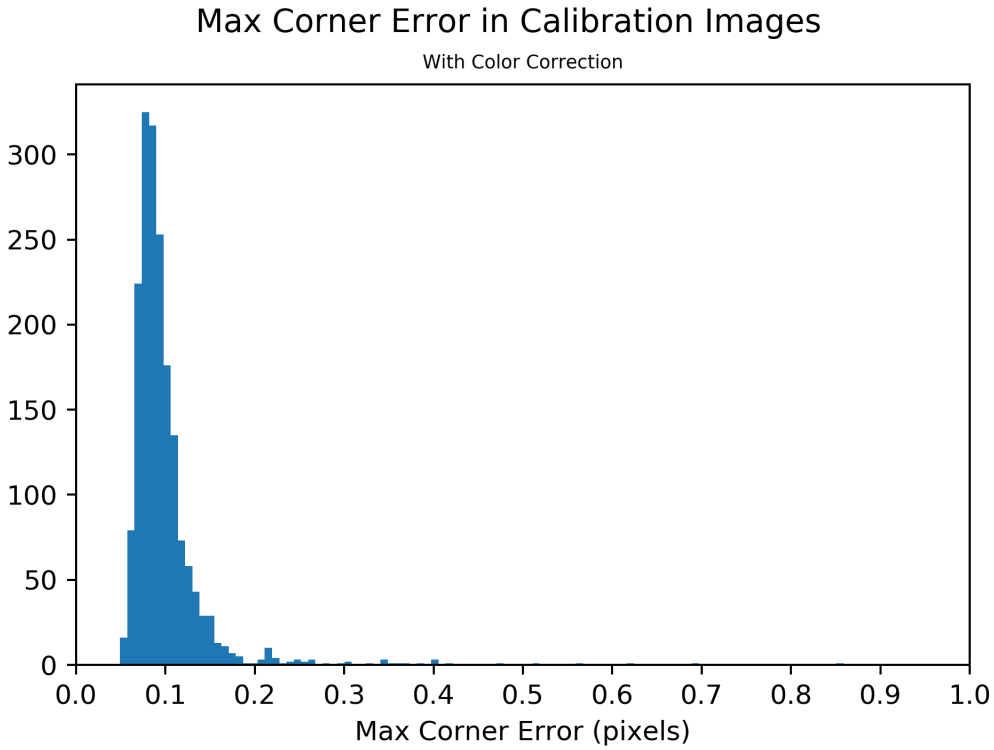

Figure 12: Histogram of max errors across all calibration images

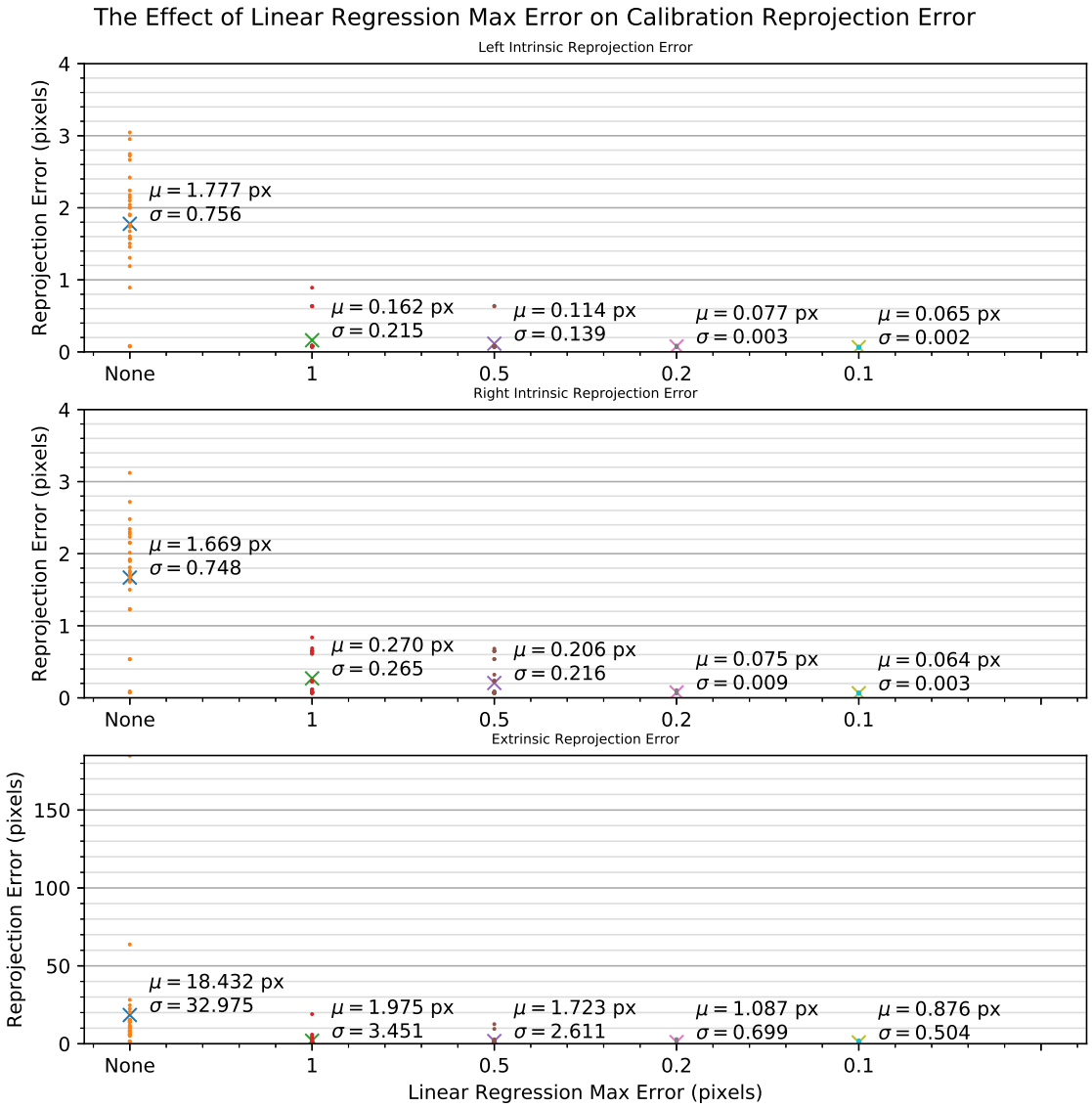

Figure 13: Evaluating linear regression max errors on final calibration error

4.2 Extrinsic Calibration with Optical Flow

Utilizing the intrinsic calibration from the previous section, the extrinsic calibration is performed using optical flow for the vertical alignment, and chessboards with a max error of less than 0.1 are used for the horizontal alignment and baseline optimization. Before evaluating optical flow's effectiveness for calibrations, its effectiveness is evaluated on the standard MPI-Sintel dataset[28].

4.2.1 Optical Flow Network

4.2.1.1 Performance on Sintel

MPI-Sintel is one of the most popular datasets for optical flow benchmarking due to its semi-realistic motion and high-quality ground truth training data. This network was only trained on flying chairs and not refined on Sintel, so the ground truth training data from Sintel can be used for evaluation. Quantitatively, the network's performance in End Point Error (EPE) is shown for each clip in Table 1 before and after an activation-based filter was applied. The filtered flow contains are all of the flow estimates with an activation (σ) greater than or equal to 0.75. Table 1 shows how the EPE drops significantly after the filter is applied. A higher activation indicates the feature descriptors have a smaller l_2 distance between them, and therefore are considered to be more similar. Using the filtered flow for calibrations, as is described in Section 3.3.2, should carry over this decreased EPE. Similarly, Table 2 shows how well the filtered network performs on a variety of flow magnitudes: from 0 to 10 pixels of movement, 10 to 40 pixels of movement, and 40+ pixels of movement. The network clearly performs best with small flow magnitudes (0-10 pixels), with larger flow magnitudes resulting in several pixels of error.

Two qualitative evaluations are shown in Figures 14 and 15. The network appears to perform particularly poorly around motion boundaries, which seems to be caused

Clip	**Average Flow**	**Average EPE**	**Filtered Flow**	**Filtered EPE**	**Average Pixels Used (%)**
alley_1	2.587	1.014	1.502	0.498	7.906
alley_2	6.656	1.805	5.893	0.504	4.056
ambush_2	63.696	46.539	51.493	29.836	0.968
ambush_4	31.988	28.575	17.245	5.272	1.873
ambush_5	22.531	16.980	10.723	6.981	4.262
ambush_6	39.209	29.006	32.872	11.606	2.824
ambush_7	4.169	2.827	0.520	0.558	16.073
bamboo_1	2.397	1.015	1.882	0.556	0.934
bamboo_2	2.371	1.969	0.913	0.326	14.749
bandage_1	3.499	1.698	0.565	0.298	36.043
bandage_2	2.124	1.107	0.177	0.113	34.738
cave_2	42.189	25.851	44.233	1.245	0.156
cave_4	15.390	9.673	12.896	1.041	0.209
market_2	2.689	1.923	1.048	0.268	14.339
market_5	38.784	27.709	18.394	1.773	2.010
market_6	19.770	12.314	9.710	1.047	2.203
mountain_1	4.979	1.967	1.920	0.965	16.001
shaman_2	1.763	0.687	0.092	0.182	25.587
shaman_3	2.787	1.225	3.666	1.606	4.124
sleeping_1	3.427	0.867	2.447	0.642	14.129
sleeping_2	2.347	0.560	1.698	0.369	2.058
temple_2	11.081	6.059	11.317	4.167	6.013
temple_3	37.119	24.703	27.950	5.188	15.473
Total	13.496	8.865	9.442	2.270	10.512

Table 1: Optical flow performance on each Sintel training clip before and after the activation filter

by the very low parameter count of the model. As the size of the network increases, its ability to correctly separate objects should increase as well. Figure 16 shows how much error there is across the image, and it becomes a lot more clear that the primary drawback of this network is its performance around motion boundaries. For the experiment in Section 4.2.3, the two images are a stereo image pair with smooth motion changes across most of the frame, so this drawback should not impact those results. The lower EPE for smaller flow magnitudes can also be seen in the background

Clip	s0-10	s10-40	s40+
alley_1	0.491	2.586	
alley_2	0.491	0.635	
ambush_2	5.386	11.488	37.554
ambush_4	1.978	9.972	37.106
ambush_5	3.402	5.786	10.410
ambush_6	4.969	10.402	17.784
ambush_7	0.553	1.327	2.933
bamboo_1	0.556	13.073	
bamboo_2	0.326	0.972	
bandage_1	0.268	2.670	
bandage_2	0.107	3.520	
cave_2	1.043	1.168	1.541
cave_4	0.980	1.145	2.616
market_2	0.241	1.306	1.619
market_5	2.766	1.664	4.878
market_6	0.970	2.474	1.474
mountain_1	0.945	2.907	58.821
shaman_2	0.182		
shaman_3	1.625	2.202	
sleeping_1	0.642		
sleeping_2	0.369		
temple_2	1.449	4.640	41.931
temple_3	2.320	4.319	18.280
Total	0.933	3.510	16.809

Table 2: Filtered optical flow performance on each Sintel training clip

of Figure 16, where the very small motion in the background is nearly perfect across the entire image. The good performance on smaller flow magnitudes indicates that the approach of repeatedly warping and re-calculating the flow should work well with this network.

4.2.2 Disparity Map

The disparity map is the first step in evaluating the quality of a calibration. A general rule of thumb is that the more non-zero disparities there are, the better the calibration. This rule of thumb isn't always the case, since the disparities could also

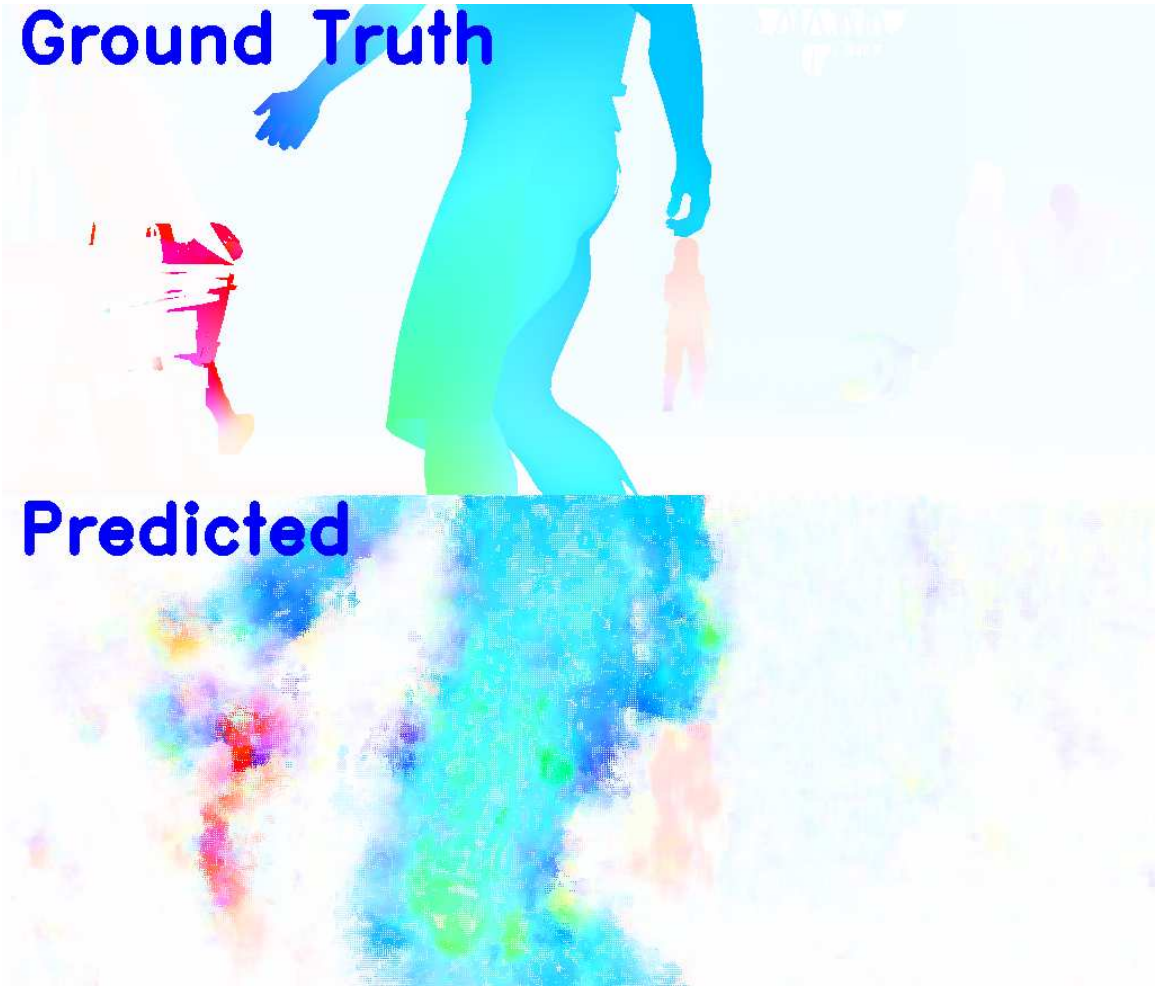

Figure 14: Optical flow performance on Sintel training clip "Market 2"

be incorrect, so the qualitative disparity analysis needs to be followed up with a 3D analysis to ensure correctness. In the case of the disparity maps in Figure 17, the proposed calibration has substantially more disparities across all parts of the image–indicating a large improvement on block matching performance. Figure 18 shows a zoomed in region where OpenCV struggled, but the proposed calibration succeeded. The green horizontal lines are the epipolar lines, where all matching features should line up. The red vertical line shows a particular u-coordinate for horizontal reference. While the proposed image pair has exactly the same feature in the same spot, enabling the Stereo Block Matching (SBM) match, the OpenCV

Figure 15: Optical flow performance on Sintel training clip "Bamboo 1"

calibration has the right image slightly shifted down, resulting in no match. This very small and subtle error is repeated across the top left of the image, resulting in very few disparity matches in that region. With optical flow aligning thousands of these features with varying depths and locations, it ensured a match was possible. The next important analysis is of the 3D reprojected points: evaluating both the estimated baseline and the quality of each match.

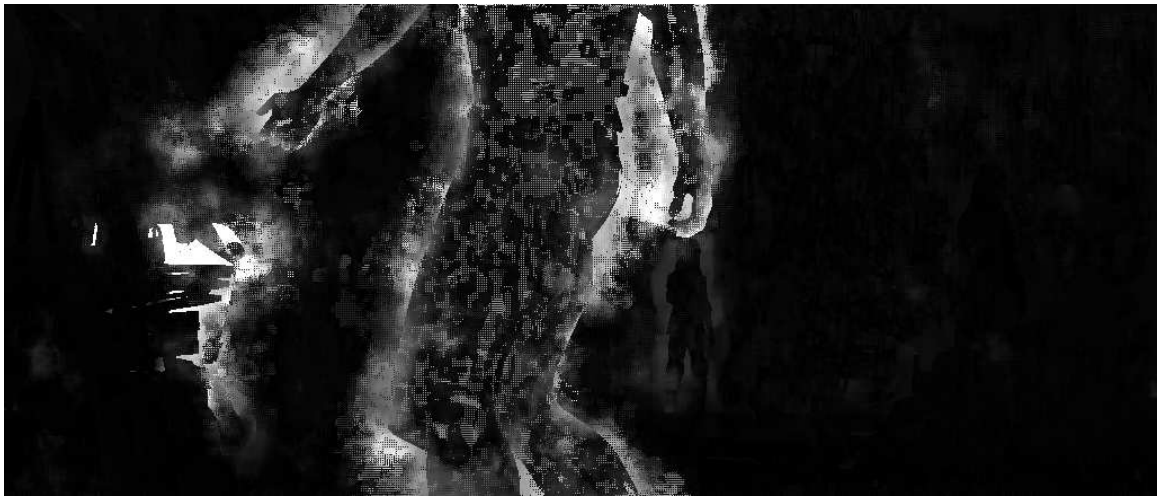

Figure 16: Normalized EPE on Sintel training clip “Market 2”

4.2.3 Pose Estimation Quality

The first analysis of the reprojected point cloud is qualitative in nature: how does the point cloud look, and what issues does it represent? The point cloud for both calibration methods is shown in Figure 19. The yellow points are the reprojected points, the red points are the output of Iterative Closest Point (ICP), and the textured model is the truth location of the receiver. OpenCV has fairly accurate reprojections, in particular along the rear stabilizers and along the body of the aircraft, although it falls apart near the wings and near the nose. The left wing has points disconnected from the wing entirely, while the right wing has points hidden under the wing - causing a roll in the ICP output.

On the other hand, the proposed calibration has a much cleaner point cloud, but still not perfect. The wings appear to have points relatively evenly distributed and there are no significant outliers, but nearly every point is hidden under the receiver. Figure 20 helps illustrate this issue well, with the translation error increasing nearly linearly as the receiver gets further away. This pattern indicates a calibration problem with the distance estimation, because the variance stays small while the error

increases. If the translation error was entirely due to camera resolution and distance from the camera, the mean error should stay approximately the same with increasing variance with distance. The roll error, on the other hand, decreases as the receiver gets further away. The roll error increasing could line up with the distance estimation error seen with the translation, since as the receiver approaches, one wing is much closer than the other, but it is surprising that the pitch and yaw error do not increase significantly along with it.

Returning to OpenCV's ICP error as seen in Figure 20, it is much harder to see distinguishable patterns from the translation error. It looks almost quasi-random, which indicates an issue with the vertical alignment. If SBM does not get a perfect match, it may get a match that is "good enough" close by that results in inconsistent matches from frame to frame. The rotational error, and in particular the roll, is very surprising. One possible explanation for it is a bad calibration in the v-direction, because that's the primary direction that the receiver moves through the frame, although it may be hard to concretely identify the problem with the inconsistent translational error.

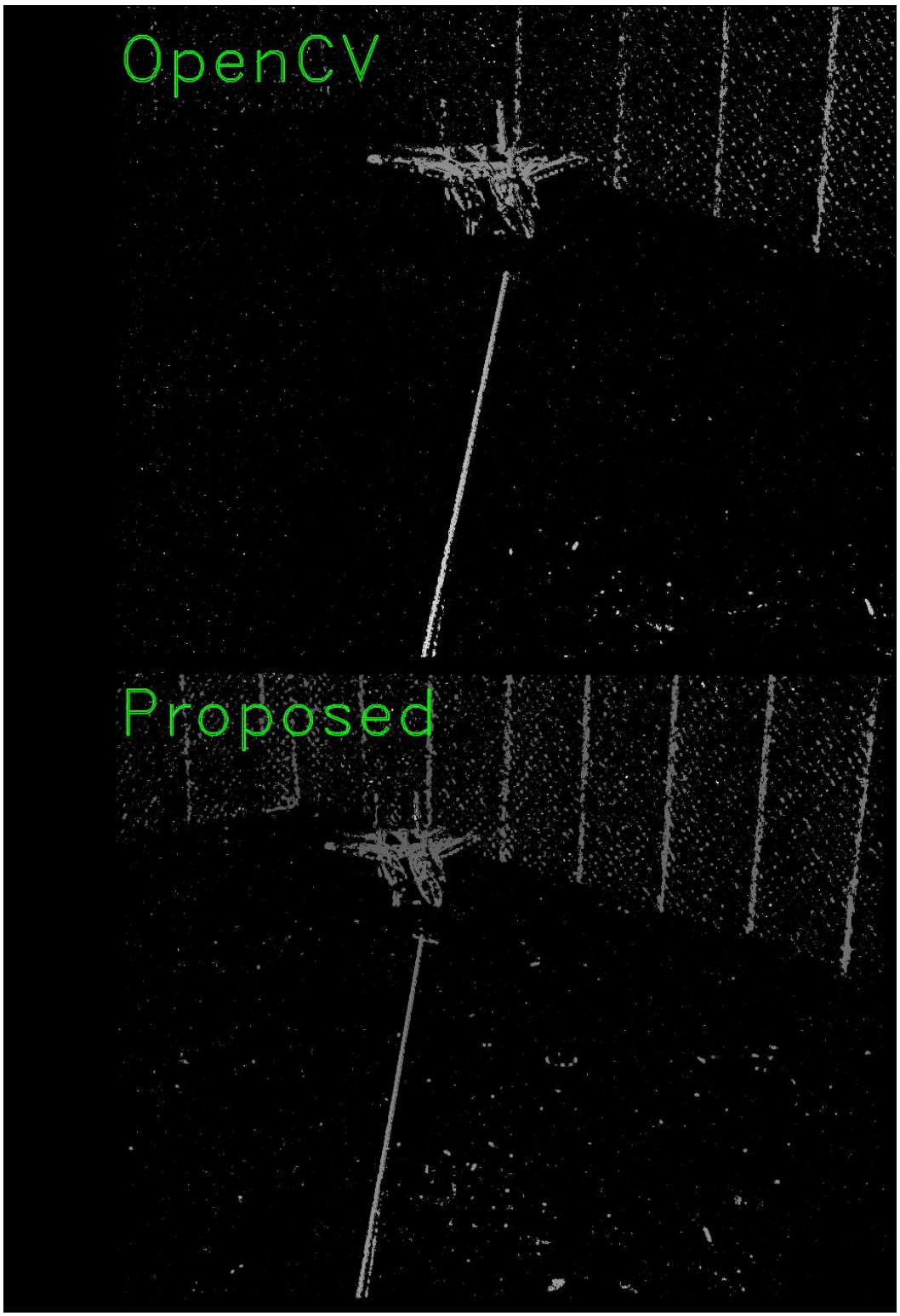

Figure 17: Disparity Map Comparison between OpenCV and Proposed

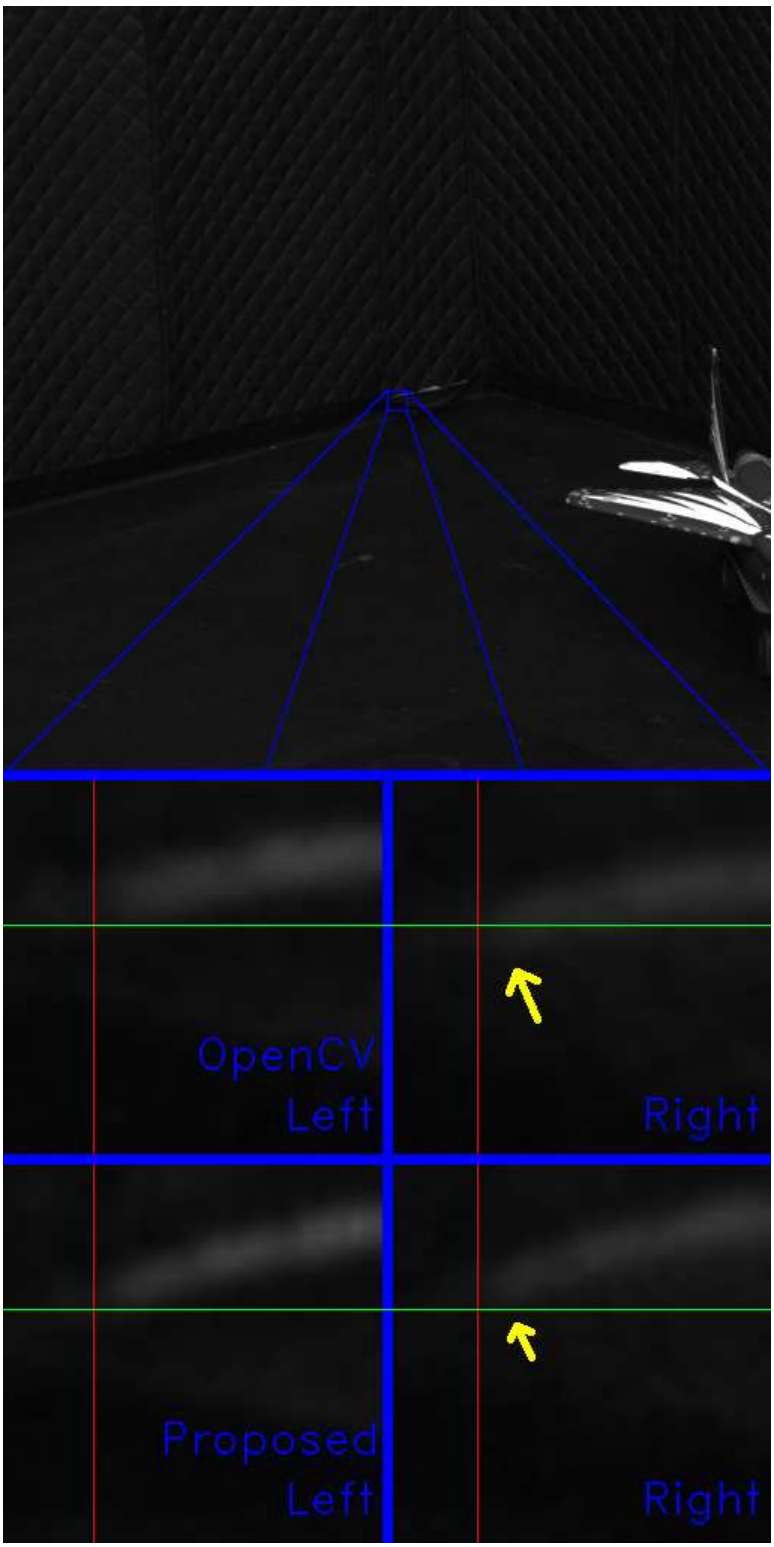

Figure 18: Epipolar Line Comparison between OpenCV and Proposed

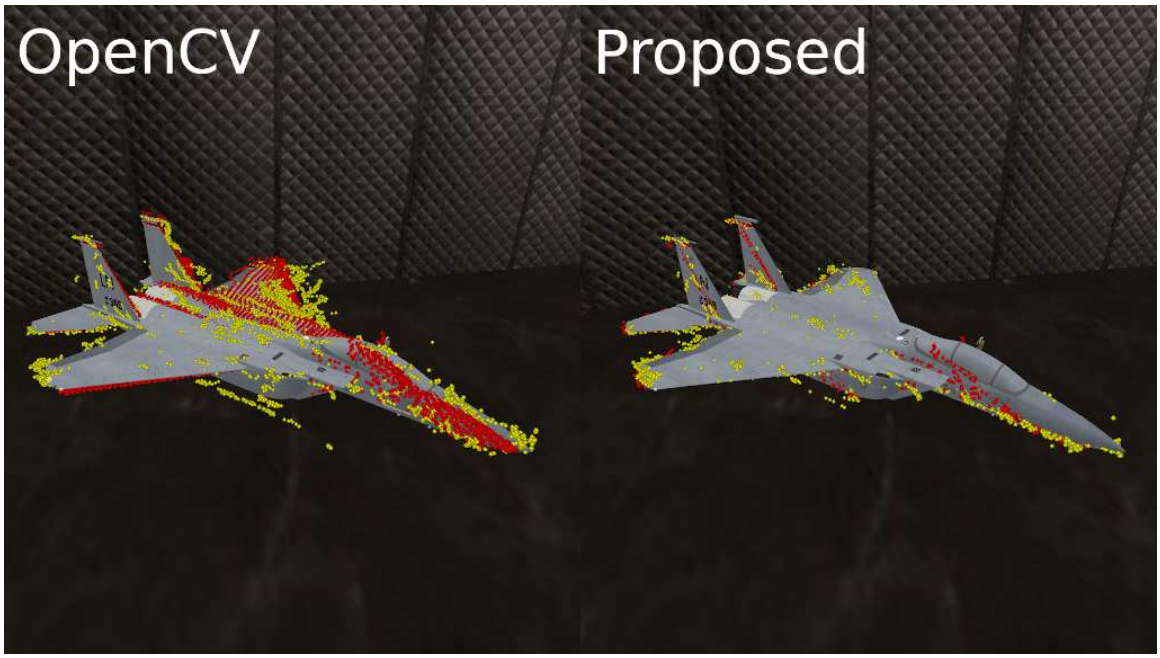

Figure 19: Point Cloud Comparison between OpenCV and Proposed

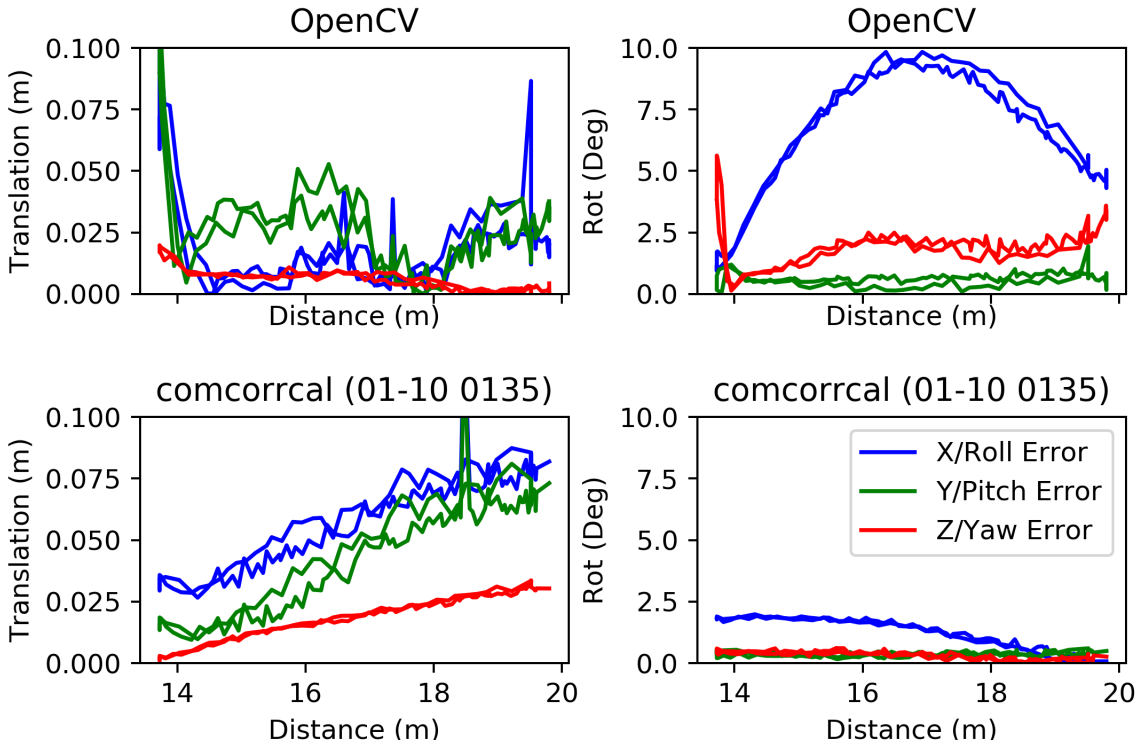

Figure 20: Pose Estimation Comparison between OpenCV and Proposed

V. Conclusions

This thesis proposes a novel calibration pipeline to improve the precision of stereo calibrations. The first step is the intrinsic calibration, with color correction to increase the quantity of calibration images and the linear regression filter to increase the quality of calibration images. The second step is the extrinsic calibration, with optical flow to perform the vertical alignment required for Stereo Block Matching (SBM) and the calibration patterns for the horizontal alignment and baseline estimation.

The results from Section 4.1.1 show how the color correction increases the quantity of viable calibration images, while Section 4.1.2 shows that extreme outliers can be trivially filtered out using the linear regression max error. The combination of the two was fed into an intrinsic calibration, which achieved a reprojection error of just 0.06 pixels–compared to an average intrinsic reprojection error of 1.669 pixels prior to this work. Finally, using that intrinsic calibration and optical flow, a stereo calibration was generated that outperformed OpenCV on vertical alignment and reprojection precision–as is seen in Figure 20.

Stereo calibration with optical flow can be further augmented to run without chessboard patterns as well, so long as the distance to several pixels are known. An example of this with Automated Aerial Refueling (AAR) is online stereo calibrations when in-flight, because the distance to the ground is infinite from the perspective of the cameras. This allows for a complete rectification to be performed without any input from the pilot or technician, and the Q matrix can be further optimized to minimize the reprojected point cloud error on the receiver aircraft.

5.1 Future Work

While this thesis improves stereo calibrations, there is more work that can be done to further improve upon it and answer additional questions.

- Change the corner linear regression to a higher-order regression and analyze the change in intrinsic calibration quality. Since the linear regression does not tolerate very much lens distortion before a chessboard is tossed out, a higher-order model should be evaluated as well–in particular on high-distortion lenses such as fisheye lenses.

- Evaluate the optical flow-based rectification on aerial imagery with Iterative Closest Point (ICP) for optimizing the reprojection matrix.

- Improve the optical flow network for use in high-resolution images with lots of non-continuous motion. While the proposed network works well for the final experiment described in this thesis, it may not work well in every situation. Improving the generalizability of this network would help its adoption in many more scenarios.

- Change the color correction network to recognize different calibration patterns, and evaluate the linear regression filter on these new patterns. An example of such a pattern is given in [8].

Bibliography

1. Edwards completes tests to extend KC-135 > U.S. Air Force > Article Display.

2. OpenCV: Camera Calibration and 3D Reconstruction.

3. OpenCV: Epipolar Geometry.

4. S. C. de Vries. UAVs and control delays. Technical report, Sep 2005.

5. Christopher Parsons, Zachary Paulson, Scott Nykl, William Dallman, Brian G. Woolley, and John Pecarina. Analysis of Simulated Imagery for Real-Time Vision-Based Automated Aerial Refueling. *Journal of Aerospace Information Systems*, 16(3):77–93, Mar 2019.

6. Bradley French. *Determining Virtual Practicality From Physical Stereo Vision Images and GPS*. PhD thesis, Air Force Institute of Technology, 2020.

7. Andrew Lee, Will Dallmann, Scott Nykl, Clark Taylor, and Brett Borghetti. Long-Range Pose Estimation for Aerial Refueling Approaches Using Deep Neural Networks. *Journal of Aerospace Information Systems*, 17(11):634–646, Nov 2020.

8. Thomas Schöps, Viktor Larsson, Marc Pollefeys, and Torsten Sattler. Why Having 10,000 Parameters in Your Camera Model is Better Than Twelve. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2532–2541, Dec 2019.

9. Yonggen Ling and Shaojie Shen. High-Precision Online Markerless Stereo Extrinsic Calibration. Technical report.

10. Paul J. Besl and Neil D. McKay. Method for registration of 3-D shapes. In Paul S. Schenker, editor, *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–606. SPIE, Apr 1992.

11. Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep Iterative Matching for 6D Pose Estimation. *International Journal of Computer Vision*, Mar 2018.

12. Tomas Hodan, Daniel Barath, and Jiri Matas. EPOS: Estimating 6D Pose of Objects with Symmetries. Technical report, 2020.

13. Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. Nov 2017.

14. Shih En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 4724–4732. IEEE Computer Society, Dec 2016.

15. Haoran Wei, Yue Zhang, Bing Wang, Yang Yang, Hao Li, and Hongqi Wang. X-LineNet: Detecting Aircraft in Remote Sensing Images by a pair of Intersecting Line Segments. Jul 2019.

16. Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *Proceedings of 2017 International Conference on Engineering and Technology, ICET 2017*, volume 2018-January, pages 1–6. Institute of Electrical and Electronics Engineers Inc., mar 2018.

17. Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, Dec 2015.

18. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning, ICML 2015*, volume 1, pages 448–456. International Machine Learning Society (IMLS), Feb 2015.

19. Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-Normalizing Neural Networks. In *International Conference on Neural Information Processing Systems*, pages 972–981, 2017.

20. Ying Tai, Jian Yang, and Xiaoming Liu. Image Super-Resolution via Deep Recursive Residual Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3147–3155, 2017.

21. Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway Networks. *CoRR*, abs/1505.0, May 2015.

22. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully Convolutional Networks for Semantic Segmentation. Technical report, 2015.

23. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9351, pages 234–241. Springer Verlag, 2015.

24. Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

25. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceed-*

ings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 580–587. IEEE Computer Society, Sep 2014.

26. Yue Zhang, Xianrui Li, Mingquan Lin, Bernard Chiu, and Mingbo Zhao. Deep-recursive residual network for image semantic segmentation. *Neural Computing and Applications*, 32(16):12935–12947, Aug 2020.

27. Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Hazırbas¸ Hazırbas¸, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. Technical report, 2015.

28. Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7577 LNCS, pages 611–625. Springer, Berlin, Heidelberg, 2012.

29. Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation? *International Journal of Computer Vision*, 126(9):942–960, Sep 2018.

30. Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12347 LNCS:402–419, Mar 2020.

31. Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and

Francesc Moreno-Noguer. Discriminative Learning of Deep Convolutional Feature Point Descriptors. Technical report, 2015.

32. Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. Technical report, 2017.

33. Jia-Ren Chang and Yong-Sheng Chen. Pyramid Stereo Matching Network. Technical report, 2018.

34. Anurag Ranjan and Michael J Black. Optical Flow Estimation using a Spatial Pyramid Network. Technical report, 2017.

35. Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation. Technical report, 2018.

36. Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz Nvidia. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. Technical report, 2018.

37. Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-End Learning of Geometry and Context for Deep Stereo Regression. Technical report, 2017.

38. Bruce D Lucas and Takeo Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Imaging Understanding Workshop*, pages 121–130, 1981.

Acronyms

6DoF 6 Degree-of-Freedom. 14

AAR Automated Aerial Refueling. iv, 1, 2, 4, 5, 10, 38, 54, 1

AFIT Air Force Institute of Technology. 1, 5

CNN Convolutional Neural Network. iv, 6, 14, 15, 18, 25, 26, 32, 1

EPE End Point Error. viii, 34, 44, 45, 49

GCS Ground Control Station. 4

GPS Global Positioning System. 1, 4

ICP Iterative Closest Point. 5, 49, 50, 55

IoU Intersection over Union. 27, 39

MSE Mean Squared Error. 27

PnP Perspective-n-Point. 35

RPA Remotely Piloted Aircraft. iv, 1, 4, 1

SAD Sum of Absolute Differences. 12

SBM Stereo Block Matching. 10, 12, 31, 35, 47, 50, 54

SELU Scaled Exponential Linear Unit. 15, 26, 33

SGD Stochastic Gradient Descent. 15

USAF United States Air Force. 1, 4

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
25–03–2021	Master's Thesis	Sept 2019 — Mar 2021

4. TITLE AND SUBTITLE

Stereo Camera Calibrations with Optical Flow

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Joshua D. Larson

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-21-M-056

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RQQC
Dan Schreiter
WPAFB OH 45433-7765
COMM 937-938-7765
Email: dan.schreiter@us.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/RQQC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

RPA are currently unable to refuel mid-air due to the large communication delays between their operators and the aircraft. AAR seeks to address this problem by reducing the communication delay to a fast line-of-sight signal between the tanker and the RPA. Current proposals for AAR utilize stereo cameras to estimate where the receiving aircraft is relative to the tanker, but require accurate calibrations for accurate location estimates of the receiver. This paper improves the accuracy of this calibration by improving three components of it: increasing the quantity of intrinsic calibration data with CNN preprocessing, improving the quality of the intrinsic calibration data through a novel linear regression filter, and reducing the epipolar error of the stereo calibration with optical flow for feature matching and alignment. A combination of all three approaches resulted in significant epipolar error improvements over OpenCV's stereo calibration while also providing significant precision improvements.

15. SUBJECT TERMS

Aerial Refueling, Optical Flow, Computer Vision

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Scott L. Nykl, AFIT/ENG
U	U	U	UU	73	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4395 scott.nykl@afit.edu