Learned Adjustment of Camera Gain and Exposure Time for Improved Visual Feature Detection and Matching

by

Justin Tomasi

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

Learned Adjustment of Camera Gain and Exposure Time for Improved Visual Feature Detection and Matching

Justin Tomasi
Master of Applied Science
Graduate Department of Aerospace Science and Engineering
University of Toronto
2020

Ensuring that captured images contain useful information is paramount to successful visual navigation. In this thesis, we explore a data-driven approach to account for environmental lighting changes, improving the quality of images for use in visual odometry (VO). We investigate what qualities of an image are desirable for navigation through an empirical analysis of the outputs of the VO front end. Based on this analysis, we build and train a deep convolutional neural network model to predictively adjust camera gain and exposure time parameters such that consecutive images contain a maximal number of matchable features. Our training method leverages several novel datasets consisting of images captured with varied gain and exposure time settings in diverse environments. Through real-world experiments, we demonstrate that our network is able to anticipate and compensate for lighting changes and maintain a higher number of inlier feature matches compared with competing camera parameter control algorithms.

Acknowledgements

Although much of the graduate school experience is largely independent, the culmination of the work and effort resulting in a thesis is hardly the accomplishment of a single individual. If it takes a village to raise a child, then it takes an even larger, more patient village to 'raise' a graduate student—in my experience, anyway. The following work would not have been possible without the support, guidance, help, and patience of a large group of people. I would like to take the opportunity to thank those individuals who made this work possible. I would first and foremost like to thank my advisor, Jonathan Kelly, who took a chance and welcomed me into his lab so that I could switch career paths and pursue a passion of mine. Jon, your mentorship and honest feedback were crucial in completing this work, and your guidance allowed me to grow and succeed academically. The culture you have fostered in the lab made my experience in graduate school incredibly fun and rewarding. I would also like to thank Steven Waslander for reviewing this document and providing valuable feedback. Thank you to my labmates: Val, Lee, Brandon, Matt, Oliver, Trevor, Filip, Olivier, Adam, Emmett, Abhinav, Chris, and Juraj, who I could always count on to provide helpful input and thoughtful discussions regarding my work and graduate school in general. More importantly, however, were the endless debates, jokes, and laughs we shared in the lab and elsewhere. You all enriched my experience by making the bad research days good, and the good days better. Brandon, you in particular, were an endless source of guidance, insight, and assistance. Thank you for your continual feedback and patience in fielding all of my questions. To my family: Mom, Dad, Mackenzie, and Cody, thank you for your limitless encouragement, support, and love. I could always count on you to raise my spirits with phone calls and during visits to London. You have shaped me into the person I am today and I am incredibly lucky to have you all in my life. Oonagh, thank you for your constant optimism, interest in my work, and for selflessly assisting with parts of the data collection and experimental tests. Norm, thank you for the positive encouragement and advice, for allowing me to bounce ideas off of you, and for much needed distractions during long days. Finally, there is an unending list of thanks I owe to Sarah. Your patience, support, and wisdom have been my sources of strength and inspiration. Thank you for assisting with much of the data collection and experiments, but more importantly, for always being there for me. I cannot put into words the amount that your warmth, compassion, humour, and love have brightened my days; thank you for everything.

Contents

1 Introduction **1**

1.1 Techniques for Improving Visual Perception . . . 3

1.2 Improving Image Quality for VO . . . 4

1.3 Contributions . . . 5

2 Background **7**

2.1 Cameras and Camera Parameters . . . 7

2.1.1 The Perspective Camera and Geometric Image Formation . . . 7

2.1.2 Camera Parameters and the Image Formation Process . . . 9

2.2 Visual Odometry . . . 12

2.2.1 The VO Pipeline . . . 14

2.2.2 Image Features and Matching . . . 18

2.2.3 Challenges in VO . . . 20

2.3 Deep Learning . . . 21

2.3.1 Deep Feedforward Networks . . . 22

2.3.2 Convolutional Neural Networks . . . 24

3 Related Work **26**

3.1 High-Dynamic Range Imaging . . . 26

3.2 Reactive Control of Camera Parameters . . . 27

3.3 Heuristic Image Metrics . . . 33

4 Learned Camera Parameter Control **35**

4.1 Problem Formulation . . . 35

4.2 Approach and Methodology . . . 37

4.2.1 Network Architecture . . . 37

4.2.2 Dataset Collection . . . 40

4.2.3 Using Image Feature Count as a Metric . . . 48

4.2.4 An Analysis of Image Metrics Derived from VO 50
4.2.5 Target Generation and Training Procedure 56

5 Experimental Results **60**
5.1 Experiment Details 60
5.1.1 Camera and Hardware Considerations 60
5.1.2 Experimental Setup 63
5.2 Real-World Experiments 64
5.2.1 Single-Parameter Case 64
5.2.2 Dual-Parameter Case 72
5.2.3 `Libviso2` Image Processing Experiment 81
5.3 Discussion 82

6 Conclusion **84**
6.1 Summary and Contributions 84
6.2 Potential Improvements 86
6.3 Future Work 86

Bibliography **88**

List of Tables

4.1 Network architecture and processing comparison 39
4.2 Comparison of image metrics (median matches) 53
4.3 Comparison of image metrics (minimum matches) 54
4.4 Comparison of selected metrics with metrics from literature 56

5.1 Single-parameter network performance in static lighting conditions 65
5.2 Single-parameter network performance in dynamic lighting conditions . . 69
5.3 Dual-parameter network performance in static lighting conditions 75
5.4 Dual-parameter network performance in dynamic lighting conditions . . . 78
5.5 Dual-parameter `libviso2` frame processing analysis 81

List of Figures

2.1 Idealized perspective monocular camera 8
2.2 Examples of various camera apertures 10
2.3 Examples of a scene captured with various exposure values 10
2.4 Examples of motion blur due to exposure 11
2.5 Examples of a scene captured with various gain values 11
2.6 Camera response function 12
2.7 Processing blocks of the VO pipeline 15

3.1 Comparison of metrics across an image sequence 34

4.1 Single-parameter CNN architecture 39
4.2 Dual-parameter CNN architecture 41
4.3 Examples of poses from the single-parameter dataset 43
4.4 Examples of trajectories from the single-parameter dataset 44
4.5 Camera rig and parameter space sampling diagram 46
4.6 Examples of poses from the dual-parameter dataset 47
4.7 Essential and rotation matrix estimation accuracy per number of features 49
4.8 Comparison of image target trajectories 55

5.1 Camera wiring diagram and setup 62
5.2 Test route diagram and tunnel example 63
5.3 Single-parameter images captured in static lighting conditions 66
5.4 Single-parameter inlier feature matching in static lighting conditions . . . 67
5.5 Single-parameter inlier feature matching in dynamic lighting conditions . 70
5.6 Single-parameter images captured in dynamic lighting conditions 71
5.7 Feature matching comparison of metrics in static lighting conditions . . . 73
5.8 Feature matching comparison of metrics in dynamic lighting conditions . 73
5.9 Target profiles of proposed target generation methods 74
5.10 Dual-parameter images captured in static lighting conditions 76

5.11 Dual-parameter inlier feature matching in static lighting conditions . . . 77
5.12 Dual-parameter images captured in dynamic lighting conditions 79
5.13 Dual-parameter inlier feature matching in dynamic lighting conditions . . 80

Chapter 1

Introduction

Visual egomotion estimation is the process by which a robot, using one or more on-board cameras, determines its motion through an environment relative to a fixed frame of reference. Egomotion estimation is typically achieved by acquiring a continuous stream of images as the robot moves. Captured images are passed as input to a visual egomotion *pipeline*. The pipeline processes the images and outputs an estimate of the change in the 3D translation and rotation (or "pose") of the robot between sequential image frames.

The use of cameras to estimate the rigid-body motion of a robot was first proposed and investigated in the 1960s. In the early 1980s, Moravec published his groundbreaking thesis on visual navigation [1]. While initially coupled to progress in the field of computer vision, visual egomotion estimation has since evolved into a largely independent field of robotics research. Over the decades, many important advances have been made, which have resulted in successful implementations and applications in a variety of environments including ground, air, underwater, and in space. Visual egomotion estimation has reached a level of maturity in that many of the fundamental concepts are now well understood and much of the focus has shifted to increasing the robustness of the pipeline [2].

With the ever-increasing interest in the realization of autonomous vehicles (AVs) for commercial purposes, many AV companies are motivated to reduce costs while expanding the capabilities of their machines. AVs are required to operate in a reliable and safe manner in a wide variety of environments and conditions. Consequently, robust sensing is crucial. One of the primary costs associated with the mass-production of AVs is the suite of active and passive sensors required for each vehicle to perceive its surroundings. AV companies have focused on both reducing costs associated with sensing and maximizing the efficacy of selected sensors. Cameras, in particular, provide a rich stream of data for use in a variety of navigation, motion estimation, and localization algorithms. Additionally, cameras are relatively inexpensive compared to other common sensors such

as LIDAR units. As a result, cameras have become a popular choice for use on board AVs.

Successful visual navigation depends on the acquisition of images that contain sufficient information. Image information can be defined loosely as a measure of the 'content' in an image that is useful for a visual task. In the case of egomotion estimation or localization, this information might be the number of unique and identifiable features, the magnitudes of the image gradients [3], or the spatial entropy [4]. Most visual egomotion estimation pipelines operate under the assumption that the inputs contain sufficient information, which has led to a dependency on high-quality, well-exposed images. The quality of captured images is not only dependent on environmental factors such as structure, ambient lighting, and the speed of the robot platform, but is also dependent on the camera parameter settings used during image acquisition. Although the former cannot always be directly controlled, camera parameters can be actively adjusted to improve the quality of images acquired under challenging environmental conditions.

Two of the primary camera parameters that directly impact the quality of images captured are *exposure time* and *gain*. The exposure time is the length of time that the shutter allows light to fall on the sensor. The shutter traditionally refers to a covering over the imaging sensor that mechanically opens and closes to expose the sensor to the scene. However, the use of a mechanical shutter in high-frame rate cameras is no longer common as the shutter wears out and its speed is limited. Machine vision cameras now employ electronic shutters that control the length of time sensor pixels accumulate photons. Camera gain is an electronic amplification of the signal generated by the imaging sensor during image acquisition. The size of the camera lens opening (or *aperture*), combined with the camera exposure time, determines how much light strikes the imaging sensor during image capture and can also be manually controlled. We describe camera parameters in more detail in Section 2.1.

For many robotics applications, the lens aperture is typically set to a fixed value while the exposure time and gain are either automatically adjusted (using built-in automatic parameter control algorithms) or manually set to a fixed value. Making use of the built-in automatic algorithms is usually adequate for vision applications in environments where lighting conditions are relatively static. However, relying on fixed parameter values or built-in automatic control algorithms can result in poorly-exposed images in situations where lighting conditions are dynamic [3]. One reason for the poor performance of built-in parameter controllers is that they operate in a *reactive* manner. Adjustments are made only after a large change in overall image brightness has been recorded, which is too late to prevent the loss of valuable information (caused by overexposure or underexposure).

For example, images can become improperly exposed due to sudden glare from the sun, during transitions between indoor and outdoor environments, and when moving into and out of tunnels. Indeed, changes in brightness of up to 120 dB are common in the case of tunnel transitions [5]. Additionally, built-in automatic parameter control algorithms are typically proprietary 'black boxes' that are intentionally obfuscated and difficult to interpret.

1.1 Techniques for Improving Visual Perception

There are three main approaches to increasing the robustness of visual navigation algorithms operating in dynamic lighting conditions [4]. The first is the application of post-processing techniques that perform pixel-wise transformations on images in an attempt to extract more useful information [6–11]. The second is the use of robust feature detection and matching algorithms that are applied in feature-based visual egomotion estimation pipelines [12–18]. The third is to compensate for dynamic lighting during the image acquisition process by actively adjusting camera parameters (e.g., exposure time and gain). The first two approaches can help to improve the performance of visual navigation algorithms when the acquired images already contain sufficient information, but will not recover information that is lost due to overexposure and underexposure [19]. Thus, ensuring that captured images contain sufficient information is paramount to the success of visual navigation algorithms. This thesis focuses on the third approach, that is, actively adjusting exposure time and gain to compensate for changes in environmental lighting.

Control of camera parameters to improve the quality of images has been an area of active study for several decades [20–22]. Almost all digital cameras available today contain a proprietary automatic parameter control algorithm that adjusts exposure time, gain, focus, and any additional camera parameters to ensure that the "best quality" images are acquired. These proprietary control algorithms are hand-crafted and adjust camera parameters based on heuristics suited for a specific purpose. Additionally, the control algorithms usually behave in a scene-independent manner and the details of their operation are often not published by the manufactures [19].

Proprietary control algorithms typically make parameter adjustments based on the brightness of the current scene [23] and are developed to ensure that images captured are aesthetically pleasing to consumers. However, the methods by which automatic algorithms operate do not guarantee that acquired images are suited for computer vision or robotics applications [24]. Although human assessment of image quality is subjective,

people are generally concerned with the appearance of the entire scene. Visual navigation algorithms, however, typically rely on smaller image subregions that have distinct and unique characteristics that aid in navigation. Moreover, it is important that these unique subregions have some type of consistency between frames (i.e., feature detection or photometric consistency). There is a need, therefore, to develop a camera parameter control scheme that is tailored specifically to improve the robustness and performance of a specific robotic vision task.

1.2 Improving Image Quality for VO

Recent work in the area of camera parameter control for robot vision has focused on the adjustment of camera parameters to maximize some measure of image quality. The approaches outlined in the literature typically involve 1) defining a unique image quality metric and 2) developing an optimization scheme that searches for a maximum of the corresponding metric in the camera parameter space. The parameter space is sampled, either by capturing real images or by generating synthetic images, to determine the contour surface upon which the optimization is performed. Although sampling-based approaches work well in static environments where there is little lighting change, they suffer in dynamic environments because the optimization surface changes as the lighting changes. Thus, prior queries of the objective surface are no longer reliable. Since real-world autonomous robots commonly operate in dynamic environments, sampling-based approaches for controlling camera parameters are insufficient. There is a need for a camera parameter control algorithm that can operate effectively in dynamic environments without the limitations introduced by sampling. Robust operation in these environments necessitates a camera parameter control scheme that behaves in a predictive (rather than reactive) manner so as to avoid sampling of the parameter space during operation.

Traditional feedback control techniques are a potential approach that could be employed to solve the camera parameter control problem. Traditional controllers, however, rely on system models that accurately capture the relation between the system inputs and outputs. The high complexity of the photometric image formation process (Section 2.1.2) is difficult to accurately model. Hand-engineered approximations of this process would rely on assumptions that are not always valid, reducing the effectiveness of such controllers in dynamic environments.

The complexity of the image formation process, the desire to predictively adjust camera parameters, and the large amount of data acquired by autonomous robots motivates the use of a data-driven approach for controlling camera parameters. Specifically, we

investigate the application of deep convolutional neural networks (CNNs) to solve the problem of predictive camera parameter adjustment. CNNs extract a hierarchy of useful features from images and can be trained to regress or classify targets from these features. We consider a CNN that takes as input a sequence of images and the associated camera parameter settings and outputs adjustments to the parameter values such that the next image captured is better suited for use in visual odometry. Our intent is to improve the performance of VO under challenging lighting conditions by producing images that contain a higher number of useful, matchable point features.

A natural choice for an optimization objective in the context of VO would be pose estimation error. However, identifying the correct camera parameters that result in the highest pose estimation accuracy is, in general, very difficult. The image acquisition process is not differentiable; we cannot employ backpropagation to update camera parameters from captured images. Further, generating training targets using pose estimation error would require a prohibitively large amount of ground truth data that is difficult to obtain. We also want our approach to work with existing VO pipelines that are known to perform well but may not be differentiable. Hence, we choose to leverage existing VO pipelines to generate training targets. Specifically, we select the number of image features (and inlier feature matches between sequential frames) obtained by the VO *front end* as the object of our optimization; we justify this selection by demonstrating empirically that these metrics[1] are good indicators of VO accuracy. The VO front end refers to the feature detection and matching component of the VO pipeline. We use training targets consisting of gain and exposure values that are appropriate for producing a high quality image during the next image acquisition. The network, producing gain and exposure predictions, is trained to regress these target values. The use of the VO front end to generate a training signal admits a self-supervised training approach since the number of image features or inlier feature matches is determined on the fly by the VO pipeline itself.

1.3 Contributions

We develop and implement a CNN that adjusts camera exposure time and gain parameters to improve the performance of visual navigation algorithms such as VO under difficult and dynamic lighting conditions. In this thesis, we make the following contributions:

- We undertake a comprehensive analysis of existing image quality metrics as they

[1] We note the use of the word 'metric' in this sense refers to a loose measure of image quality rather than in the strict mathematical sense of a distance measure.

relate to the task of visual egomotion estimation. We propose and analyze task-specific image quality metrics for visual egomotion estimation and justify the selection of our final metrics.

- We develop two variants of a predictive CNN that adjusts camera parameters to obtain images with an increased amount of inlier feature matches between sequential frames. The first network adjusts the camera exposure time while the second network adjusts both exposure time and gain.

- To effectively train and test our networks, we collect unique datasets consisting of a variety of camera trajectories captured with varying camera parameter settings.

- We compare the performance of our predictive networks to a built-in automatic parameter control algorithm. Additionally, we reimplement an existing camera parameter controller from the literature and compare its performance to that of our method. We show that our network yields images containing a higher number of inlier feature matches over various trajectories under static lighting conditions and outperforms these controllers in dynamic lighting conditions. Finally, we demonstrate that our approach results in a higher number of useful images for a monocular VO application over a large real-world driving route along which lighting conditions vary.

Chapter 2

Background

This chapter provides background material related to the major technical topics discussed throughout the thesis. First, key concepts related to the camera image formation process are described. Next, we present a primer on visual egomotion estimation (or visual odometry), followed by an overview of the VO front end. We conclude with a brief review of deep learning.

2.1 Cameras and Camera Parameters

We first discuss monocular cameras and the perspective camera model. Next, we cover the geometric process by which images are formed. Finally, important camera parameters related to the image formation process are introduced.

2.1.1 The Perspective Camera and Geometric Image Formation

The monocular camera can be modelled as a pinhole (or an ideal perspective camera) that maps a point in 3D world coordinates, $\boldsymbol{\rho} = [x, y, z]^T$, to a normalized 2D point, $\mathbf{p} = [x_n, y_n]^T$, on the image plane. The normalized 2D point is then mapped to *pixel coordinates*, (u, v) in the digital image. Figure 2.1 shows the frontal projection model of a monocular camera, where the image plane is placed in front of the lens (pinhole) for simplicity. In reality, the image plane is actually located behind the lens and the image is inverted.

The z-axis (denoted by the dashed line) of the camera frame $\underrightarrow{\mathcal{F}}_s$ is the *optical axis*, and is orthogonal to the image plane. The optical axis passes through the image plane at the *principle point*, expressed in units of horizontal pixels c_u and vertical pixels c_v; the distance between the principle point and the optical centre of the camera is the camera

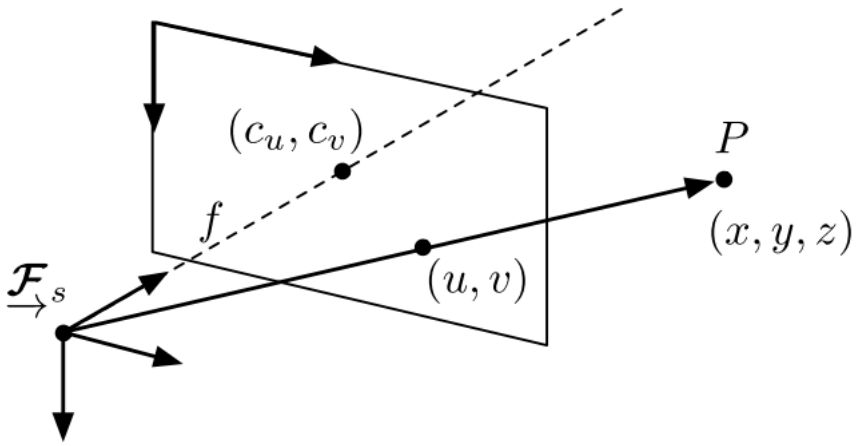


Figure 2.1: The frontal projection model of an idealized perspective monocular camera, from [25]

.

focal length $f = (f_u, f_v)$, also expressed in units of horizontal pixels f_u and vertical pixels f_v.

To transform a 3D world point, $\boldsymbol{\rho}$, into 2D *normalized image coordinates*, $\mathbf{p}$, we assume an idealized camera model where $f = 1$. We express $\boldsymbol{\rho}$ in the camera frame as:

$$\mathbf{p} = \frac{1}{z}\boldsymbol{\rho} = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{z}{z} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}, \tag{2.1}$$

where $x_n = \frac{x}{z}$ and $y_n = \frac{y}{z}$ are the normalized image coordinates of point $\boldsymbol{\rho}$. From the normalized image coordinates, we map to the pixel location $\mathbf{y} = [u, v, 1]^T$ through the intrinsic parameter matrix $\mathbf{K}$:

$$\mathbf{y} = \mathbf{K}\mathbf{p}, \tag{2.2}$$

$$\mathbf{y} = \underbrace{\begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix}. \tag{2.3}$$

The intrinsic camera matrix $\mathbf{K}$ is composed of the focal lengths and the principle point.

We can combine both steps to define the full perspective camera model,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{PK}\frac{1}{z}\boldsymbol{\rho}, \tag{2.4}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ \frac{z}{z} \end{bmatrix}, \tag{2.5}$$

where $\mathbf{P}$ is simply a projection matrix used to remove the z-component of the projected point. Note that we have not included the compensation for lens distortion above, which is usually carried out as a preprocessing step.

2.1.2 Camera Parameters and the Image Formation Process

The geometric mapping from 3D world point to 2D image point is only one aspect of the image formation process. Images are a collection of pixels, each with an assigned intensity, that together form a representation of a scene. In this section, we describe the image formation process, which determines how these pixel intensities are assigned, and the camera parameters that control how these values can change. Three important camera parameters that determine the overall brightness (and quality) of captured images are the lens aperture, the shutter speed (or exposure time), and the gain.

The lens aperture refers to the size of the lens opening that allows light to fall on the image sensor, contributing to both image brightness and focus [26]. The aperture is typically measured in *f-stops*[1] which define the ratio between the lens' focal length and the effective aperture diameter. The larger the effective camera aperture (or smaller the f-stop) for a given exposure time, the brighter the image. The aperture also affects the range of focus, or *depth of field*, which generally determines at what range objects in an image can be in focus. A large aperture results in a shallow depth of field, typically allowing for objects closer to the camera to be in focus and objects further away to be out of focus. A small aperture yields a larger depth of field, resulting in objects both near and far from the camera being in focus. However, the depth of field only controls the depth of focus, while a separate function typically allows the user to control the position of the focus. Examples of various lens apertures are shown in Figure 2.2.

For most robot vision tasks, we are concerned with both the objects in the immediate

[1]It is customary to represent the f-stop or (f-number) as the ratio preceded by 'f/'. Note that this 'f' does not represent the focal length and should not be considered as a part of the ratio.

Figure 2.2: Examples of various camera apertures with associated f-stop (or f-numbers). The f-stop represents the ratio between the lens' focal length and the effective aperture. (*Credit: commons.wikimedia.org*).

vicinity of the robot as well as the structure of the environment surrounding the robot. For this reason, the lens aperture is typically set to a relatively small value, ensuring that image details both near and far are in focus. For our work, we manually fix the lens aperture to a small value.

The second important camera parameter is the shutter speed (exposure time), which is the length of time that the camera shutter is open during image capture, exposing the camera sensor to light [26]. Since most machine vision cameras now have electronic shutters, the length of time the shutter is "open" refers to the length of time the sensor accumulates photons. Exposure time is typically measured in microseconds (μs) and contributes to both the brightness and the amount of motion blur in an image. Intuitively, the overall image brightness increases with exposure time, due to more photons striking the sensor (see Figure 2.3). Increased exposure time has the added effect of increasing image blur in scenes containing motion. Blur is a result of 3D scene points moving while the shutter is open, which causes a 3D point to be mapped to multiple pixels. The effect

(a) Exposure time = 190 μs (b) Exposure time = 680 μs (c) Exposure time = 5185 μs

Figure 2.3: Examples of a scene captured with varying exposure values while the gain was held constant at 10 dB.

Figure 2.4: Motion blur caused by high exposure. The left image corresponds to a shorter exposure time, resulting in a much sharper image, while the right image corresponds to a higher exposure time, resulting in a brighter, more blurry image. Note the increase in motion blur, particularly in the trees.

is magnified with longer exposure times. Shorter exposure times result in darker but sharper images. The effects of image blur due to camera motion are shown in Figure 2.4.

The final parameter, camera gain, is the amplification of the signal produced by the image sensor and is measured in decibels (dB). The amplification is applied after an image has been captured [26]. Gain increases the image brightness, but also increases the amount of noise in the image. An example of this effect is presented in Figure 2.5, where the same scene is captured at three different gain settings while the lens aperture and exposure time are held fixed. The brightest image, corresponding to the highest gain, contains a significant amount of noise.

The mapping from the quantity of light striking a camera sensor to the pixel intensity values is determined by the *camera response function* (CRF) (or photometric response

(a) Gain = 15 dB (b) Gain = 28 dB (c) Gain = 35 dB

Figure 2.5: Examples of a scene captured with varying gain values while the exposure time is held constant at 2000 μs.

function) [26–29]. The CRF is a nonlinear mapping that is unique to each camera and image sensor. It defines the relationship between the quantity of photons measured by the sensor q, in energy per unit time, and the exposure time E, and the pixel intensity, I typically expressed between 0–255. For pixel i, and exposure time j:

$$I_{i,j} = f(q_i E_j). \tag{2.6}$$

The CRF is dependent on both the lens aperture and exposure time, but not on the gain, which is applied later in the image processing pipeline. An example of a CRF is shown in Figure 2.6. Each of the lines in the figure represents the response of a particular RGB colour channel.

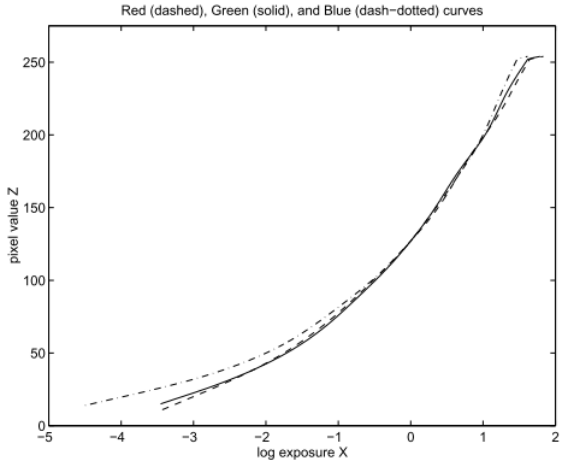


Figure 2.6: An example of a typical camera response function, obtained from [27] for each RGB channel.

2.2 Visual Odometry

Estimating the trajectory of a robot as it moves through an environment is an important challenge. To effectively estimate the trajectory, the robot must be able determine its pose (position and orientation) over time. This is a fundamental requirement that enables a mobile robot to perform more advanced tasks. For example, the simultaneous localization and mapping (SLAM) task involves constructing an internal map or representation of the environment and localizing within this map. In a motion planning task, a robot determines a safe route through the environment based on its current map.

Sensors used for perception are typically organized into categories based on the properties they measure and how measurements are acquired. Sensors can first be categorized

based on the type of measurements they record, as interoceptive or exteroceptive. Interoceptive sensors measure quantities internal to the robot itself, such as the rate of rotation of its wheels. Exteroceptive sensors measure a quantity external to the robot, such as the distance to an object in the environment. Sensors can also be categorized as active or passive based on the process by which they acquire measurements. Active sensors emit a signal, typically in the form of radiation such as light or radio waves, and measure its interaction with the environment. For example, the distance to an object can be determined based on the time-of-flight (ToF) of a light or radio signal. Passive sensors, alternatively, measure existing signals within an environment such as the natural lighting, without contributing any additional signal into the environment.

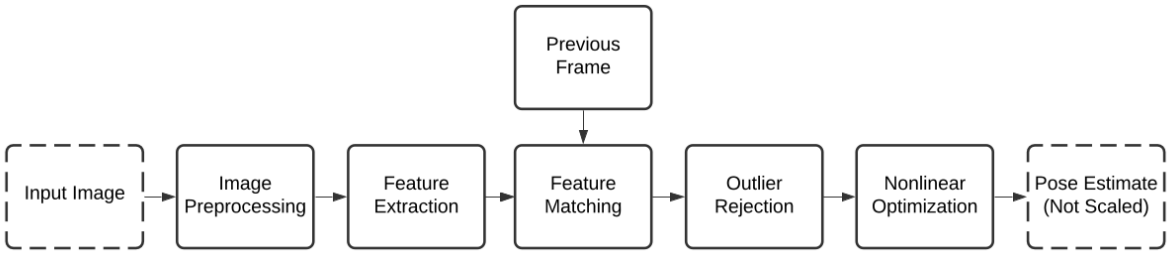
Cameras are exteroceptive, passive sensors that have become very popular for mobile robotics applications due to several key attributes. First, cameras are relatively simple devices that do not contain complex moving parts, unlike most 3D LIDAR sensors. Second, multiple cameras can operate within the same area without interference, again unlike many LIDAR sensors. Third, cameras are relatively inexpensive compared to more niche exteroceptive robotics sensors. Cameras also provide a rich, dense stream of data that can be used for multiple purposes simultaneously, such as object detection and avoidance, place recognition, navigation, tracking, and terrain classification. There are many different types of cameras, including perspective, omnidirectional, and spherical, that can be used individually or in combination. In fact, it is common for multiple cameras (along with other sensors) to be used on a single mobile robot platform to increase the performance of various navigation algorithms.

Prior to the rise of visual sensing in robotics, egomotion estimation was typically accomplished using only wheel odometry, which is the process of inferring motion relative to an initial position by integrating wheel rotation rates over time. Wheel odometry accuracy, in practice, is greatly reduced due to wheel slippage, which can occur on a variety terrains including sand, mud, and rocks. The inaccuracy of wheel odometry resulted in the need for a more robust and accurate motion estimation pipeline, one that was agnostic of the terrain traversed by the robot. Estimation of robot motion using cameras alleviated many of the issues related to wheel slippage. After Moravec's [1] groundbreaking thesis in 1980, the majority of progress in this field for the next two decades was driven by Larry Matthies and his team at NASA's Jet Propulsion Laboratory. Their efforts were crucial in the success of the Mars Exploration Rover program [30] where wheel slippage on sandy Martian terrain was a common occurrence. The term "visual odometry," referring to the process of using images to estimate the egomotion of a rigid body over time, was formally proposed by Nister in 2004 [31].

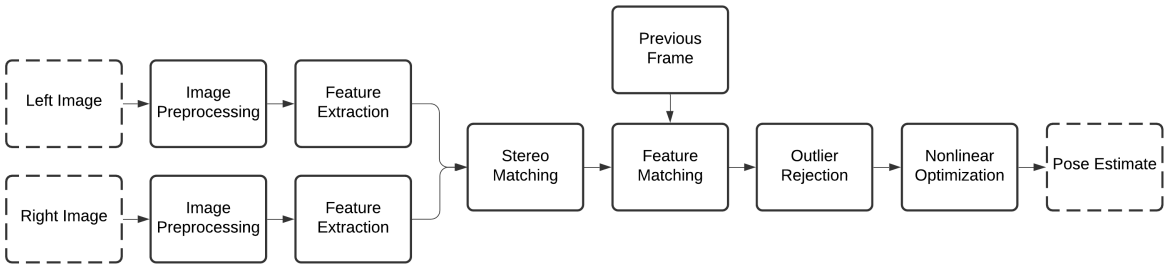
2.2.1 The VO Pipeline

Visual odometry operates as a pipeline that takes as input images captured by one or more cameras and produces a six degrees-of-freedom (6-DOF) pose change estimate between sequential frames. There are several distinct types of VO that share this pipeline structure but differ in the number and type of cameras used and in the general methodology of the approach. The three general classes of VO are direct (or appearance-based) methods, feature-based methods, and hybrid methods that are a combination of the first two. Direct VO works by matching the intensity information of sequential images (or subregions of images) to estimate the pose change of the camera between frames. Direct methods, which are primarily used in monocular VO applications, are generally more computationally expensive than feature-based VO, more difficult to implement in the stereo VO case, and less robust to occlusions [32]. Feature-based VO operates by extracting and matching (or tracking) salient point features across sequential images. Since only a small subset of the image information is used, feature-based methods are typically faster than direct methods. This thesis is focused on improving the performance of feature-based VO specifically, however the general approach proposed could also be modified to operate with direct methods. Although feature-based VO often relies on features that are designed to be relatively invariant to intensity changes, dynamic lighting can still result in spurious matches, or even failure of the VO system. Further, the outputs generated by the front end of common feature-based VO pipelines allow for a convenient gauge of image quality. For brevity, we will refer to feature-based VO simply as *VO* throughout the remainder of this thesis.

Block diagrams of typical monocular and stereo VO pipelines are shown in Figure 2.7. Each block in the diagram performs a key function. The image preprocessing step involves image dewarping (and image rectification in the stereo case) using the camera intrinsics and distortion parameters, discussed in more detail in Section 2.1. Feature extraction is the process by which key regions or pixels of interest that have some defining characteristic are identified in the input image. Once image features have been extracted from the current image, they are matched to image features from a previous image using common feature matching algorithms. The specifics of feature extraction and matching are covered in more detail in Section 2.2.2. In the case of stereo VO, the 2D features in the left and right images are matched first, before being matched to the previous frame. By matching 2D features between left and right cameras with a known baseline, the 3D position of feature points in the environment relative to the camera can be inferred using simple trigonometry. Thus, stereo VO outputs absolute pose estimates, while in the monocular VO case, only 2D feature locations are available and



(a) Monocular VO



(b) Stereo VO

Figure 2.7: Processing blocks of (a) a feature-based monocular VO system, (b) a feature-based stereo VO system.

translation estimates can be recovered only up to an unknown scale factor. Proposed sets of feature correspondences are then processed by an outlier rejection module to eliminate spurious matches. A common outlier rejection algorithm is random sample consensus (RANSAC) [33]. Finally, the pose change is computed using nonlinear optimization. The output of the pipeline is the 6-DOF pose change between image acquisitions. Pose changes can be concatenated together to form a local trajectory over time. The estimated trajectory can be further refined through windowed bundle adjustment (BA) discussed later in this chapter.

Formally, using the notation of Barfoot [25], we define the trajectory of a camera through an environment, and by extension, the robot the camera is rigidly fixed to, as a set of poses, $\{\mathbf{T}_0, ..., \mathbf{T}_K\}$. The poses describe the 6-DOF position and orientation of a rigid body located at the origin of a vehicle frame of reference, $\underrightarrow{\mathcal{F}}_v$, relative to an inertial (or world) frame of reference, $\underrightarrow{\mathcal{F}}_i$, over time. Poses can be represented as elements of the *special Euclidean* matrix Lie group SE(3) in the form of equation 2.7 as 4×4 transformation matrices. We define a set of poses at discrete time instants $k, k \in \{0, ..., K\}$ that are relative to the initial pose at $\mathbf{T}_0$. The pose can be represented

as a 4×4 transformation matrix of the form:

$$SE(3) = \left\{ \mathbf{T}_k = \begin{bmatrix} \mathbf{C}_k & \mathbf{r}_k \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \,\middle|\, \mathbf{C} \in SO(3), \mathbf{r} \in \mathbb{R}^3 \right\}. \tag{2.7}$$

The transformation matrix is constructed from a rotation matrix, $\mathbf{C}_k$, and a translation vector, $\mathbf{r}_k$. The *special orthogonal* group $SO(3)$ represents rotations as the set of valid rotation matrices, $\mathbf{C}_k = \mathbf{C}_{iv,k} \in \mathbb{R}^{3\times3}$, from the vehicle frame[2] $\underrightarrow{\mathcal{F}}_v$ to our inertial frame $\underrightarrow{\mathcal{F}}_i$ at time k:

$$SO(3) = \left\{ \mathbf{C}_k \in \mathbb{R}^{3\times3} \,|\, \mathbf{C}\mathbf{C}^T = \mathbf{1}, \det(\mathbf{C}) = 1 \right\}. \tag{2.8}$$

The rotation matrix has nine parameters but only three degrees of freedom due to orthogonality constraints. The translation at time k is represented in the inertial frame as $\mathbf{r}_k = \mathbf{r}_{i,k}^{v,i} \in \mathbb{R}^3$. The translation vector represents a three dimensional position vector originating at the origin of the inertial frame $\underrightarrow{\mathcal{F}}_i$ and terminating at a point v, measured relative to the inertial frame. The transformation matrix has six free parameters. Two camera poses $\mathbf{T}_{k-1}$ and $\mathbf{T}_k$ are related by the 6-DOF pose change $\mathbf{T}_{k,k-1}$:

$$\mathbf{T}_{k,k-1} = \begin{bmatrix} \mathbf{C}_{k,k-1} & \mathbf{r}_{k,k-1} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.9}$$

The output of the VO pipeline is an estimated pose change, $\hat{\mathbf{T}}_{k,k-1}$. Using this pose change, we can calculate the current pose of the camera by multiplying the pose from the previous time step $\hat{\mathbf{T}}_{k-1}$ with the estimated pose change, $\hat{\mathbf{T}}_{k,k-1}$. The camera pose at time k relative to the inertial frame at $\mathbf{T}_0$, is then:

$$\hat{\mathbf{T}}_k = \hat{\mathbf{T}}_{k,k-1}\hat{\mathbf{T}}_{k-1}. \tag{2.10}$$

This thesis focuses on the application of monocular VO. There are two general approaches to estimating the transformation between two camera poses $\hat{\mathbf{T}}_{k,k-1}$ in monocular VO. The first is using 2D-to-2D feature correspondences [32]. Here, we first find the 2D features in images I_k and I_{k-1} and determine the 2D inlier feature correspondences in normalized image coordinates, $\tilde{\mathbf{p}} = [\tilde{u}, \tilde{v}, 1]^T$ and $\tilde{\mathbf{p}}' = [\tilde{u}', \tilde{v}', 1]^T$, respectively. From these feature correspondences, the *essential matrix* $\mathbf{E}$ between the two camera poses can be computed up to an unknown translational scale factor using the *epipolar constraint*:

$$\tilde{\mathbf{p}}'\mathbf{E}\tilde{\mathbf{p}} = 0. \tag{2.11}$$

[2]For clarity, we use the notation from Barfoot [25]. In this case, our vehicle frame is interchangeable with the camera frame.

The epipolar constraint defines the relationship between the projections of a 3D scene point into two images acquired from different poses. The projection of the 3D scene point in one image must lie along an *epipolar line* in the second image. The epipolar line is defined by the 3D position of the point and the optical centre of the first camera projected onto the image plane of the second camera.

The essential matrix relates corresponding image points using the epipolar constraint, and can be estimated using the five-point algorithm [31] or the eight-point algorithm [34]. From $\mathbf{E}$, we can extract four possible combinations of rotation and translation estimates. The correct rotation, $\mathbf{C}_{k,k-1}$, and translation, $\mathbf{r}_{k,k-1}$, can be determined by triangulation of a point and selection of the rotation and translation pair that result in the point appearing in front of both cameras. The rotation and translation estimates can then be used as the initial guess in a nonlinear optimization of the reprojection error of the point correspondences. The relative scale of the translation estimate can be computed by triangulating point correspondences in three sequential frames.

An alternative method for recovering the relative pose estimates is to use 3D-to-2D correspondences as outlined in [32]. Here, at least three point correspondences are required, unlike the 2D-to-2D method which requires a minimum of five, resulting in faster data association. In the 3D-to-2D case, 3D points, $\boldsymbol{\rho}_{k-1}$, are calculated from three previous sequential frames, while the 2D features, $\mathbf{p}_k$, are obtained from the current frame. Next, the 3D points are transformed into the current frame using the transformation $\hat{\mathbf{T}}_{k,k-1}$ as $\hat{\boldsymbol{\rho}}_k$ and projected into the current image as $\hat{\mathbf{p}}_k$. The reprojection error between $\hat{\mathbf{p}}_k$ and $\mathbf{p}_k$ is then minimized by optimizing over the parameters of $\hat{\mathbf{T}}_{k,k-1}$:

$$\hat{\mathbf{T}}_{k,k-1} = \underset{\hat{\mathbf{T}}_{k,k-1}}{\arg\min} \sum_i \|\mathbf{p}_k^i - \hat{\mathbf{p}}_k^i\|^2, \quad \hat{\mathbf{p}}_k = \frac{1}{z}\hat{\mathbf{T}}_{k,k-1}\boldsymbol{\rho}_{k-1}. \tag{2.12}$$

Fischler [33] outlines one method to solve this problem but there are a variety of other approaches that can be used [32].

Regardless of the method used to estimate the inter-frame transformation $\hat{\mathbf{T}}_{k,k-1}$, the end result is a sequence of pose-to-pose transformations that begins at the initial pose of the camera, $\{\hat{\mathbf{T}}_{1,0}, \ldots, \hat{\mathbf{T}}_{k,k-1}, \ldots, \hat{\mathbf{T}}_{K,K-1}\}$. Using Equation 2.10, we can process the sequence of pose-to-pose transformations to obtain a trajectory of poses, $\{\mathbf{T}_0, \ldots, \hat{\mathbf{T}}_k, \ldots, \hat{\mathbf{T}}_K\}$.

Pose estimates can be iteratively improved through the use of the windowed bundle adjustment (BA) algorithm [35]. BA is one approach to reducing the drift of VO by utilizing feature measurements over more than two frames. It operates by performing a nonlinear optimization of both the camera poses and 3D feature coordinates over the last

n frames. It requires the same features to be tracked over multiple frames. Scaramuzza and Fraundorfer [2] formulate this optimization the following way:

$$\arg\max_{\hat{\mathbf{T}}_k, \boldsymbol{\rho}^i} = \sum_{i,k} \|\mathbf{p}_k^i - g(\boldsymbol{\rho}^i, \hat{\mathbf{T}}_k)\|^2. \tag{2.13}$$

Here, $\mathbf{p}_k^i$ is the projection of the 3D point $\boldsymbol{\rho}^i$ in image k, and $g(\boldsymbol{\rho}^i, \hat{\mathbf{T}}_k)$ is the reprojection of that same 3D image point using the pose estimate $\hat{\mathbf{T}}_k$. The squared reprojection error is then minimized using a nonlinear optimization algorithm such as the Levenberg-Marquardt algorithm.

At each time step, the pose estimate will not be perfectly accurate. As the VO algorithm runs, pose estimation error is propagated forward in time. The compounding of errors causes VO to drift; drift in the estimated rotation is the most detrimental since this affects the position and translation accuracy. For this reason, it is vital that pose estimates are as accurate as possible. Consequently, the image correspondences used to generate the pose estimates should be of high quality and quantity [2].

2.2.2 Image Features and Matching

A key component of the VO pipeline is the front end, which carries out the feature detection and matching processes. The problem of identifying and matching unique regions of interest in images has been extensively studied; we provide a brief introduction to common feature detectors, descriptors, and matching algorithms below.

Feature detectors identify pixels or groups of pixels in an image that differ from their surrounding pixels by a significant amount. Identifying whether a given pixel is considered an image feature is usually determined by comparing the pixel intensities in a small region. Govender [36] and Fraundorfer [2] provide comprehensive reviews of commonly used feature detectors for computer vision and robotics applications. Included in these reviews are the Harris corner detector [1], the Kanade-Lucas-Tomasi (KLT) detector [37,38], the scale-invariant feature transform (SIFT) [12], speeded-up robust features (SURF) [13], and the features from accelerated segment test (FAST) detector [14]. The primary requirements of feature detectors are that they produce unique and distinguishable features that can be identified and matched across multiple images; are robust to image degradation due to noise, blur, and image compression; have a high repeatability, meaning that the same features are identified in successive images of the same scene; are computationally inexpensive to permit real-time operation at a high frame rate; and that the features produced are relatively invariant to changes in perspective, image distortion,

and illumination.

Once a feature has been detected by one of the above methods, it is assigned a unique identifier so that it can be quantitatively compared to other features and matched between images. The unique feature identifier is called the *descriptor* of the feature and is generated by a feature descriptor algorithm. Descriptors are typically generated by analyzing specific properties of the region around the feature point. Most feature descriptor algorithms analyze image patches on varying scales or with image operators (e.g., gradient, Laplacian, and Gaussian edge and line detectors) applied in different directions. It is common for feature detection and description to be combined into the same algorithm. Common vector-based feature descriptors are those found in SIFT [12] and SURF [13]. Other descriptors include the binary robust invariant scalable keypoint descriptor (BRISK) [15], the binary robust independent elementary feature (BRIEF) descriptor [16], and the oriented FAST and rotated BRIEF (ORB) descriptor [17]. The latter three descriptors are binary (bit string) descriptors and are much faster to compute than SIFT and SURF. Consequently, they have been used more extensively in real-time robotics applications for VO and SLAM [2]. Alternatively, the entire feature detection and description process can be learned, as in [18]. Our method is agnostic to the feature detection and description algorithm used.

The process of feature matching, that is, identifying the same feature across multiple images, is accomplished by comparing descriptors using a similarity metric. Descriptors that fall within a specific distance threshold are considered matches while those that do not are discarded. There exist a number of methods used to match features between two images. Brute-force matching consists of comparing every descriptor in one image to every descriptor from a second image and selecting the pairs that are closest to each other. The similarity between two vector descriptors (e.g., SIFT or SURF) can be calculated by measuring the Euclidean distance between the vectors, while the similarity between binary string descriptors can be calculated using the Hamming distance. A mutual consistency check is typically performed on a potential match [2]. The mutual consistency check consists of performing the feature matching in the opposite direction (i.e., comparing a feature in the second image to every feature in the first image) and verifying that the proposed match from the first image to the second is the same as the proposed match from the second to the first. The mutual consistency check helps in cases where there are multiple potential matches for a specific feature. Alternatively, rather than searching an entire image for potential matches, the search window can be constrained through the use of a known or fixed motion model [2], which improves the speed of the feature matching process.

After a proposed set of feature correspondences has been obtained, an outlier removal scheme is usually employed to eliminate spurious matches since, even with the use of a mutual consistency check, some feature correspondences may not be correct. A common approach to remove spurious matches is to consider all feature correspondences as a group when determining which matches are legitimate (inlier matches) and which matches are wrong (outlier matches). The most widely used algorithm for outlier rejection is the random sample consensus (RANSAC) [33]. RANSAC determines whether a set of matches are inliers or outliers by testing how well they fit an underlying model. In the case of VO, the model is a 6-DOF pose change from one image frame to another in the form of a transformation matrix, $\mathbf{T}_{k,k-1}$. First, a small subset of point correspondences from the set of all proposed correspondences is randomly selected. Using these matches, an estimate of the transformation $\hat{\mathbf{T}}_{k,k-1}$ is determined through minimization of the point-to-epipolar line distance for each of the selected matches. Next, this transformation is applied to all of the remaining features in one image. For each of these features, the point-to-epipolar line error between the epipolar line of the transformed feature and its proposed correspondence is calculated. Features are classified as inliers if their point-to-epipolar line error falls below a user-defined threshold. The number of inliers for the proposed $\hat{\mathbf{T}}_{k,k-1}$ is recorded and the process is repeated for a large number of trials, each with a new subset of randomly sampled feature correspondences. Upon completion, the transformation yielding the highest number of inlier matches is selected, and all matches deemed as outliers are discarded. The number of trials, k, needed to find the solution with a specific probability of being outlier-free can be calculated with the following formula:

$$k = \frac{\ln(1-p)}{\ln(1-w^N)}, \tag{2.14}$$

where p represents a user defined success rate, or the probability that one of the random proposals is successful after RANSAC finishes. The value N is the number of data points (correspondences) used to estimate the transformation during each iteration, and w represents the probability that a datapoint is an inlier.

2.2.3 Challenges in VO

Accurate VO relies on finding correct feature correspondences between sequential images over the entire camera trajectory. VO performance degradation is most apparent in environments where conditions for visual navigation are challenging. As discussed in Chapter 1, there are three general approaches for improving the robustness of the VO

front end [4]. The first two approaches (i.e., post-processing and robust feature detection and matching) have been extensively studied, resulting in many key contributions to the field of visual navigation. However, the two approaches can only serve to improve (or operate on) images that already contain a sufficient amount of information. It is exceedingly important that captured images are of relatively high quality, which necessitates the third approach for improving the robustness of the VO front end. The third approach is the adjustment of camera parameters during the process of image acquisition. Improving the performance of the image acquisition process is often overlooked, mainly due to the assumption that built-in automatic parameter controllers are sufficient, or that there are minimal lighting changes within an environment [39–41]. In practice, these assumptions do not always hold, especially in outdoor scenes where lighting conditions cannot be controlled and the speed of camera motion may be high. Images that are too bright or dark (i.e., overexposed or underexposed), or that contain significant motion blur or noise, can cause visual navigation algorithms such as VO to fail outright [3, 39, 40], no matter how sophisticated the feature detection and matching algorithms are. Through appropriate adjustments of the camera exposure time and gain, higher-quality images can be captured and used in conjunction with the first and second approaches to improve the robustness of the VO front end in difficult conditions.

2.3 Deep Learning

Machine learning has seen a resurgence in popularity due to both increased computational power at lower costs and the development and support of widely used machine learning libraries. Autonomous mobile robotics is a natural application area for machine learning algorithms. Mobile robot sensors capture large amounts of data that are ideal for the training of data-driven models.

The predictive control of exposure time and gain, described in this work, is achieved through the use of neural networks and *deep learning* [42]. A comprehensive overview of deep learning is provided in [43]. Deep learning is a branch of machine learning that is built on the assumption that certain kinds of data can be modelled using a hierarchy of representations. Low level features extracted from an input are used to construct more complex, higher-level features. The hierarchical structure can be displayed as a graph multiple layers deep, hence the moniker *deep* learning.

2.3.1 Deep Feedforward Networks

Deep feedforward networks are the most common deep learning models. These models seek to approximate some function f^* by defining a parametrized mapping $\mathbf{y} = f(\mathbf{x}; \theta)$. The goal of the learning process is to find the parameter weights, θ^*, that best approximate the desired function, f^*. A network is designated as *deep* because it is a composition of many functions, called *layers*, in a directed acyclic graph structure. The layers can each be represented as a function, $f^{(n)}$, and chained together to create a mapping from the input to the output:

$$\mathbf{y} = f^{(N)}(f^{N-1}(...f^{(2)}(f^1(\mathbf{x}))...)). \tag{2.15}$$

The length of this chain of functions gives the *depth* of the network. The layers are often vector-valued and the parameters of the network, θ, are the combination of the parameters for each vector-valued function. The size of the largest vector-valued function is referred to as the network's *width*. The models are *feedforward* because they evaluate an input $\mathbf{x}$ by passing it through a sequence of layers until it reaches the output, without any feedback into previous layers.

Supervised learning is the process by which a network learns optimal parameters from labelled data such that it approximates a function f^*. *Self-supervised* learning refers to a subset of supervised learning techniques in which the training targets used for learning are generated automatically. Our training methodology, discussed in Section 4.2.5, can be considered self-supervised, as we make use of the VO front end to generate training targets from image data. We note that there exist other methods used to train these models, (e.g., semi-supervised learning, unsupervised learning), but we only make use of supervised/self-supervised learning in this work. Supervised training requires a set of inputs, $\mathbf{x}$ and associated output targets $\mathbf{y}^*$ generated from f^*. This set of inputs and output targets is referred to as a *dataset*, $\mathcal{D}$. The inputs are passed through the network and the output $\mathbf{y}$ is compared to the target $\mathbf{y}^*$. The difference between the network output and target is calculated using a *loss function*, $\mathcal{L}$, and is used to adjust the values of θ such that the difference between the output $\mathbf{y}$ and the target $\mathbf{y}^*$ is minimized,

$$\theta^* = \arg\min_{\theta} \mathcal{L}(f(\mathbf{x}; \theta), \mathbf{y}^*). \tag{2.16}$$

Though the desired output of the network is fixed, the intermediate representation of the input as it passes through the network is not predefined and thus, not interpretable. For this reason, all layers in the network apart from the input and output are referred to as

hidden layers.

Each layer in a deep network takes the input $\mathbf{z}_n$ from the previous layer, applies an affine transformation, then passes the result through a nonlinear *activation function*, $\sigma(\cdot)$. These outputs then feed into the next layer of hidden units, $\mathbf{z}_{n+1}$ (or the output of the network $\mathbf{y}$ in the case of the final network layer):

$$\mathbf{z}_{n+1} = \sigma(\mathbf{W}_n \mathbf{z}_n + \mathbf{b}_n), \tag{2.17}$$

where $\mathbf{W}_n$ is the weight matrix for layer n, and b_n is the vector of bias terms for the layer. The weights and biases are the parameters that the network learns during training. The nonlinear activation function, $\sigma(\cdot)$ applies a nonlinear transformation to the input. The addition of a nonlinear activation function allows the network to learn complex nonlinear functions. A common activation function, and the one used in this thesis, is the rectified linear unit (ReLU). The ReLU is a very simple function that maps an input to either itself or zero:

$$\sigma(x) = \max(0, x). \tag{2.18}$$

The process of learning is achieved through modifications of the parameter weights. The difference between the network output and target is propagated backwards through the network, and the gradient of this difference is calculated for each weight using the *back-propagation* algorithm [42]. The gradient of the loss function with respect to each weight indicates the direction of maximal increase of the loss. Each weight is then adjusted in the direction opposite the gradient by a small amount, proportional to a scaling factor called the *learning rate*. The adjustment of every weight in the model results in a reduction of training error over each training iteration. The adjustment of weights proportional to the gradient of the loss function is called *gradient descent*. A common implementation of gradient descent is *stochastic* gradient descent (SGD). In SGD, a small random subset of training samples are passed through the network and their gradients are calculated and averaged. The average gradient is then used to update the weights. The advantage of SGD is that it avoids calculating the gradient of the entire dataset during each step. SGD iterations are repeated for all of the samples in the dataset multiple times until convergence. A modern modification to SGD is the ADAM optimizer [44], which is the technique used throughout this thesis.

2.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a unique family of neural networks specifically suited to operate on data that is structured in the form of multiple arrays or matrices, such as images. CNNs differ from traditional neural networks in that they perform a convolution operation over an input using a *kernel*, which is more tractable for high dimensional images. A convolution is a linear operation that maps unique *features* in a local area of an image to a more compact representation. Note that the features mapped in a CNN are not the same as the hand-crafted features discussed in Section 2.2.2. The continuous and discrete convolution operations are shown in Equations 2.19 and 2.20 respectively:

$$y(t) = (x * k)(t) = \int x(a)k(t-a)da, \tag{2.19}$$

$$y(t) = (x * k)(t) = \sum_{a=-\infty}^{\infty} x(a)k(t-a). \tag{2.20}$$

An input signal, x, is convolved with a *kernel*, k, outputting a *feature map*, y. In the case of a 2D $(M \times N)$ image $\mathbf{I}$, the convolution operation takes the following form:

$$\mathbf{Y}(i,j) = (\mathbf{K} * \mathbf{I})(i,j) = \sum_{m}\sum_{n} \mathbf{I}(i-m, j-n)\mathbf{K}(m,n). \tag{2.21}$$

The kernel, $\mathbf{K} \in \mathbb{R}^{m \times n}$ is typically a square 2D matrix that is applied to local regions of the same size over an entire image, outputting a 2D feature map. It is common to skip over pixels to reduce the number of convolution operations required. The number of pixels skipped between convolution operations is called the *stride* of the convolution. Optionally, *padding* can be added to the edges of the image to ensure the kernel can be applied to edge pixels.

CNNs consist of sequential groups of convolutional layers, with each group reducing the dimensionality of its input. A typical CNN layer group consists of a convolutional layer, a pooling layer that reduces the size of the feature map, and a nonlinearity (typically ReLU). The output is passed into the next group, consisting of a similar set of layers. The pooling layer combines neighbouring features within a region, reducing the size of the input feature map while retaining important information. Common pooling strategies include taking the maximum or average value of a local area of the feature map. The combination of multiple local features into one has the added benefit of introducing an invariance to small translations of features. The final layer of a CNN is usually a fully connected (FC) layer.

One of the main advantages of using CNNs for images is the smaller number of parameters to learn. The parameters of the kernel are *shared* across spatial locations of the input, reducing the number of parameters compared to traditional networks where there exists a separate parameter for each input-output connection. In traditional neural networks, a matrix multiplication consisting of each of the individual parameters is applied to the input at each layer. For images, consisting of thousands or millions of pixels, the number of parameters is prohibitively large. CNNs, employing convolutional operations with shared parameters, only learn the parameters of the convolution kernels, $\mathbf{K}$, and the weights of any fully connected layers. The reduction in the number of parameters means that CNNs have reduced memory requirements and increased statistical efficiency [43].

Chapter 3

Related Work

Modern digital cameras typically are equipped with automatic exposure and gain controllers that are designed to adapt to varying lighting conditions. Depending on the sophistication of the camera system, the user can often fine-tune specific settings related to these automatic controllers, or turn them off all together. Built-in algorithms, however, are suited for general purpose photography (or videography) and are not intentionally designed for use in computer vision applications. Additionally, built-in algorithms tend to favour stability over reaction speed, as changes to exposure time and gain are made gradually while transitions in scene lighting may occur rapidly, resulting in overexposed or underexposed images. For these reasons, it is apparent that a more targeted approach is required for robotics applications.

3.1 High-Dynamic Range Imaging

One approach to maximize the information content of an image is high-dynamic range imaging (HDR) imaging. HDR scenes are characterized by a very wide brightness/radiance range. It is very difficult to capture all of the important details in an HDR scene within a single image, as the dynamic range of the scene can be orders of magnitude larger than the dynamic range of the camera. In order to capture most of the important details, both dark and bright regions need to be appropriately exposed. HDR imaging is the process by which images acquired at various exposure times and gain values are fused together. HDR imaging has been shown to perform well in a variety of challenging environments, including outdoors. The intuition behind HDR imaging is that important features that appear in either bright or dark regions of a scene can be combined into one image. An HDR image is constructed by combining low-dynamic range (LDR) images of a scene for many different exposure and gain values. The LDR images are typically acquired from

the same perspective and are combined using an image fusion algorithm. The combination results in a well-exposed final image with a high amount of detail in both bright and dark scene regions. However, HDR imaging is not suited for real-time operation.

HDR images can be acquired in real time with the use of at least two cameras that have different exposure time settings and with a very small baseline (as in, e.g., [45]). The cameras simultaneously capture LDR images of the scene and, given the known baseline, the images can be fused using a saturation mask. The obvious downside to this approach is that two cameras are needed. In challenging environments, even more cameras may be required. Additionally, the fixed settings of a specific camera may result in useless LDR images in cases where lighting is extreme (e.g., the low-exposure camera will result in severely underexposed images when the external illumination is low).

Alternatively, LDR images of the same scene obtained with a single camera at different times can be fused into an HDR image [27]. This approach removes the requirement for multiple cameras but introduces several new challenges. First, the camera needs to capture multiple images from the same pose. Second, any changes in the environment between the LDR images will impact the quality of the HDR representation, in particular around object edges. Instead, the camera may move, but images must then be transformed and stitched together. In [46], HDR videos are generated from sequential images using LDR frames acquired with auto-exposure control and varying gain values. However, increasing the gain introduces noise into the images and in turn, into the resulting video.

Although HDR imaging can aid in extracting useful image features in HDR scenes, it is not suited for use in a visual egomotion estimation application. The motion of the camera, coupled with the changes in the environmental lighting limit the effectiveness of HDR imaging. The requirement that images must be acquired from the same position limits the allowable speed of the robot, while fusing images from multiple positions is potentially both slow and imprecise. Due to these limitations, the use of HDR imaging for real-time robotics applications is limited.

3.2 Reactive Control of Camera Parameters

Recent works have sought to increase the performance of visual navigation algorithms by adjusting camera parameters to maximize a specific image ‘quality’ metric. These approaches adjust parameters in a reactive fashion: an image is captured, the appropriate metric is calculated, and adjustments to the camera parameters are made based on the quality value or score.

Lu et al. [4] use the Shannon entropy of an image as the quality metric. The Shannon entropy of an RGB image is calculated as:

$$M_{ent} = -\sum_{i=0}^{255} P_{Ri} \log P_{Ri} - \sum_{i=0}^{255} P_{Bi} \log P_{Bi} - \sum_{i=0}^{255} P_{Gi} \log P_{Gi}, \tag{3.1}$$

where P_{Ri}, P_{Bi}, and P_{Gi} represent the probability of a pixel in each channel having an intensity value of $J \in \{0 - 255\}$ based on the image histogram. The Shannon entropy defines the information content of an image. Lu et al. [4] posit that increased image entropy will result in images with more useful information, resulting in a greater number of matchable features. The exposure time and gain of the camera are adjusted to maximize the entropy; the best parameter values are obtained by performing an optimization over a surface constructed by acquiring images at different exposure time and gain values and measuring their entropy. A 1-D optimization is carried out in the parameter space such that the exposure time and gain are adjusted by the same relative magnitude. The optimization is typically performed once, as it is assumed that the environmental lighting is relatively static. To determine if the lighting has significantly changed, a simple heuristic check is employed which involves measuring the mean brightness of a known reference object in the scene and determining if it has changed by a specific amount. If the mean brightness change is larger than a user-defined threshold, the optimization is performed again. The method works in environments that have stable lighting but it is not suited for dynamic conditions. In fact, the authors state that their approach breaks down in conditions where lighting changes suddenly. Additionally, the approach requires a known reference object.

Shim et al. [3] investigate the use of image gradients as a metric. The presence of a strong gradient indicates a substantial brightness difference between neighbouring pixels, corresponding to edges, corners, and boundaries between regions of differing colours. Moreover, many feature detectors find features in gradient-rich areas. By maximizing the amount of gradient information in an image, the authors assume that more useful image features will be identified, improving the performance of visual navigation. First, a mapping function is defined that quantifies the gradient magnitudes in one useful information metric:

$$M_{grad} = \sum_i \bar{m}_i, \quad \bar{m}_i = \begin{cases} \frac{1}{N} log(\lambda(m_i - \delta) + 1) & \text{for } m_i \geq \delta \\ 0 & \text{for } m_i < \delta \end{cases}, \tag{3.2}$$

where $m_i = \|\nabla I(i)\|^2$ represents the gradient magnitude at pixel i, $N = \log(\lambda(1-\delta)+1)$

is a normalization factor that maps $\bar{m}_i$ to $[0, 1]$, δ controls the magnitude of the gradients that are considered, and λ controls the degree of linearity in the mapping between the gradient magnitude m_i and the gradient information value $\bar{m}_i$ of pixel i. The optimal exposure value is determined by generating seven synthetic images of the current scene at different exposure values and using the computed metric values of the images in a simple optimization algorithm. The synthetic images are generated using the current image and a gamma-correction technique, $I_{out} = I_{in}^{\gamma}$, where $\gamma = [0.1, 0.5, 0.8, 1.0, 1.2, 1.5, 1.9]$. The metric value of each image is then measured as $M_{grad,\gamma}$, and a fifth-order polynomial is fit to the values. The maximum of this polynomial in the range of sampled γ values is determined as $\hat{\gamma}$ and used to find the corresponding exposure time. The next exposure time of the camera, E_{t+1}, is then set using a simple feedback controller:

$$E_{t+1} = E_t(1 + \alpha K_p(1 - \hat{\gamma})), \quad \alpha = \begin{cases} 1/2 & \text{for } \hat{\gamma} \geq 1 \\ 1 & \text{for } \hat{\gamma} < 1 \end{cases}. \tag{3.3}$$

There are several downsides to this approach. First, this method only adjusts the exposure time, ignoring gain. Additionally, the generation and use of synthetic images for optimization may not accurately reproduce the photometric effects of varying exposure times, such as blur due to motion. The approach also requires careful tuning of metric and control parameters. Finally, the approach is purely reactive, in that it selects the optimal exposure time based on the current image alone.

Similar to [3], Zhang et al. [39] seek to increase the amount of useful image information by maximizing a measure of the image gradients, again assuming that an increase in the amount of gradient information will result in an increased number of matchable image features. The use of gradient information is extended by defining two new metrics. One is simply a *percentile score* of the gradient magnitudes:

$$M_{perc}(p) = \text{percentile}(\{m_i\}_{i \in I}, p), \tag{3.4}$$

where p indicates the percentage of pixels that have gradient magnitudes smaller than M_{perc}. The other metric, called the *soft percentile* metric, is a weighted sum of sorted gradient magnitudes, $\{m_{i\text{th}}\}_{i \in [0,S]}$, and is smoothly differentiable with respect to exposure-time:

$$M_{soft}(p) = \sum_{i \in [0,S]} W_{i\text{th}}(p) \cdot m_{i\text{th}}, \quad W_{i\text{th}} = \begin{cases} \frac{1}{N} \sin(\frac{\pi}{2\lfloor p \cdot S \rfloor} i)^k & i \leq \lfloor p \cdot S \rfloor \\ \frac{1}{N} \sin(\frac{\pi}{2} - \frac{\pi}{2(S - \lfloor p \cdot S \rfloor)} i)^k & i > \lfloor p \cdot S \rfloor \end{cases}, \tag{3.5}$$

where N normalizes the sum of the weights to 1, S is the number of pixels in the image, and k controls how similar M_{soft} is to M_{perc}. The CRF is used to predict how an image will change with exposure time. The CRF, combined with the proposed image quality metric, is used in a simple gradient-ascent algorithm to increase the value of the metric for the next image:

$$E_{t+1} = E_t + \gamma \frac{\partial M_{soft}}{\partial E_t}, \quad \frac{\partial M_{soft}}{\partial E_t} = \sum_{i \in [0,S]} W_{\text{ith}} \frac{\partial m_{\text{ith}}}{\partial E_t}. \tag{3.6}$$

As with [3], the approach is reactive and only adjusts exposure time. Furthermore, the controller only makes adjustments of the exposure time in the direction of the optimal value at each step, rather than directly jumping to the optimal value as in [3].

A more complex image metric is introduced in [40] that is constructed by combining image entropy and gradient information:

$$M_{Kim} = \sum_i g_i, \quad g_i = W_i m_i + \pi(H_i) S_i(H_i) W_i \frac{1}{N} \sum_j m_j. \tag{3.7}$$

In Equation 3.7, H_i is the local entropy at pixel i, and W_i is the entropy weight:

$$W_i = \frac{w_i}{\sum_{i=0}^{N-1} w_i}, \quad w_i = \frac{1}{\sigma} \exp\left\{ \frac{(H_i - \text{mean}(H_i))^2}{2\sigma^2} \right\}. \tag{3.8}$$

An activation function, $\pi(\cdot)$, weighs the importance of image entropy relative to image gradients:

$$\pi(H_i) = \frac{2}{1 + \exp(-\alpha H_i + \psi)} - 1, \quad 0 \leq H_i \leq 1. \tag{3.9}$$

The value α determines the balance between maximizing entropy and maximizing gradient magnitudes. An α favouring image entropy will result in fewer saturated image regions by lowering the exposure time or gain, while an α favouring image gradients will typically increase the exposure time and gain to magnify intensity differences. The value ψ controls the minimum pixel intensity value that can be considered saturated. Finally, $S_i(H_i)$ is a mask that identifies regions of the image that are saturated (and thus, have near zero entropy).

$$S_i(H_i) = \begin{cases} 0 & H_{threshold} \leq H_i \leq 1 \\ 1 & 0 \leq H_i \leq H_{threshold} \end{cases}. \tag{3.10}$$

As with the previous work in [3] and [39], the goal is to adjust the exposure time of the camera such that the metric is maximized. The Kim et al. metric is unique in

that saturation is incorporated through the entropy measurement. Areas in an image with low entropy correspond to regions with similar pixel intensities, which commonly occurs in areas of undersaturation and oversaturation, but can also occur in properly exposed grey regions. An image saturation mask is applied to areas with an entropy value lower than a user-set threshold so that the overexposed and underexposed regions can be compensated for. A user-controlled parameter that weights the importance of minimizing saturation or maximizing gradient information is used. Additionally, the gradient information is weighted through the use of entropy, meaning that areas of the image with large gradients are given a larger weight in the metric calculation, which minimizes the effects of noise. Control of the exposure time is achieved through Bayesian optimization. Bayesian optimization is employed to sample the parameter space so that a minimal number of images are acquired to determine a near-optimal exposure time. The downsides to this approach are that only exposure time is controlled, the requirement that several images need to be captured to construct an objective surface for optimization, and the difficulty in tuning the large number of metric and optimization parameters. Moreover, the rate at which optimal images are obtained is significantly reduced if the environmental lighting conditions continuously change.

Shin et al. [41] propose an even more robust image metric to use as an optimization objective, consisting of a combination of gradient information, entropy, and a noise term. The gradient metric from [3] is used, with additional consideration given to the uniformity of the gradient information throughout the image. The image is divided into N_C cells and the sum of the gradient information of each cell is measured:

$$\hat{M}_{grid,j} = \sum_{i \in C_j} \bar{m}_i, \quad j = 1, 2, ..., N_C. \tag{3.11}$$

The gradient contribution of the metric is determined by taking the ratio of the mean value of the cells and the standard deviation of the cells and multiplying it by a normalization factor K_g:

$$L_{gradient} = K_g \cdot \frac{\text{mean}(\hat{M}_{grid})}{\text{std}(\hat{M}_{grid})}. \tag{3.12}$$

Entropy from [4] is used as a base metric to measure the total information contained in the image and multiplied by a normalization factor K_e:

$$L_{entropy} = K_e \cdot M_{ent}. \tag{3.13}$$

Finally, a noise term is incorporated to measure the noise introduced by increased gain.

The noise component of the metric is determined through a filter-based approach by applying a noise estimation kernel to determine the level of noise in regions of the image. Noise is only measured in homogeneous regions by masking-out areas with large gradients through $\zeta(i)$, and masking out oversaturated and undersaturated regions through $U(i)$:

$$\omega = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}, \quad \zeta(i) = \begin{cases} 1 & \text{for } m_i \leq \delta \\ 0 & \text{for } m_i > \delta \end{cases}, \quad U(i) = \begin{cases} 1 & \text{for } \tau_l \leq I(i) \leq \tau_h \\ 0 & \text{otherwise} \end{cases}. \tag{3.14}$$

The value δ defines the magnitude of gradients that are considered 'large' and removed from the noise calculation metric. The values of τ_l and τ_h define the lower and upper bounds respectively of pixels that are not considered saturated and therefore valid for use in the noise metric calculation.

$$L_{noise} = \sqrt{\frac{\pi}{2}} \frac{1}{N_V} \sum_i \zeta(i) \cdot U(i) \cdot |I * \omega|(i). \tag{3.15}$$

Here, N_V denotes the number of valid pixels in the mask $\zeta \cdot U$. For colour images, the noise values of each channel are averaged. Finally, the three metrics are combined with user-selected parameters to weight each component.

$$M_{Shim} = \alpha \cdot L_{gradient} + (1 - \alpha) \cdot L_{entropy} - \beta \cdot L_{noise}. \tag{3.16}$$

The goal of optimizing the above metric is to obtain images with large gradient and entropy components while minimizing the noise component. For online optimization of camera parameters, the Nelder-Mead algorithm is employed. The Nelder-Mead technique does not require the derivatives of the objective function, unlike aforementioned methods. Nelder-Mead operates by searching the parameter space until a point is reached near the maximum value of the objective function. To test this approach, the authors collect a dataset consisting of different scenes captured with stereo cameras using a range of exposure times and gains. Various image metrics from previous works are compared to the proposed metric by finding the image pairs that correspond to a maximum value of each metric. Using these images, various experiments including pose-estimation and object detection are performed. In these experiments, optimization of the proposed metric results in images producing the lowest reprojection error when used in a pose estimation problem. Further, this method results in the best performance in an object detection experiment. The drawbacks of this approach are the substantial image metric computation time, the requirement that real images need to be acquired to construct

the objective surface, and the difficulty of parameter tuning.[1] Further, the experiments in this work were conducted in static scenes, where lighting was fixed between image samples. The amount of time required to obtain a sufficient number of samples to build the optimization surface would prevent this approach from performing well in a dynamic environment on a moving platform.

3.3 Heuristic Image Metrics

The above methods seek to maximize one or more image metrics, many of them heuristic, that quantify the quality of an image for use in robotic vision tasks. The assumption is made that the images obtained will result in increased robustness of vision tasks in challenging conditions. This assumption, however, is unfounded, in that these metrics are not directly tied into robotic vision pipelines and may not actually improve their performance in all cases. For example, we computed several of the above metrics for a single image sequence from one of our datasets and found that these different metrics indicate widely varying levels of 'quality' for the same images, as shown in Figure 3.1. We normalized the values in the plot to permit direct comparison. We see that the metrics do not agree, which suggests that image quality is subjective, or task-driven. Most of the image metrics from the literature are essentially hand-crafted heuristics that measure quantitative aspects of images but are not tied to the robotic vision pipeline and thus may not directly improve its performance (and might even make it worse).

[1] We found this method to be particularly sensitive to changes in tuning parameters during experimentation

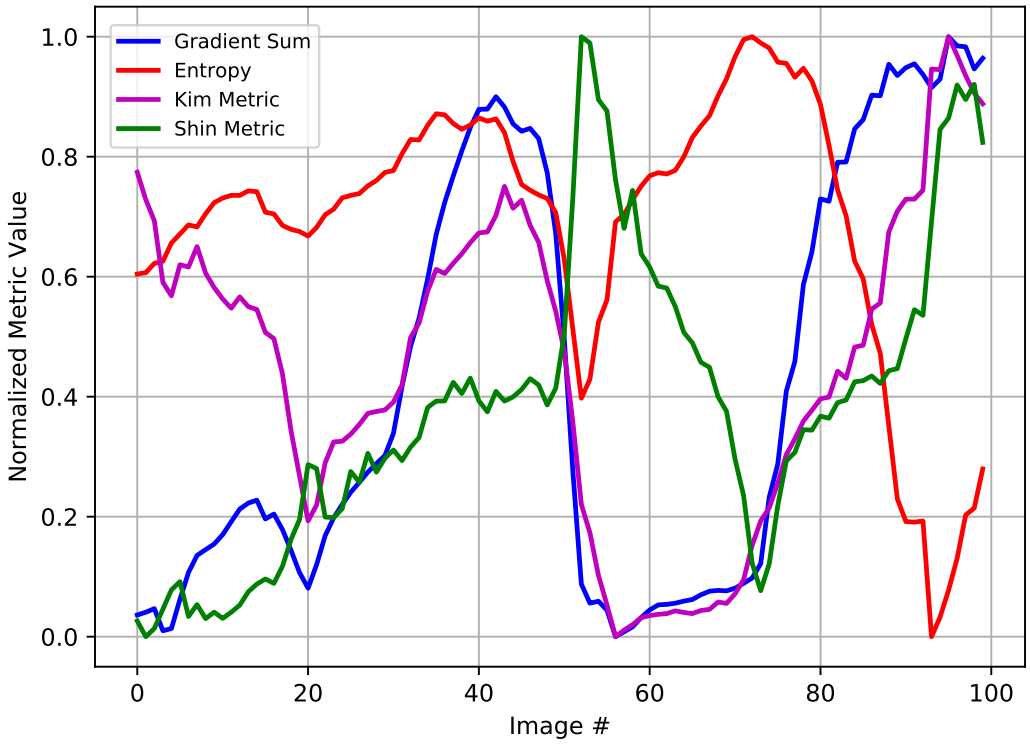


Figure 3.1: A comparison of image metrics from the literature. The plot is of the normalized metric score of each image over a sequence from one of our datasets. It is evident that the image metrics do not agree on image quality. The metrics used are the gradient sum, M_{grad} [3], entropy, M_{ent} [4], entropy-weighted gradient, M_{Kim} [40], and gradient + entropy + noise, M_{Shin} [41].

Chapter 4

Learned Camera Parameter Control

This chapter provides a summary of the problem described in earlier chapters and our motivation for a learning-based solution. We follow up with an overview of our learning-based approach, covering the specifics of our network architecture and our dataset collection process. We next discuss our proposed image metrics used to generate training targets, consisting of the number of inlier feature matches between sequential images, and the number of features in an individual image. We provide motivation for using the number of image features as an image quality metric by providing results from a simple pose estimation experiment. Finally, we outline the network training methodology.

4.1 Problem Formulation

The literature discussed in Section 3.2 describes two general approaches to determining camera exposure time and gain through optimization of heuristic image metrics. Both approaches, however, have their own limitations. The first approach requires sampling real images at different camera parameter settings to construct an objective surface upon which an optimization can be performed. Doing so limits the effectiveness of these methods for real-time applications since images cannot be sampled instantaneously. Collecting samples from the parameter space over several frames can drastically affect the performance of these methods in environments where scene lighting rapidly changes. Dynamic lighting can prevent the optimization algorithm from converging due to the changing shape of the objective surface, which in turn results in poor performance, and sometimes outright failure of a vision algorithm. Examples of robots operating in challenging environmental lighting conditions include mobile robots or drones quickly transitioning from indoors to outdoors and autonomous vehicles (such as cars and trains) entering and exiting tunnels.

The second approach discussed in Section 3.2 alleviates the concerns related to sampling the parameter space in real time by using synthetic images as an alternative. Synthetic images of a scene are generated at various camera parameter settings for each new image acquired and used to construct the objective surface. The near-instantaneous sampling of the parameter space negates the issues of sampling real images, but introduces a source of photometric inaccuracy in the images used for optimization. Synthetic images may not accurately depict real-world variations of image details due to adjustments of camera parameters. For example, it is difficult to accurately model motion blur due to a change in camera exposure time in a single image, as both the speed and direction of motion is unknown.

Both of the above approaches require the respective metrics to be calculated from the samples collected at each time step, which contributes to the algorithm processing time. Additionally, most of these approaches involve algorithms and metrics that contain specific parameters that need to be tuned (requiring some user expertise) depending on the intended application environment. Our method requires far fewer tuning parameters.

We seek to alleviate issues related to the sampling-based optimization of camera parameters by introducing a self-supervised learning-based approach. Our approach can predictively adjust camera parameters, which eliminates the need to collect samples at each time step for optimization. Additionally, we avoid generating synthetic images. We propose a CNN that takes as input a sequence of the past three frames along with the respective camera parameter settings, and regresses adjustments to the exposure time and gain. The sequential input of three images ensures that temporal information is passed into the network to observe how the images are changing, while the inclusion of the current gain/exposure allows the network to decouple the changes due to varying parameters from the changes due to varying external illumination. The camera parameter adjustments output by our network yield images containing a higher number of features, resulting in more sequential inlier feature matches. The only additional processing required by our method is a forward pass through the network after each new image is captured. Using modern graphics processing units, the forward pass only contributes a small amount of additional time to the image acquisition process. We demonstrate that our network can be trained offline in a self-supervised manner using the VO front end to generate regression targets directly from real images. We test our approach in real-world scenarios, demonstrating its performance in both static and dynamic lighting conditions.

As a proof-of-concept, we first implement, train, and test a network that adjusts a single camera parameter (exposure time) to reduce the complexity of the problem. Following the successful implementation of the single-parameter network, we then move

on to the dual-parameter (exposure time and gain) problem.

4.2 Approach and Methodology

To train our networks, we devise a unique self-supervised training methodology incorporating the VO front end, similar to that of [6]. Using this methodology, we train a CNN to regress values of exposure time in the single-parameter case, and both exposure time and gain in the dual-parameter case. Our training methodology necessitates the collection of novel datasets for which training targets can be generated offline. Section 4.2.1 outlines the network architecture for both the single-parameter and dual-parameter networks. Section 4.2.2 discusses the details of the two novel datasets used for training and the methodology used to collect them. Section 4.2.4 provides an analysis of our proposed metrics derived from the VO front end used to generate training targets. We also provide motivation for increasing the number of features in images. Finally, section 4.2.5 covers the label generation process and the training approach for the single-parameter and dual-parameter case. The experimental results for each case are described in Chapter 5.

4.2.1 Network Architecture

We first investigated the feasibility and effectiveness of a single-parameter network. We then modified the network structure, training dataset, and training methodology to accommodate the dual-parameter case.

Single-Parameter Architecture

We selected exposure time as the parameter to adjust in the single-parameter case as it directly impacts the brightness and sharpness of images. Our network takes as input a sequence of the previous three images $\{I_t, I_{t-1}, I_{t-2}\}$ along with their respective exposure times $\{E_t, E_{t-1}, E_{t-2}\}$, and outputs a predicted exposure value, $\hat{E}_{t+1}$. The camera then adjusts its exposure to this value and acquires the next image. We fixed the gain to a value of 10 dB for data collection, training, and testing as this value is roughly the average of typical gain values selected by the camera's onboard automatic gain control algorithm for both indoor and outdoor scenes.

The initial challenge for this task was to determine the architecture of the network. Ensuring that the network had sufficient representational capacity to learn the relation

between image structure and exposure time while making certain that the network did not over-fit the dataset was an important consideration.

To ensure that the network had sufficient capacity, we first investigated the use of a pre-trained deep residual network (ResNet) [47] with 18 layers. The pre-trained fully-connected (FC) layers used for image classification were removed and replaced by untrained FC layers that reduced the final convolutional layer output to a single value. Since the ResNet architecture accepts 3-channel RGB images as input, we initially tried using only a single image as an input. This approach did not generalize well, so we adopted a multi-image input, which consisted of three sequential images. We reasoned that a multi-image input would capture the dynamics in the environment (e.g., approaching the entrance or exit of a tunnel) and the corresponding exposure values would provide context for the training targets to better facilitate learning. Various iterations of training were completed, each with a varying amount of the pre-trained convolutional layer weights frozen in order to change the representational capacity. Initial training results indicated that fully freezing all of the convolutional layers of the pre-trained network (i.e., using the pre-trained weights for the convolutions and only training the FC layers) was not beneficial. The frozen parameters had been trained for an image classification task, and were not suitable for regressing exposure values. Conversely, completely unfreezing the network (i.e., training the entire network) tended to result in an over-fitting of the training data. Consequently, we froze only the first several layers of the ResNet, as these layers are responsible for detecting simple image features and structures such as corners and edges. The deeper layers that are responsible for extracting more application-specific information were unfrozen and retrained.

One of the main issues with using a pre-trained ResNet was that the input dimensions could not be altered. The pre-trained ResNet was designed to process three channel RGB image inputs. Using a sequential stream of images as a single input was not possible without altering the structure of the network, which would have eliminated the advantage of using pre-trained weights. Additionally, the input exposure values could not be added directly to the input of the convolutional layers. To maintain the advantage of using the pre-trained weights, we passed each image in the input sequence through the convolutional layers independently. For each input image, the corresponding exposure value was appended to the one-dimensional output of the convolutional layers. Finally, the outputs of the convolutional layers for each image were concatenated and then passed through the FC layers. To accommodate the larger input, the FC layers were adjusted accordingly. Preliminary results indicated that this approach did not work well. We suspect that the poor performance of this approach was due to the input images

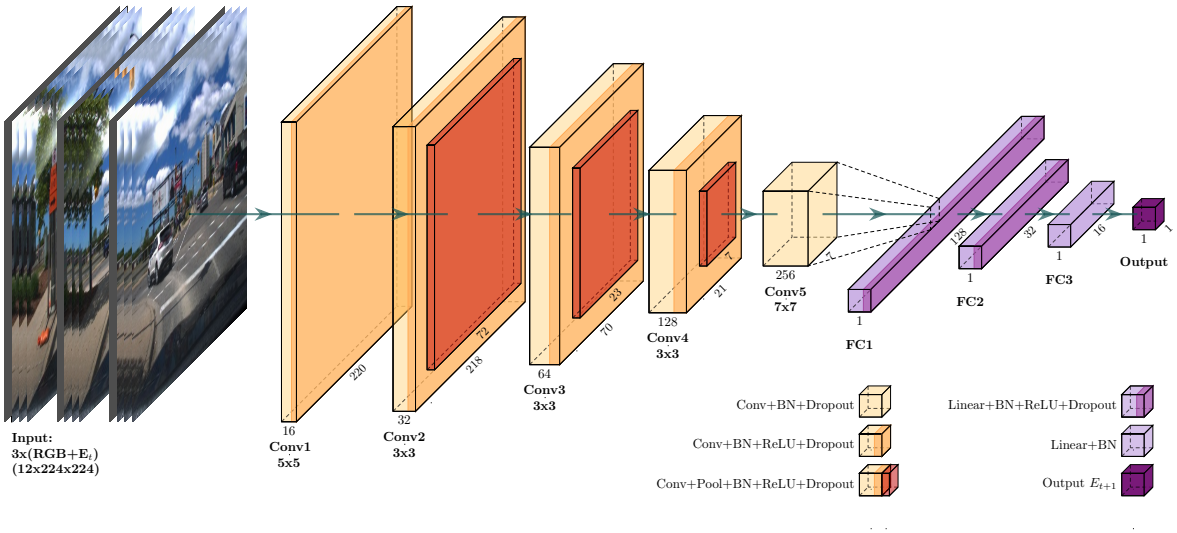


Figure 4.1: The custom CNN developed for estimating the next exposure E_{t+1} given an input sequence $\{I_t, I_{t-1}, I_{t-2}\}$ and corresponding exposures $\{E_t, E_{t-1}, E_{t-2}\}$.

being processed independently in the convolutional layers. Processing the input images independently resulted in a loss of the contextual temporal information contained in the three sequential images. Finally, the size of the ResNet architecture resulted in a relatively slow forward pass (see Table 4.1). The poor performance of the ResNet led us to investigate an alternative network architecture.

Table 4.1: Network processing time comparison - modified ResNet vs. custom CNNs for a three image (+ parameters) input sequence.

Network Architecture	Forward Pass (ms)	Parameters
Modified ResNet-18 - single	$\sim$ 11.8	12,726,337
Custom CNN - single	$\sim$ **1.5**	**1,746,609**
Custom CNN - dual (large)	$\sim$ 2.2	6,150,322
Custom CNN - dual (small)	$\sim$ **1.3**	**1,546,498**

A custom lightweight CNN architecture was developed that addressed the disadvantages of the ResNet approach (Figure 4.1). First, the input was structured to process a 12-dimensional input consisting of the concatenation of each three-channel RGB image, along with a fourth channel representing the corresponding exposure (described in Section 4.2.5). Next, the number of layers was significantly reduced to help speed up the processing time and reduce over-fitting. Table 4.1 demonstrates the performance improvement using an NVIDIA GeForce GTX 1050 Ti laptop graphics card. Our custom CNN only adds an additional 1.5 ms to the overall image processing time, while the modified ResNet-18 architecture added almost 12 ms, resulting in a 87% reduction

in processing time. Additionally. the overall size of the network was decreased by 86%. The network processes all three images as one single input and outputs a predictive adjustment to the exposure time at each frame. Our custom CNN generalized well during training relative to the ResNet-18 architecture due to its smaller size and custom input architecture.

Dual-Parameter Architecture

After successfully implementing the lightweight custom CNN architecture for the single-parameter case, we sought to modify and extend this approach. As we discuss in Chapter 5, many of the downsides of using the single-parameter network are due to controlling only one camera parameter. To improve upon the experimental results of the single-parameter approach, we adapted the network to also output adjustments to camera gain.

Adding predictive gain capability to the network necessitated modifications to the network structure. With the addition of gain, the network input contained not only three RGB images and their corresponding exposure times, but also the corresponding gain values of each image. This addition increased the size of the input from twelve channels to fifteen channels. Additionally, the network was required to output both gain and exposure adjustments. To accommodate these changes, the network structure was adapted to handle the larger input and to output two values. Two versions of this adapted network were developed: a larger network containing additional convolutional layers, and a smaller network with fewer layers. These two networks were both trained using the same input data to determine if there was an advantage to using the larger network. The results of the training indicated that there was little difference in validation performance. In addition, the smaller network contained fewer parameters, resulting in a faster forward pass as shown in Table 4.1. The smaller network also had the added bonus of faster training speeds. For these reasons, the smaller network was selected for our experiments. The architecture of the smaller network is shown in Figure 4.2.

Although the selected network was the smallest out of all tested, including the single-parameter network, we still found that the training error was reduced to a small value and that the network generalized well compared to both the single-parameter version and the larger dual-parameter version.

4.2.2 Dataset Collection

There are currently no publicly-available datasets that are suitable for the supervised training of a network that adjusts camera exposure time and gain for visual navigation

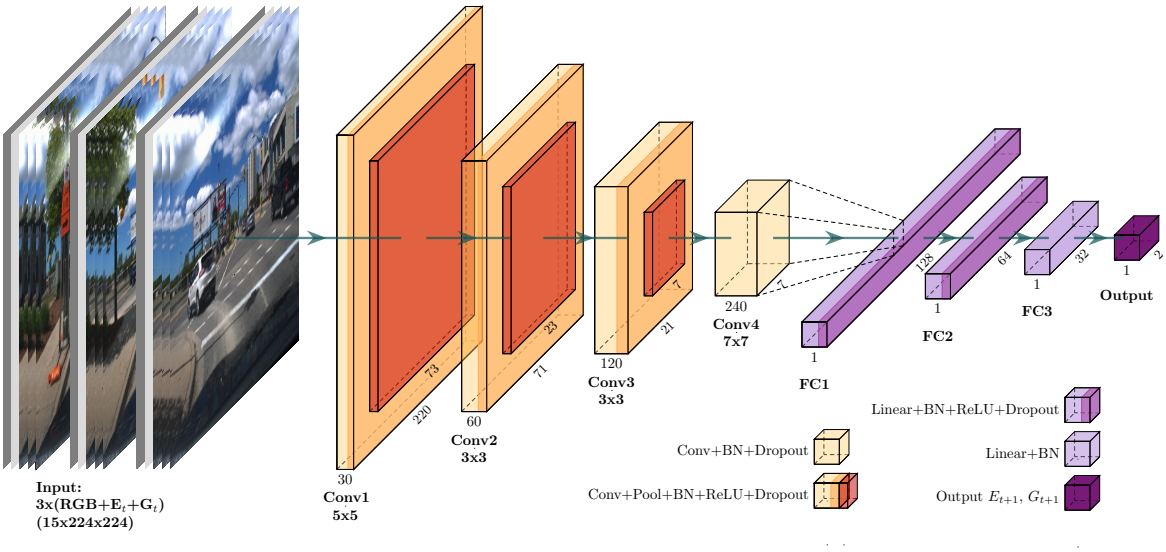


Figure 4.2: The smaller version of the custom CNN developed for estimating the next exposure E_{t+1} and gain G_{t+1} given an input sequence $\{I_t, I_{t-1}, I_{t-2}\}$, corresponding exposures $\{E_t, E_{t-1}, E_{t-2}\}$, and gains $\{G_t, G_{t-1}, G_{t-2}\}$.

algorithms. Consequently, we collected a pair of novel datasets and used them for offline self-supervised training: one dataset for the single-parameter case and the other dataset for the dual-parameter case.

Single-Parameter Dataset

To generate training targets for the single-parameter network, a dataset consisting of several unique trajectories was collected. At each pose, the camera was fixed in place and roughly fifty images were collected. During image acquisition, the gain was held constant, while a variety of appropriate exposure values were sampled from predetermined ranges. These ranges depended on the environmental lighting conditions and were determined empirically by testing to see when the images became oversaturated or undersaturated. The images were acquired using a machine vision camera mounted to a camera tripod. The camera lens aperture was fixed to a small value to ensure a deep depth of field. RGB images were acquired at a 2048×1536 pixel resolution. Examples of images from a single pose from three different trajectories are shown in Figure 4.3. After collecting images at a single pose, the camera was manually moved to the next pose in the trajectory, typically 10–50 cm forward and with only a small amount of rotation. Additionally, some trajectories were obtained with the camera held stationary while a door to the outdoors was opened or closed. The poses in these stationary trajectories consist of the door opening growing larger or smaller over time. Examples of these trajectories are

shown in Figure 4.4.

The trajectories were collected in both indoor and outdoor environments and during indoor-outdoor transitions, with rotational and linear movements of the camera between poses. The purpose of incorporating variety in the training dataset was to assist in helping the network to generalize. Thirty-six trajectories were collected, containing a total of 68,265 images. It is important to note, however, that none of the images in this dataset contain motion blur due to increased exposure time as all images were acquired using a stationary camera and there were no dynamic objects within the captured scenes.

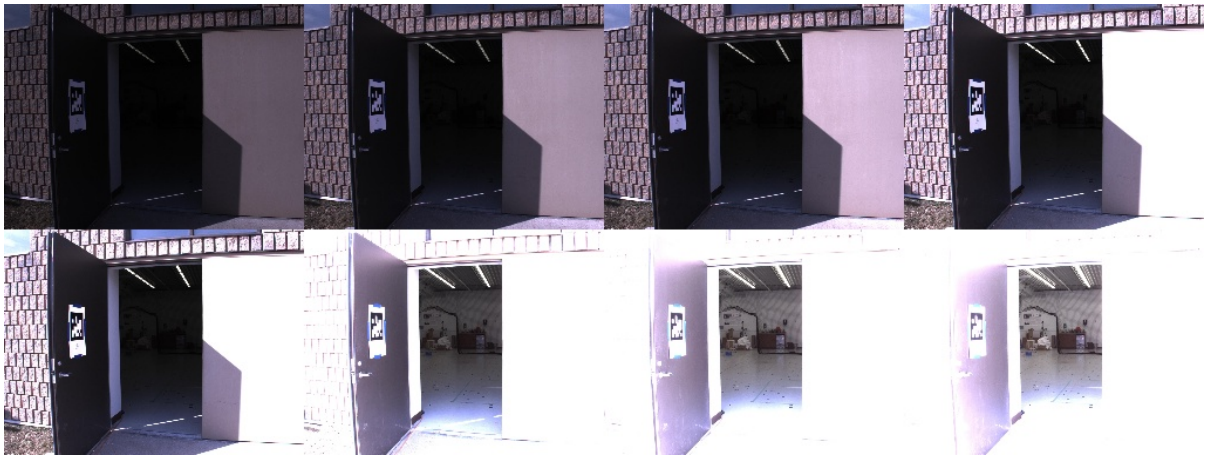
The single-parameter dataset allows for the analysis of a scene captured under a variety of exposure values. Using the VO front end (discussed in section 4.2.4), the settings that yield the best image from each pose can be determined and used as a training target. Furthermore, our training approach is highly data efficient. Many combinations of properly exposed, overexposed, and underexposed input images at each pose can be used as training inputs, allowing for a large number of diverse training samples. Thus, the network learns the relation between exposure time and image quality in a variety of environments. During training, images were downsampled to a size of 224 $\times$ 224, however, training labels were generated with full sized images.

Dual-Parameter Dataset

In the dual-parameter case, the data collection approach used for the single-parameter network proved to be infeasible for a number of reasons. First, adding a second dimension, in the form of camera gain, to the camera parameter space exponentially increased the number of images that would need to be acquired at each pose. Rather than fifty images sampled at each pose, there would need to be $\sim$2500 images, dramatically increasing the amount of time required to collect the dataset and introducing an even larger increase in processing time for generating labels at each pose. Additionally, it would be hard to ensure that the scene remains relatively static during image acquisition, especially outdoors. For instance, capturing images at a frame rate of 30 Hz would take roughly 83 seconds per pose and any changes to the scene would make it difficult to fairly compare images. Second, motion blur due to increased exposure time was not accounted for in the single-parameter case. One of the main advantages of controlling gain is that gain can provide an increase to image brightness without increasing the exposure time. Maintaining a lower exposure time is useful in applications where an increase would lead to severe image blur such as in indoor, low-light, environments or when moving at high speeds. Collecting images from a stationary camera tripod prevents any motion blur from being introduced into the dataset. Moreover, with a stationary camera, it is likely

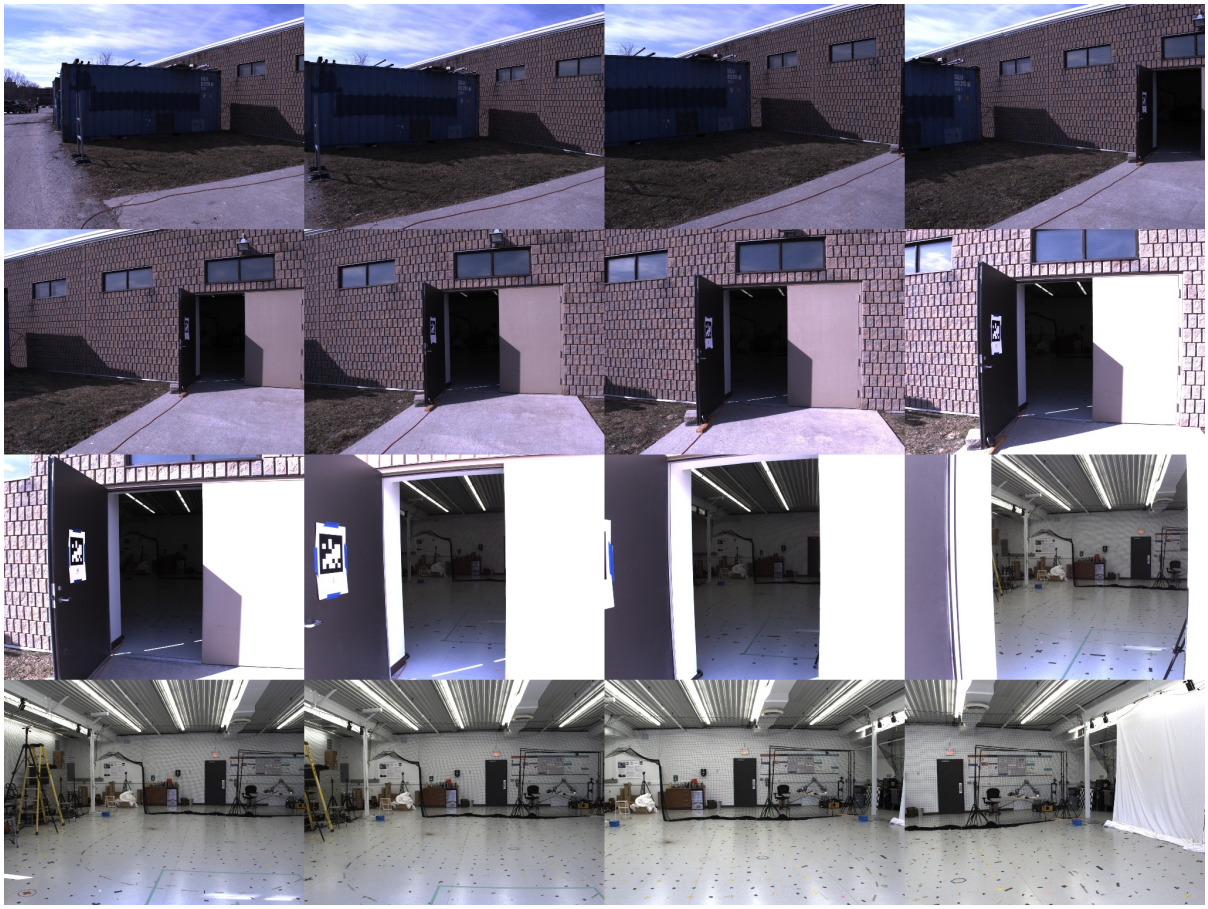
(a) Outdoor Trajectory

(b) Indoor-Outdoor Transition



(c) Outdoor-Indoor Transition

Figure 4.3: Examples of the images acquired at three poses from different trajectories. At each pose, the same scene was captured with a variety of exposure times. Important image features during transitions can be found both indoors and outdoors, as in (b) and (c), and are highly dependent on exposure time.



(a) From left to right: An outdoor-indoor trajectory. The camera is moved between poses.

(b) From left to right: A stationary trajectory consisting of a door opening between poses.

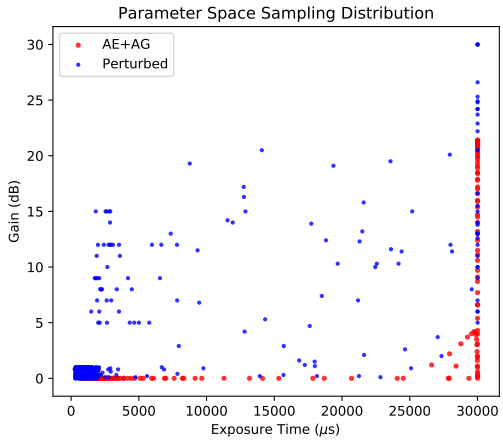
Figure 4.4: Examples of trajectories of poses in the dataset. For each pose, there exist many images captured at various exposure times.

that different combinations of gain and exposure time would result in very similar images, introducing some ambiguity in the training targets and reducing the data efficiency of the training. Finally, the exponential increase in the number of images required could introduce storage concerns depending on the system and number of trajectories desired.

For the above reasons, an alternative approach for the collection of a dual-parameter dataset was devised. We incorporated motion blur into our dataset by acquiring images from a moving vehicle. Our data collection setup consisted of using two identical cameras, rigidly mounted as closely as possible next to each other and with identical lens apertures (Figure 4.5). This setup ensured that images were obtained from the nearly the same viewpoint, as if acquired by a single camera. The cameras were fixed to a vehicle in a forward-facing direction and driven through a number of environments. To ensure that we sampled values in a good region of the parameter space while moving, our data collection method consisted of sampling the parameter space close to the outputs of a built-in auto-exposure and auto-gain controller (hereafter AE+AG). One camera (Camera 1) continuously acquired images using AE+AG, which ensured that half of the collected images were of reasonable quality in most cases. At each frame, the values of the exposure time and gain for Camera 1, E_i and G_i, were extracted and modified by a scaling factor. These perturbed settings, $E_{i,\text{pert}}$ and $G_{i,\text{pert}}$ were applied to the second camera (Camera 2) and used to acquire an image. Thus, for each camera pose, we acquired two images of a near-identical scene; one image was captured using AE+AG, and one was captured with perturbed settings. At each pose, therefore, the target parameters were generated from either the perturbed image, or the AE+AG image. Although only two images were acquired at each pose as opposed to fifty in the single-parameter case, there were far more poses contained in each trajectory. Further, this method of collecting images while moving ensured that motion blur was incorporated in the dataset. Effort was taken to ensure that the acquired images were useful through careful selection of the perturbation method, which we describe next.

It was important that the method by which the AE+AG settings were perturbed sampled the parameter space to find better image settings but did not select settings that resulted in very poor images. When designing the sampling method, we needed to consider the trade-off between the amount by which the parameter space was explored and the quality of the images. In addition to the scale of the perturbations, the direction of the perturbations was also considered. The two parameters could each be increased or decreased, resulting in four possible regions of the parameter space from which to generate samples. The magnitude of these perturbations was determined using two different methods to ensure a variety of data. The first method consisted of randomly sampling

(a) Camera rig used for experiments and data collection.



(b) AE+AG settings (in red) and perturbed settings (in blue) sampled during a tunnel transition.

Figure 4.5: (a) The camera rig and (b) parameter setting sampling distribution from a tunnel transition trajectory. The majority of AE+AG exposure times were located at 0 μs and 30000 μs.

a scaling factor from a uniform distribution between empirically determined values. To ensure an even distribution of perturbations, the four regions of the parameter space were sampled in a cycle, repeating once over every four frames.

$$E_{i,\text{pert}} = E_i * (1 \pm U(0, 0.5)), \quad G_{i,\text{pert}} = G_i * (1 \pm U(0, 0.5)), \tag{4.1}$$

for exposure time and gain separately. We found that scaling in the range of 0.5–1.5 resulted in a good trade-off between exploration and high quality images. In the case of $G_i = 0$ (only occurred for gain), we applied an addition (rather than scaling) of a small (< 2) or a large (> 5) value drawn from one of two uniform distributions to ensure a wide variety of samples. The second method for scaling the perturbations was to use a heuristic search strategy based on the average intensity of the AE+AG image. In this approach, the average intensity, J, of the image was calculated and used to inform the direction of the scaling:

$$E_{i,\text{pert}} = \begin{cases} E_i * (1 + U(0, 0.5)) & \text{for } J \leq 128, \\ E_i * (1 - U(0, 0.5)) & \text{for } J > 128. \end{cases} \tag{4.2}$$

We applied the same function above for gain. If the current AE+AG image was bright, the parameters were adjusted to sample from regions of the parameter space that resulted in

(a) A trajectory captured using the four-quadrant cycle method.

(b) A trajectory captured using the heuristic average-intensity sampling method.

Figure 4.6: Examples of trajectories of poses in the dual-parameter dataset. The top row in each example corresponds to four sequential AE+AG images while the bottom row consists of the corresponding perturbed images. We seek to identify the best image within a window of four frames as our target exposure time and gain values.

images with reduced brightness, and vice versa. The downside to the heuristic searching approach was that both the gain and exposure time were adjusted in the same direction, meaning that two regions of the parameter space were never sampled from. Consequently, most of the data was collected using the first approach.

Examples of images from the dual-parameter dataset are shown in Figure 4.6. The images in Figure 4.6a were acquired using the first parameter sampling approach in which the perturbed examples were drawn from all four regions of the parameter space. The images in Figure 4.6b were acquired using the second parameter sampling approach. The images in Figure 4.6b are all brighter than the AE+AG images, which had an average

intensity of less than 128. This dataset consists of RGB images acquired at a 2048 $\times$ 1536 pixel resolution. The allowable range of exposure time values was 75–30000 μs while the allowable range of gain values was 0–30 dB. A total of thirty-eight trajectories were collected, containing 47,848 images.

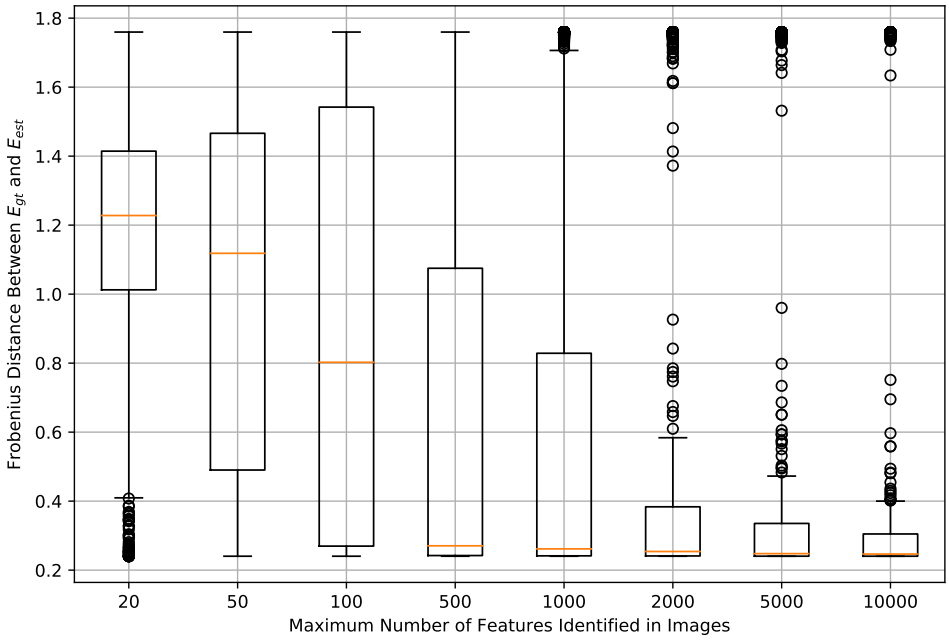
4.2.3 Using Image Feature Count as a Metric

Utilizing the camera settings that result in a maximum number of inlier feature matches between sequential images is an intuitively useful target for a VO application. It cannot be assumed, however, that maximizing for features in an individual image alone will result in good images for VO.

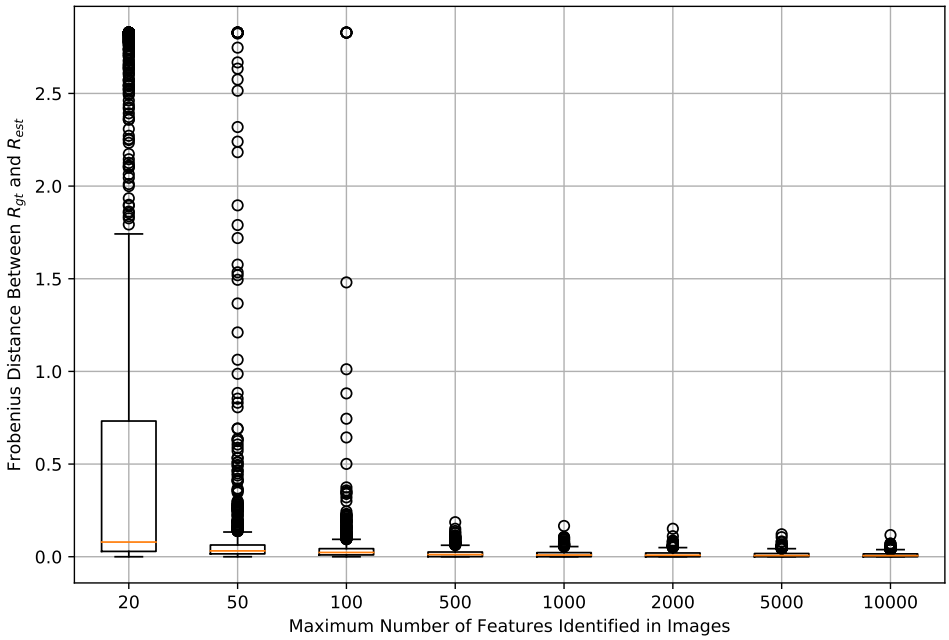
To determine if using the number of features in an image as a metric is warranted, we performed a pose estimation experiment using the KITTI dataset [48]. We sought to determine if an increased amount of image features results in improved pose estimates. To test this notion, we utilized a stereo image pair from the KITTI dataset, and extracted a limited number of features in each image. We then matched these features and computed the inlier feature matches. Using the inlier feature matches, we computed the essential matrix $\mathbf{E}$ and then extracted the correct rotation from $\mathbf{E}$ using the cheirality constraint [49]. The estimated essential matrix and rotation matrix were then compared to the ground truth essential matrix and rotation matrix, respectively. We used the rotation matrix rather than the transformation as the decomposition of the essential matrix only yields the translation up to scale. We used the Frobenius distance to determine the accuracy of essential matrix estimate $\hat{\mathbf{E}}$ (and rotation matrix) relative to the ground truth $\mathbf{E}^*$:

$$||\hat{\mathbf{E}} - \mathbf{E}^*||_F = \sum_{i=1}^{3}\sum_{j=1}^{3} |(\hat{e} - e^*)_{ij}|^2. \tag{4.3}$$

To test how the number of features affects pose estimation accuracy, we repeated the essential matrix estimation test using a varying upper limit on the maximum allowable number of features in each image, ranging from 20 to 10000 features. For each maximum allowable feature count, this test was performed using 1000 stereo image pairs and the results were then averaged. We obtained the stereo image pairs by randomly sampling 100 image pairs from 10 KITTI trajectories. Figure 4.7 shows that, in general, more features available in images led to better pose estimates. Despite other works showing that good VO accuracy can be achieved on the KITTI dataset using a small amount of features [50], we note that in general, having more features increases the chance that a smaller subset of 'good' features is present in the image.



(a) Essential matrix estimation errors.



(b) Rotation matrix estimation errors.

Figure 4.7: (a) Essential and (b) rotation matrix estimation errors averaged over 1000 stereo image pairs for various upper limits on the number of features allowed to be extracted from images. Increasing the maximum number of features results in a more accurate pose estimate.

4.2.4 An Analysis of Image Metrics Derived from VO

The literature provides several different image quality metrics based on various image characteristics. We make note that our use of the term 'metric' refers to a (qualitative) measure of image quality rather than a distance measurement in the mathematical sense. Figure 3.1 showed that the metrics from the literature do not agree on what constitutes a favourable image. Instead of using these existing metrics, we take inspiration from [6] and use the VO front end to inform the quality of the images. Making use of image quality metrics derived from the VO pipeline will naturally result in training targets that are suitable for training a network to improve the quality of acquired images for VO applications. As discussed in Section 1.2, using pose estimation accuracy as a training target requires a prohibitively large amount of ground truth training data. Due to this limitation, we used the front end to generate training targets. This notion, of using the intended visual algorithm to determine which images are best, is a general concept that can be extended to other visual algorithms [6]. The VO front end consists of feature extraction and matching, so we selected four variants of metrics to investigate based on the number of image features and inlier feature matches.

To analyze and compare these image metrics, we used our single-parameter dataset, as it contains images of the same scenes captured with a wide variety of exposure values. For a trajectory in our dataset, we found the 'optimal' image (and corresponding camera parameters) in each pose as determined by one of our proposed metrics described below. We denote the set of optimal images over the entire trajectory for a specific metric as an $\mathcal{M}$-trajectory. That is, the $\mathcal{M}$-trajectory consists of the set of single images from each pose that returned the highest scores using a particular image quality metric. Thus, there exists one $\mathcal{M}$-trajectory per metric for each trajectory in our dataset. We then used the VO front end to compare the resulting $\mathcal{M}$-trajectories by measuring the number of sequential inlier feature matches over each.

The first metric we investigated was simply the number of ORB features within an image I at a pose t:

$$M_{feat}(I_t) = n_{features}(I_t). \tag{4.4}$$

The best image within the set of images collected at each pose was the one that contained the largest number of ORB features. We created an $\mathcal{M}$-trajectory using the best image (and corresponding exposure time) in each pose.

The second metric was a measure of the uniformity of the ORB features within an image, similar to the uniformity of the gradients from [41]. We divided the image into S

subregions each denoted as R_j for $j = \{1, ..., S\}$, counted the number of ORB features within each subregion, n_j, and then calculated the mean number of features and the standard deviation of the number of features between subregions. We then calculated a metric over the entire image as follows:

$$n_j = n_{features}(R_j), \quad j = 1, 2, 3...S, \quad \mathbf{n} = [n_1, n_2, ..., n_S] \tag{4.5}$$

$$M_{uni}(I_t) = \frac{\text{mean}(\mathbf{n})}{\text{std}(\mathbf{n})}. \tag{4.6}$$

We selected the image with the largest value as it corresponded to the image with the largest number of features that are uniformly distributed. Again, we created an $\mathcal{M}$-trajectory using the best images at each pose. We note that uniformly-distributed features are a valuable metric in environments where there are useful features contained throughout an image, such as those found indoors. However, measuring the uniformity of images features may not be a suitable metric for environments that contain feature-sparse regions such as the sky in outdoor applications.

Next, we investigated inlier feature matches between poses as an image quality metric. Intuitively, selecting camera settings that result in images that yield the maximum number of inlier feature matches between frames should result in ideal images for VO, as feature-based VO pipelines depend on good feature correspondences for accurate motion estimates. Using our dataset, we investigated this hypothesis using two methods. For the first method, referred to as the *sequential* method, we selected a good initial image in the first pose, I_t^*, then found the number of inlier feature matches with an image in the second pose I_{t+1}:

$$M_{seq}(I_t^*, I_{t+1}) = n_{matches}(I_t^*, I_{t+1}). \tag{4.7}$$

The image in the second pose that corresponded to the largest number of inlier feature matches was determined as the best image for the pose pair $\{t, t+1\}$. We then used the second image as the best image I_{t+1}^* for matching with the next pose $t+2$ and repeated this process until the end of the trajectory. Using this metric, consistent feature matches are propagated forward through the $\mathcal{M}$-trajectory.

The second inlier feature match metric, referred to as the *gridsearch* method, treated a pair of sequential poses as independent from the rest of trajectory. We then found the image pair from the two poses that resulted in the highest number of inlier feature matches,

$$M_{grid}(I_t, I_{t+1}) = n_{matches}(I_t, I_{t+1}). \tag{4.8}$$

We matched every image from the first pose to every image in the second pose, and found

the image pair resulting in the maximum number of matches. The image in the second pose was selected as the best image for that pair and added to the $\mathcal{M}$-trajectory. This process was repeated for each sequential pose pair until the end of the trajectory.

After collecting $\mathcal{M}$-trajectories for each of the above metrics, we were able to perform an analysis of the obtained images. The findings of our analysis are shown in Table 4.2 and Table 4.3. For each $\mathcal{M}$-trajectory, we measured the *median number of inlier matches* (Table 4.2), and the *minimum number of inlier matches* (Table 4.3) between sequential frames.[1] The $\mathcal{M}$-trajectories yielding the highest number of inlier matches are highlighted in bold. Table 4.2 shows that using the number of features, M_{feat}, and the gridsearch inlier match method, M_{grid}, both resulted in the best trajectories for median inlier matches while Table 4.3 shows that the sequential inlier match method, M_{seq}, yielded the highest minimum number of inliers in most cases. The M_{uni} performs poorly in all cases, likely due to there not being many features in the sky.

Although Table 4.3 might suggest that the sequential metric is preferred, as it yielded the highest number of minimum inlier feature matches, further investigation reveals that this is not the case, especially during transitions. The sequential inlier metric is not ideal because it selects the images (and the corresponding camera settings) that propagate the initial feature matches forward through the trajectory. In the case of a transition from indoors to outdoors, for example, the images selected are those that maintain the indoor parameter values, as these values ensure the same indoor features are matched over time. When the camera transitions, the image settings that properly expose features outdoors are never selected. Features that are identified in the outdoor environment while the camera utilizes the indoor parameter settings are the features that are maintained as the trajectory continues to proceed outdoors. The settings used to identify these features initially (i.e., the indoor settings) are maintained through the transition. Maintaining indoor feature tracks results in severely overexposed images during indoor-outdoor transitions and underexposed images during outdoor-indoor transitions. We see evidence of improperly exposed images in plots of the metric statistics during an indoor-outdoor transition shown in Figure 4.8. The bottom two plots show the image entropy and gradient magnitudes as a measure of the amount of information in the images. The amount of information in the images determined by the sequential metric is greatly reduced during and after the transition (beginning around Pose 9). The sequential method does not reward exploration of the parameter space, favouring instead to maintain the selected

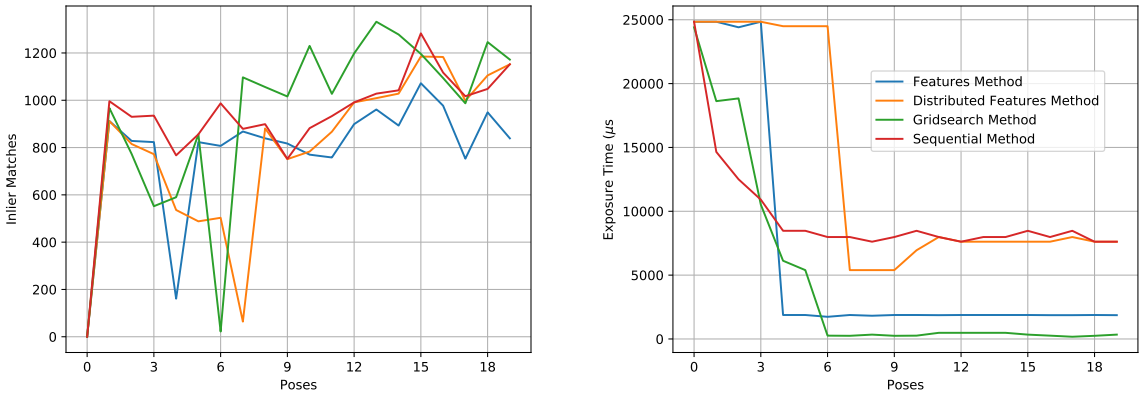
[1]The trajectories presented in these tables are from the single-parameter dataset described in Section 4.2.2 and contain between 10–167 poses. These trajectories consist of indoor, outdoor, and transitional sequences.

Table 4.2: Median number of inlier feature matches across each trajectory in the single-parameter dataset. Each $\mathcal{M}$-trajectory was created using the best images identified by the respective metric.

	Median Inlier Feature Matches			
Trajectory	M_{feat}	M_{uni}	M_{grid}	M_{seq}
Indoor Hallway	230	70	229	**233**
Indoor Lab	512	363	502	**516**
Indoor Linear A	**455**	260	452	**455**
Indoor Linear B	642	570	**645**	644
Indoor Linear C	574	525	579	**584**
In-Out Moving A	**399**	195	385	305
In-Out Moving B	173	143	203	**231**
In-Out Moving C	**247**	112	244	223
In-Out Moving D	438	379	424	**445**
In-Out Moving E	**422**	357	361	403
In-Out Moving F	356	360	**377**	338
In-Out Moving G	562	444	**563**	431
In-Out Rotating A	**192**	92	190	91
In-Out Rotating B	**193**	103	**193**	173
In-Out Rotating C	**221**	94	209	173
In-Out Stationary A	333	155	**355**	185
In-Out Stationary B	**248**	191	240	234
In-Out Stationary C	352	160	**384**	167
In-Out Stationary D	425	226	**644**	243
In-Out Stationary E	165	158	170	**173**
Indoor Path A	**425**	283	422	421
Indoor Path B	**316**	239	306	311
Indoor Path C	479	329	**493**	**493**
Indoor Rotating A	526	383	523	**529**
Indoor Rotating B	**324**	213	319	**324**
Indoor Stationary A	**723**	55	**723**	**723**
Indoor Stationary B	**885**	776	**885**	**885**
Indoor Window A	352	264	**366**	319
Indoor Window B	370	311	**399**	386
Out-In Moving A	**506**	367	471	482
Out-In Moving B	323	148	**329**	315
Out-In Moving C	484	360	**486**	470
Out-In Rotating A	296	135	**297**	291
Out-In Rotating B	**399**	118	375	336
Outdoor Moving A	533	497	562	**570**
Outdoor Moving B	514	453	553	**556**
Total Maximums	**15**	0	**15**	14

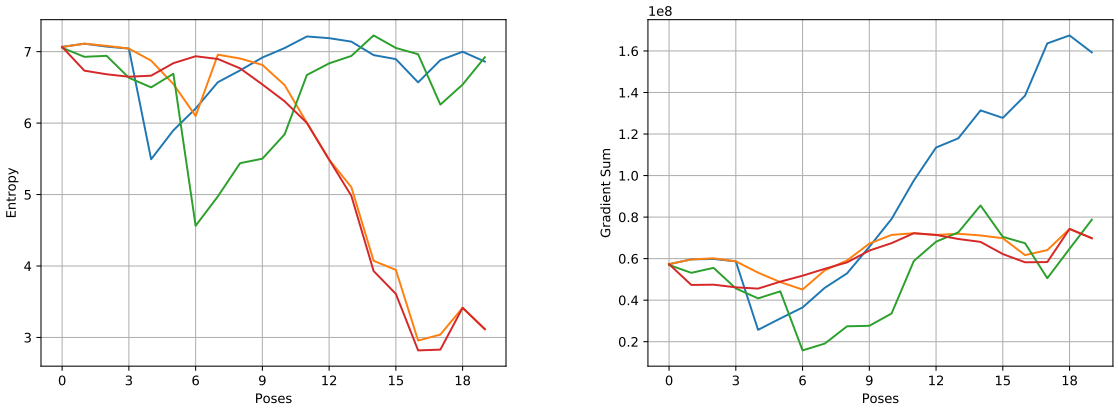
Table 4.3: Minimum number of inlier feature matches across each trajectory in the single-parameter dataset. Each $\mathcal{M}$-trajectory was created using the best images identified by the respective metric.

	Minimum Inlier Feature Matches			
Trajectory	M_{feat}	M_{uni}	M_{grid}	M_{seq}
Indoor Hallway	**7**	0	**7**	**7**
Indoor Lab	169	20	**172**	170
Indoor Linear A	27	8	**30**	27
Indoor Linear B	549	25	**551**	**551**
Indoor Linear C	328	212	324	**345**
In-Out Moving A	95	14	**112**	**112**
In-Out Moving B	**18**	**18**	17	17
In-Out Moving C	25	17	51	**90**
In-Out Moving D	**35**	10	25	31
In-Out Moving E	18	16	18	**28**
In-Out Moving F	18	19	**21**	19
In-Out Moving G	**22**	13	19	19
In-Out Rotating A	15	17	15	**24**
In-Out Rotating B	**52**	15	46	48
In-Out Rotating C	16	8	0	**47**
In-Out Stationary A	55	21	82	**148**
In-Out Stationary B	171	33	184	**195**
In-Out Stationary C	44	40	75	**111**
In-Out Stationary D	152	81	**174**	**174**
In-Out Stationary E	122	127	125	**142**
Indoor Path A	247	73	**254**	251
Indoor Path B	144	35	**155**	**155**
Indoor Path C	**294**	57	**294**	**294**
Indoor Rotating A	333	66	339	**347**
Indoor Rotating B	179	107	178	**187**
Indoor Stationary A	712	45	**718**	**718**
Indoor Stationary B	867	768	861	**873**
Indoor Window A	79	51	**103**	**103**
Indoor Window B	28	14	47	**60**
Out-In Moving A	23	**32**	26	22
Out-In Moving B	21	**64**	22	27
Out-In Moving C	24	18	**25**	23
Out-In Rotating A	26	23	19	**29**
Out-In Rotating B	27	20	**28**	22
Outdoor Moving A	326	222	277	**343**
Outdoor Moving B	324	268	441	**445**
Total Maximums	6	3	14	**24**



(a) Inlier ORB matches.

(b) Exposure time.



(c) Image entropy

(d) Image gradients

Figure 4.8: (a) Inlier matches, (b) exposure times, and corresponding image (c) entropies and (d) gradient magnitudes for our four metrics of interest over an indoor-outdoor trajectory. Note that all metrics yield a similar number of feature matches over the trajectory, but the propagating method and distributed features methods do not compensate for lighting change during the transition, maintaining a relatively high exposure. The resulting overexposed images have low entropy and gradient magnitudes.

settings from the first images in the trajectory. Using the sequential method to generate targets for training is not ideal, as features that are indoors will not remain in the image frame for long, and few new features in the outdoor environment will be identified. The other methods, namely M_{feat} and M_{grid}, sacrifice a small reduction in the minimum number of feature matches across the transition (Table 4.3) for a higher number of matches over the trajectory (Table 4.2) and more information in the images overall (Figure 4.8). For continuous operation in dynamic environments, well-exposed images with more information are favoured. Consequently, the two metrics that we select for

training our networks are the number of features, M_{feat}, and the gridsearch inlier feature match method, M_{grid}.

We further investigated the performance of the M_{feat} and M_{grid} metrics relative to existing image metrics from the literature in Section 3. Table 4.4 provides a concise summary of our proposed metrics compared to the existing heuristic image metrics. We again recorded the highest median and minimum number of inlier matches over all trajectories. We see that our two metrics produced the most $\mathcal{M}$-trajectories with the highest number of median matches. Additionally, the M_{grid} method yielded the most $\mathcal{M}$-trajectories with the highest minimum inlier matches. The low number of top-scoring $\mathcal{M}$-trajectories for the M_{feat} metric in the bottom row of Table 4.4 was not entirely representative as most of the values were only slightly smaller than those of M_{grid}.

Table 4.4: Comparison of our selected metrics with existing metrics from the literature. For each metric, an $\mathcal{M}$-trajectory was generated. This table is a summary of the number of trajectories that had the highest average median and minimum number of inlier matches between images.

	Inlier Feature Matches							
	Entropy [4]	Gradient Info [3]	Gradient Sum [3]	Kim [40]	Shin [41]	Zhang [39]	Ours (M_{feat})	Ours (M_{grid})
# top scoring trajectories (*Median*)	3	3	8	0	2	6	10	**13**
# top scoring trajectories (*Minimum*)	8	6	11	1	6	9	2	**15**

4.2.5 Target Generation and Training Procedure

For the single-parameter case, the training label for image I_t is generated by obtaining the exposure value of the image I_{t+1} at the next pose that maximizes the selected image metric (M_{feat} or M_{grid}). The specifics of this procedure are slightly different in the dual-parameter case.

Single-Parameter Case

Using the single-parameter dataset, targets are generated for the current image at pose t by analyzing the images in the next pose, $t+1$. The network accepts an input sequence of previous images (including the current image), and using the proposed training targets, learns to regress the target exposure time for the next image that is to be captured.

Using our proposed metrics to generate training targets for the network ensures the next image captured is of a high quality for VO applications (i.e., features or feature matches are maximized).

Obtaining the target exposure time $E^*_{t,feat}$ using the M_{feat} metric is straightforward. At the t^{th} pose, each image $I^{(i)}_{t+1}$ for $i \in 1, ..., N$ at pose $t+1$ is processed using an OpenCV ORB feature extractor. We select the image $I^{(i^*)}_{t+1}$ where i^* is the index of the image at pose $t+1$ with the maximal M_{feat} value (i.e., number of features). Finally, exposure time is extracted from image $I^{(i^*)}_{t+1}$ and is used as the target for training inputs at pose t.

$$i^* = \underset{i}{\operatorname{argmax}}\, M_{feat}(I^{(i)}_{t+1}), \quad E^*_{t,feat} = E^{(i^*)}_{t+1}. \tag{4.9}$$

Obtaining the target exposure time $E^*_{t,grid}$ using the M_{grid} metric involves matching two sequential images. For the t^{th} pose, each image $I^{(i)}_t$ for $i \in 1, ..., N$ in the current pose t, is matched to each image $I^{(j)}_{t+1}$ for $j \in 1, ..., M$ in the next pose $t+1$ using an OpenCV ORB feature matcher. We select the image pair $(I^{(i^*)}_t, I^{(j^*)}_{t+1})$ that results in the maximal M_{grid} value (i.e., number of inlier feature matches between poses). Finally, exposure time is extracted from image $I^{(j^*)}_{t+1}$ and is used as the target for training inputs at time t. We define an overloaded argmax function that obtains the values of two arguments i and j such that the function M_{grid} is maximized:

$$i^*, j^* = \underset{i,j}{\operatorname{argmax}}\, M_{grid}(I^{(i)}_t, I^{(j)}_{t+1}), \quad E^*_{t,grid} = E^{(j^*)}_{t+1}. \tag{4.10}$$

Dual-Parameter Case

Exposure time and gain targets E^* and G^* are generated for the current image at the t^{th} pose by analyzing the images in a window of future poses $\{t+1, t+2, t+3, t+4\}$ and finding the image that maximizes the selected metric. The reason for using a windowed approach is because the dual-parameter dataset, captured using a moving vehicle, only contains two samples from the parameter space at each pose. To generate training labels, it is important that the parameter space is explored as much as possible so that the network learns the best camera settings. Using a windowed approach ensures that all four quadrants in the parameter space are explored. The downside to this method is that this look-ahead in time risks the scene changing. The number of features or inlier matches in future images may not represent the true number that would exist if we could sample various camera settings simultaneously from the same pose, but we consider our method a close approximation. Although the dual-parameter dataset does not have the same sampling resolution as the single-parameter dataset, the downsides of sparse sampling

are balanced by the inclusion of motion blur and the realistic data used for training.

Obtaining the target exposure time $E^*_{t,feat}$ and gain $G^*_{t,feat}$ using the M_{feat} method is again straightforward. For the t^{th} pose, each image $I^{(i)}_{t+a}$ for $i \in \{1,2\}$ and $a \in \{1,2,3,4\}$ is processed using an OpenCV ORB feature extractor. The number of features in each image is counted and we select the image $I^{(i^*)}_{t+a^*}$ where i^* is the index of the image in pose $t+a^*$ with the maximal M_{feat} value (i.e., number of features). Finally, the exposure time and gain values, $E^{(i^*)}_{t+a^*}$ and $G^{(i^*)}_{t+a^*}$, used to acquire image $I^{(i^*)}_{t+a^*}$ are obtained and used as the target for training inputs at pose t. Again, we make use of the overloaded argmax function:

$$i^*, a^* = \underset{i,a}{\operatorname{argmax}}\, M_{feat}(I^{(i)}_{t+a}), \quad E^*_{t,feat} = E^{(i^*)}_{t+a^*}, \quad G^*_{t,feat} = G^{(i^*)}_{t+a^*}. \tag{4.11}$$

Obtaining the target camera parameters $E^*_{t,grid}$ and gain $G^*_{t,grid}$ for the M_{grid} method involves performing multiple image matchings over a window of images. Both images $I^{(i)}_t$ for $i \in \{1,2\}$ in the current pose t are matched with both images $I^{(j)}_{t+1}$ for $j \in \{1,2\}$ in pose $t+1$. This sequential matching is repeated for four pose pairs $\{I_{t+a}, I_{t+a+1}\}$ for $a \in \{0,...,4\}$. We select the image pair $(I^{(i^*)}_{t+a^*}, I^{(j^*)}_{t+a^*+1})$ that results in the maximal M_{grid} value. Finally, the exposure time and gain values, $E^{(j^*)}_{t+a^*+1}$ and $G^{(j^*)}_{t+a^*+1}$, used to acquire image $I^{(j^*)}_{t+a^*+1}$ are obtained and used as the target parameter values for training inputs at pose t. We again make use of the overloaded argmax function for three arguments i, j, and a:

$$i^*, j^*, a^* = \underset{i,j,a}{\operatorname{argmax}}\, M_{grid}(I^{(i)}_{t+a}, I^{(j)}_{t+a+1}), \quad E^*_{t,grid} = E^{(j^*)}_{t+a^*+1}, \quad G^*_{t,grid} = G^{(j^*)}_{t+a^*+1}. \tag{4.12}$$

In addition to the M_{feat} and M_{grid} metrics, we also investigate the use of a combined metric, designated as M_{hybrid}, which consists of averaging the values of the target parameters generated by M_{feat} and M_{grid}. Various values of weighting $\alpha \in [0,1]$ can be applied to the hybrid method:

$$E^*_{t,hybrid} = \alpha E^*_{t,feat} + (1-\alpha)E^*_{t,grid}, \qquad G^*_{t,hybrid} = \alpha G^*_{t,feat} + (1-\alpha)G^*_{t,grid}. \tag{4.13}$$

To create training samples, we need to generate pairs of input image sequences and target parameter values. For the single-parameter input sequence, three sequential images $\{I_t, I_{t-1}, I_{t-2}\}$ are combined with their corresponding exposure times $\{E_t, E_{t-1}, E_{t-2}\}$ into a 12-channel input, shown in Figure 4.1. In the dual-parameter case, gains $\{G_t, G_{t-1}, G_{t-2}\}$ are added to create a 15-channel input, shown in Figure 4.2. Using either dataset, a large number of training samples can be generated at each pose by sampling from the collected

images and using these sampled images to construct an input sequence. Generating a large variety of training samples both maximizes data efficiency and provides the network with a diverse range of inputs. In the single-parameter case, a maximum of twenty-five training samples were generated for each pose. In the dual-parameter case, the maximum of eight training samples were generated. The dual-parameter dataset contains significantly more poses, making up for the smaller number of training samples per pose.

To create an input, image I_t is concatenated with its corresponding parameters E_t (and G_t) along the channel dimension, to create a 4-channel (or 5-channel) image. The extra input channels are created by assigning the same value to each pixel in a $1 \times 224 \times 224$ tensor, and concatenating it to the $3 \times 224 \times 224$ RGB image. This concatenation is repeated for each image in the input sequence, and these images are concatenated with each other sequentially along the channel dimension to result in a final $12 \times 224 \times 224$ (or $15 \times 224 \times 224$) dimensional input. The value for the parameter channel is determined by scaling the current value to a value between 0–255. To perform this scaling, an upper and lower limit for exposure and gain are required. We set the allowable range of exposure times as 75 μs–30 ms and the range of allowable gain values as 0-30 dB. The scaled parameter value can then be calculated as:

$$E_{t,\text{scaled}} = \frac{E_t - E_{\min}}{E_{\max} - E_{\min}} * 255, \quad G_{t,\text{scaled}} = \frac{G_t - G_{\min}}{G_{\max} - G_{\min}} * 255. \tag{4.14}$$

The network takes the above input and regresses a scaled version of the absolute value of exposure (and gain) clamped between 0–1. The targets are also scaled to be between 0–1 using Equation 4.14 and the scaled targets and network outputs are passed to the loss function. The reason for the parameter standardization is to ensure that the two parameters are weighted equally and the magnitude of the gradients are on the same scale. Through empirical testing, we determined that regressing the absolute value of the exposure (and gain) resulted in better convergence than estimating a scaling or additive term to the current parameter value. The loss function used to train our network is:

$$\mathcal{L} = \epsilon \frac{1}{N} \sum_{i=0}^{N} |\hat{G}_i - G_i^*| + (1 - \epsilon) \frac{1}{N} \sum_{i=0}^{N} |\hat{E}_i - E_i^*|, \tag{4.15}$$

for a batch of N training samples where we set $\epsilon = 0.5$. In the single-parameter case, the loss is simply the L_1 loss of the estimated exposure time. For training, we used the ADAM optimizer [44] with a batch size of 64 and learning rate of 1×10^{-4}. Training hyperparameters were determined empirically using a random-search hyperparameter tuning strategy on a held-out set of training data.

Chapter 5

Experimental Results

We determined the efficacy of the proposed networks through several experiments conducted in a real-world driving scenario. In Section 5.1, we first outline the experimental setup including the camera hardware. Next, the single-parameter network results are presented in Section 5.2.1 and the dual-parameter network results are presented in Section 5.2.2. Finally, we discuss our overall findings in Section 5.3.

5.1 Experiment Details

Our experiments consisted of measuring the number of inlier feature matches in the images acquired by our networks under a variety of lighting conditions. We compared the performance of our networks with both built-in AE+AG controllers and an approach from the literature, [41].

5.1.1 Camera and Hardware Considerations

Real-world experiments are affected by environmental structure and lighting and cannot be identically replicated over sequential runs. To ensure that a fair comparison was made between our proposed approach and other methods, we used two cameras that operated simultaneously with different camera parameter control algorithms. Further, the cameras acquired images at the same time to account for the fast speed of the vehicle during experimentation and the rapidly changing lighting conditions. Finally, images were captured from nearly the same perspective.

We used the camera setup described in Section 4.2.2, consisting of two FLIR Blackfly S U3-31S4C machine vision cameras. These cameras have a native resolution of 2048 $\times$ 1536 (3.1 MP) and can capture at a maximum frame rate of 55 Hz. The cameras

have a global shutter and are fully controllable with the FLIR PySpin API. The range of possible exposure times is 11 μs–30 s while the range of possible gain values is 0–47 dB. The cameras can be powered and controlled simultaneously via USB 3.0 connections and also contain a GPIO port for hardware triggering. The lenses used were Fujinon HF6XA-5M 2/3" C-mount lenses with a 6.23 mm focal length. They have adjustable focus and aperture knobs that can be locked in place. For the experiments, the cameras were fixed to a 3D-printed mount in a fronto-parallel configuration with a baseline of 3.92 cm, as shown in Figure 4.5a.

It was imperative that the cameras acquired near-identical images when their parameters were the same. Although exposure time and gain can be set to the same values via the software API, the lens apertures needed to be manually adjusted such that the openings were equal in diameter. To ensure that the lens apertures were accurately set, a simple aperture calibration was performed. The cameras were positioned in front of a well-lit, blank surface and set to continuously acquire images at the same exposure time, gain, and frame rate. The mean intensity values of the images were then compared. The lens aperture on the first camera was arbitrarily set to a relatively small value. We used a small value to ensure that the camera had a large depth of field, and that objects at various distances were in focus, as discussed in Section 2.1.2. The second aperture was adjusted such that the mean image intensities produced by each camera were identical and then the apertures were securely locked in place. The lens aperture calibration ensured that the cameras captured near identical images (when all other camera settings were the same), allowing for fair comparisons of the approaches.

Equally important to the aperture calibration was that the two cameras acquired images simultaneously. Drastic lighting changes can occur over a timespan of only a few frames so it was critical that images were captured at the same time. To collect images simultaneously, we made use of hardware triggering. Hardware triggering consists of the first camera sending a signal to the second camera over a GPIO cable when it begins the image acquisition process. This signal triggers the second camera to capture an image at the same time instant. Hardware triggering was achieved through the use of the GPIO connections available on the cameras, shown in Figure 5.1.

Another consideration made regarding the cameras was the method by which the cameras continuously acquired images. There are two operational modes that can be utilized for continuous image capture. The first mode ensures that a constant frame rate is maintained. The drawback of using the fixed-frame rate mode is that adjustments made to camera parameters are not applied to the next frame. Changes are likely to be applied 3–4 frames later than the frame they are intended to be applied to. The second

Figure 5.1: The back of our camera rig displaying the USB3.0 and GPIO connections required for hardware triggering.

operational mode applies parameter changes to the very next frame. The drawback of using the next-frame mode is that the camera frame rate is variable and dependent on the exposure time of the current frame. Longer exposure times result in a lower frame rate. We are interested in adjusting camera parameters based on the current image and applying these changes to the next frame. The input to our network consists of the three most recent images and corresponding parameter values and the network processes this input and outputs the exposure (and gain) of the next frame. Consequently, introducing a 3–4 frame delay may cause the network to behave in an unexpected manner. Therefore, we selected the second (i.e., next-frame) mode of operation for these experiments and ensured that the changes to parameter values were applied to the next frame. Although using this mode of operation required a variable camera frame rate, it did not significantly impact our results as we found the minimum frame rate to be $\sim$10 Hz, obtained at a maximum allowable exposure time of 30 ms, which occurred infrequently during experimentation.

Finally, both cameras needed to be calibrated to determine their intrinsic camera parameters. Intrinsic calibration was accomplished using OpenCV and a checkerboard calibration target. The calibration target was fixed to a flat surface and multiple images were acquired using both cameras from a variety of perspectives. The checkerboard corner locations were then extracted and used to determine the intrinsic camera parameters of each camera.

5.1.2 Experimental Setup

The camera rig was mounted to the inside of a vehicle in a forward-facing position and driven on busy urban roads in London, Ontario. For dynamic lighting experiments, we designed a closed-route containing two tunnel passages. The tunnels consisted of railway overpasses above urban streets and are roughly eighty metres in length. Additionally, the route contained straight sections of road where lighting was relatively constant. The selected route allowed for testing in both dynamic and static lighting conditions. A map of the route is shown in Figure 5.2a.

(a) Arial view (*Credit: Google Maps*). (b) Test route tunnel.

Figure 5.2: (a) A top-down view of the test route used for experiments. The route contains two tunnels highlighted by dotted lines located on the east and west sections of the route. (b) An example of one of the ∼80 metre long tunnels contained in the test route.

Experiments consisted of collecting full traversals of the above route, starting and ending at roughly the same pose. In each experiment, one camera acquired images using our proposed single- or dual-parameter network while the other camera captured images using the built-in AE (+AG in the dual-parameter case) controller or using the method from Shin et al. [41]. To determine the performance of our networks, we processed the images using two VO front end feature matching algorithms and compared them to AE (+AG) and the Shin method [41]. Namely, we used both OpenCV feature matching and `libviso2` as our two VO front ends. We recorded the number of inlier feature matches measured over the trajectory as a proxy for VO performance, as more inlier matches typically results in better VO performance. We are interested in both increasing the *median number of inlier feature matches* and increasing the *minimum number of inlier feature matches* over the entire trajectory. The minimum number of inlier feature matches is a statistic referring to the smallest amount of inlier matches between sequential

image frames measured over a trajectory. The minimum number of inlier matches is an important performance metric as VO can fail in cases where the minimum number of inlier matches is too low. The minimum number of inlier matches in a trajectory is typically the result of sequential frames that are similarly oversaturated or undersaturated and do not contain any identifiable features. The minimum number of inliers can also be a result of sequential frames that are significantly different from each other. Large differences between sequential images can be caused by combinations of fast motion (especially rotation), blur, noise, and lighting changes.

The cameras were mounted in the vehicle behind the windshield on the front passenger side. The cameras and networks were operated using a Lenovo Legion Y730 laptop with an Intel i7-8750H CPU at 2.20 GHz and an Nvidia GeForce GTX 1050 Ti GPU. To ensure sufficient power for an extended period of experimentation, the laptop was powered by a 12 V, 18 A·h sealed lead-acid rechargeable battery through a 500 W power inverter.

5.2 Real-World Experiments

As a proof-of-concept for a learning-based approach, we conducted our initial experiments using the single-parameter network. We then carried out experiments with the dual-parameter network to determine the benefits of controlling both parameters.

5.2.1 Single-Parameter Case

To determine the feasibility of a network-based approach, we initially focused on learning adjustments to camera exposure values and ignored gain. The single-parameter network was trained using the single-parameter dataset described in Section 4.2.2 and with targets generated using the M_{feat} metric. We initially conducted a simple driving experiment in relatively static lighting conditions on a rural highway road. We then tested our approach under dynamic conditions in the route described in Section 5.1.2.

Static Lighting Experiments

In addition to predictively adjusting camera parameters to compensate for dynamic environmental lighting, it was equally important that the network did not output erratic exposure time estimates or oscillate when lighting was relatively stable. To test the network behaviour in static lighting conditions, several trajectories were captured in relatively static, open-road, rural environments.

The results of these experiments are displayed in Table 5.1. In this table, our proposed method is compared with two variants of built-in AE: one with fixed gain, similar to our method, and one with auto-gain control enabled (AE+AG). We also compared our proposed network with the Shin method. The Shin method is relatively brittle and is highly dependent on the selection of appropriate tuning parameters. The most important tuning parameter is the convergence criterion which determines whether the current parameter settings are optimal based on a comparison between the metric value of the current image and metric value of the most recent optimal image. The convergence criterion dictates the maximum allowable difference between the metric values in which the algorithm is still considered to have converged to the optimal parameter settings. The optimization process restarts if the difference in metric values becomes greater than the threshold. Since our camera and lens systems were different from those used in the Shin et al. experiments, we selected three different values for the convergence criterion for testing: low-tolerance, mid-tolerance, and high-tolerance.

Table 5.1: Single-parameter network performance in static lighting conditions. A comparison of average inlier feature match statistics over the recorded trajectories.

Method	Inlier Matches *Median*		Inlier Matches *Minimum*	
	`ORB`	`libviso2`	`ORB`	`libviso2`
AE	277	3165	43	1857
Ours	**317**	**3810**	**47**	**2545**
AE+AG	**504**	**6709**	53	**4538**
Ours	475	6464	**56**	4382
Shin method [41] (*High-Tolerance*)	526	**5655**	144	**4269**
Ours	**542**	5206	**162**	3459
Shin method [41] (*Mid-Tolerance*)	545	**5398**	**294**	3611
Ours	**555**	5369	251	**3786**
Shin method [41] (*Low-Tolerance*)	**410**	**4939**	45	12
Ours	396	4040	**48**	**228**

From Table 5.1, we can see that our proposed network outperforms AE control with fixed gain and has comparable results to all three variants of the Shin method. Our approach, however, fails to outperform built-in AE+AG, which is expected, since we are limited to controlling exposure time only.

(a) Single-parameter network (*left*) vs AE (*right*).

(b) Single-parameter network (*left*) vs Shin method (mid tolerance) (*right*).

Figure 5.3: Examples of individual frames from ORB matching experiments showing our proposed method compared with (a) AE, and (b) the Shin method with medium tolerance for the thresholding hyperparameter. Note that our proposed network favours higher exposures, resulting in more feature matches.

We highlight the performance of our method compared with both fixed-gain AE control and the mid-tolerance version of the Shin method. Figure 5.3a shows example frames from an experiment comparing our method with fixed-gain AE. The network, trained using targets generated by the M_{feat} metric, selected exposure times that are generally higher than the exposure times output by AE, resulting in images with a high number of features. In Figure 5.3a, we see that the network selected exposure times that brightened the road and surrounding scenery while overexposing the sky. The brighter scene, consequently, resulted in more features and a higher number of inlier feature matches over the trajectory. The increased number of inlier feature matches is due to an increase of brightness in the regions containing useful features for VO. The M_{feat}

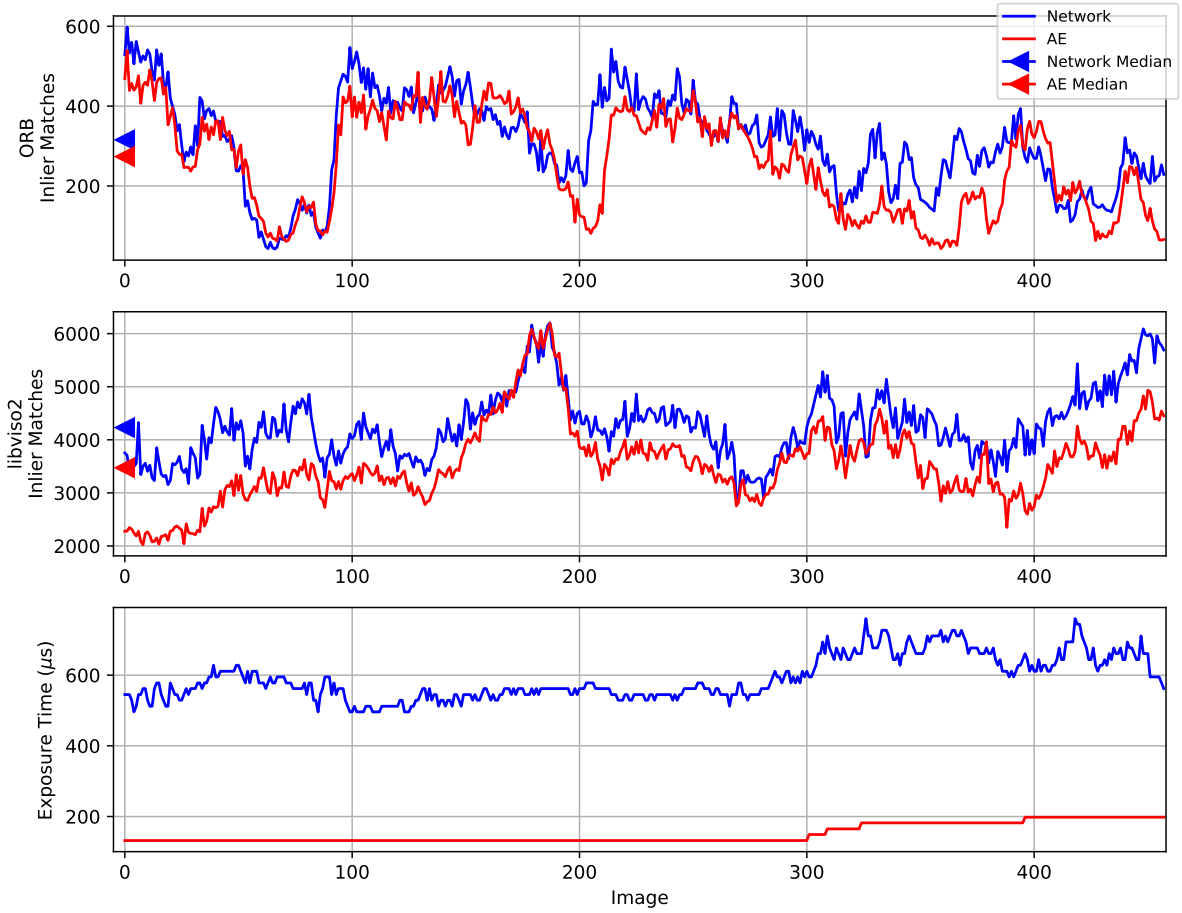


Figure 5.4: ORB and `libviso2` inlier feature matches along with exposure times for our method and built-in AE across an entire static lighting trajectory.

method of generating training targets yields higher exposure times, as images with an overexposed sky and a brighter surrounding environment typically have more detectable features than images that properly expose the entire scene, including the sky. The same performance can be seen in Figure 5.3b when compared to the Shin method. Again, our network selected exposure times that yield brighter images, often resulting in more inlier features matches.

Figure 5.4 shows plots of the inlier ORB and `libviso2` feature matches along with the corresponding exposure times for our method compared with AE with fixed gain. The average number of inlier feature matches was improved for both feature matching algorithms as a result of the brighter images. Note, however, that the single-parameter network did not output a smooth exposure time profile over the entire trajectory, which suggests that there may be a trade-off between the speed of lighting compensation and the stability of the network outputs.

The downsides of the single-parameter network approach are due to the training dataset and the limitation of single-parameter control. Since all of the training images were captured with a stationary camera, images acquired with higher exposure times were just as sharp as images captured with lower exposure times, but were brighter and contained more features. Consequently, the training targets were typically identified as images that had a relatively high exposure time. Unfortunately, the training images did not contain motion blur that would have been included had the higher exposure images been captured while the camera was moving, and was likely the reason why the AE+AG controller outperformed our approach. Additionally, we noticed some oscillatory behaviour in the outputs of our network in a few instances, resulting in lower inlier feature match statistics.

Dynamic Lighting Experiments

Once it was determined that the single-parameter network performed relatively well under static lighting conditions, our next task was to determine its performance in dynamic conditions. We made use of the route outlined in Section 5.1.2 to conduct experiments in dynamic lighting conditions.

In our analysis of the experimental results, we focused on sections of the route containing tunnel transitions as these introduced the greatest challenges for feature matching. For these experiments, we fixed the gain value of AE to be the same as our network (10 dB) to ensure a fair comparison in performance. We also selected the mid-threshold Shin method variant as the representative case for the Shin method as performance was similar in all cases in the static lighting experiments. From this point, all experiments involving the Shin method were conducted using the mid-tolerance criterion. Table 5.2 summarizes the average performance of the single-parameter network across ten tunnel transitions.

It is clear to see that our method resulted in a higher number of median inlier feature matches on average compared with AE using both ORB and `libviso2`. The downside of our approach, however, is that it resulted in a significantly lower average minimum number of inlier matches for both feature matching algorithms, which is a more important metric for image quality in dynamic lighting conditions. When compared with the Shin method, our network yielded higher average median and minimum inlier feature matches using `libviso2`, but lower averages for both performance metrics using ORB. These results were not ideal, as the Shin method is quite brittle and not intended to be used in dynamic lighting conditions. To determine the cause of the low minimum inlier feature match averages, we examined the network outputs over the course of a trajectory.

Table 5.2: Single-parameter network performance in dynamic lighting conditions. A comparison of average inlier feature match statistics across ten tunnel transition trajectories.

Method	Inlier Matches *Median*		Inlier Matches *Minimum*	
	`ORB`	`libviso2`	`ORB`	`libviso2`
AE	324	5514	**68**	**509**
Ours	**342**	**6467**	21	293
Shin method [41] (*Mid-Tolerance*)	**341**	3739	**78**	205
Ours	284	**4420**	32	**282**

Figure 5.5 shows a plot of the inlier feature matches across a tunnel transition. Our method obtained a higher average inlier feature match count but performed poorly during the fast transition into the tunnel. It is apparent from the exposure time plot that our method correctly predicted the approach of, and exit from the tunnel, and compensated for the changes in lighting by adjusting the camera exposure time before the AE algorithm. Additionally, the network began to lower the exposure time almost immediately after entering the tunnel and greatly reduced it closer to the exit of the tunnel. The predictive exposure time adjustments resulted in a higher number of inlier feature matches during the transition out of the tunnel (between frames 120–140). However, the predictive adjustments yielded poorly matched images at the entrance of the tunnel. The experimental results suggest that increases in exposure time while the camera is in bright daylight can reduce feature matching performance, while decreasing exposure before exiting from a darker environment can benefit feature matching performance.

Figures 5.6a and 5.6b each show a frame during the transition into and out of the tunnel respectively and demonstrate the quality of images obtained using the network. With both the network and AE, the number of inlier feature matches was drastically reduced at the entrance of the tunnel, however, the faster adjustment made by the network actually resulted in worse performance. There are several reasons why this faster response may have resulted in worse matching performance. First, increasing exposure time quickly, rather than gradually, can cause sequential frames to be quite different from one another. Reductions in matching due to differences in exposure during the transition into the tunnel occurred with both approaches, but to a greater degree with the quicker adjustments made by the network. Second, the cameras were located within the vehicle, behind the windshield. As the vehicle transitioned into the tunnel, we noticed that a significant amount of glare appeared on the windshield. Additionally,

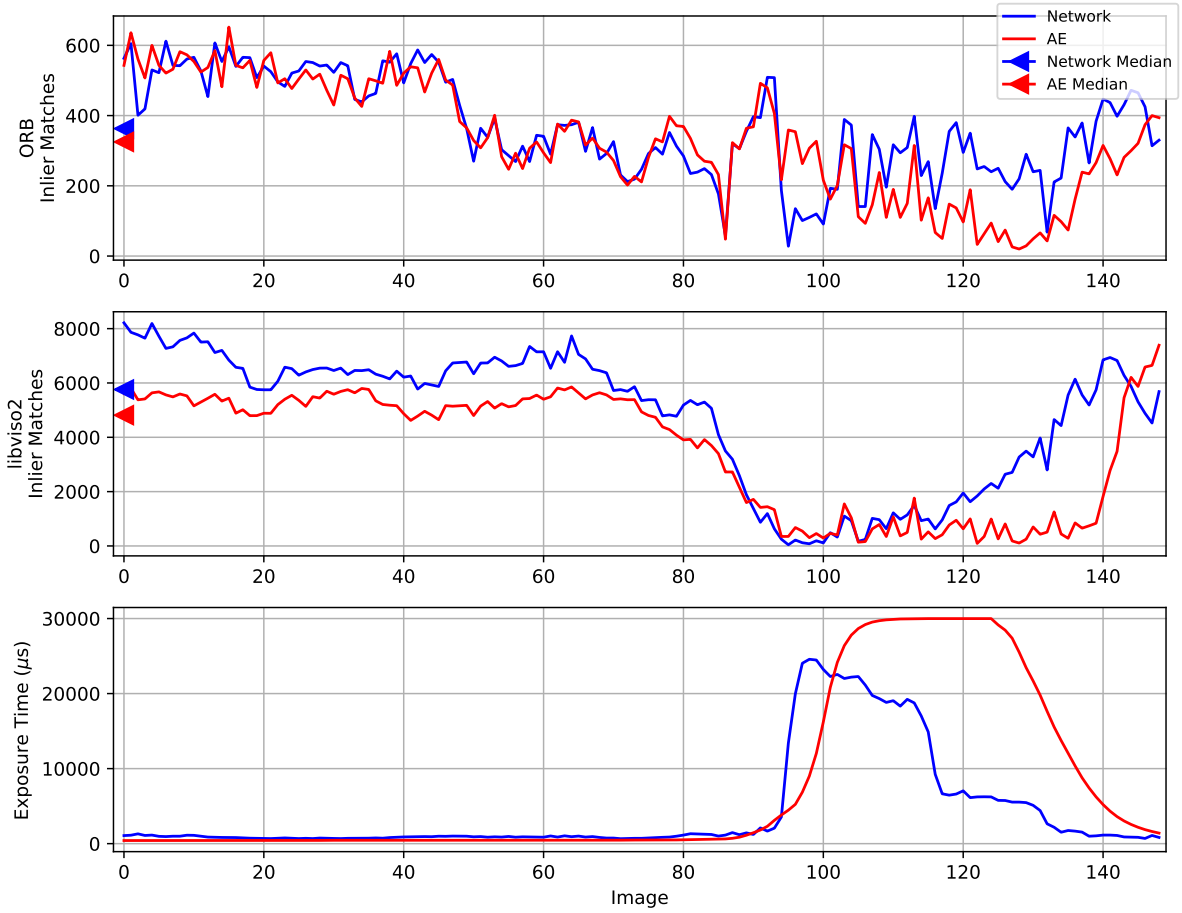
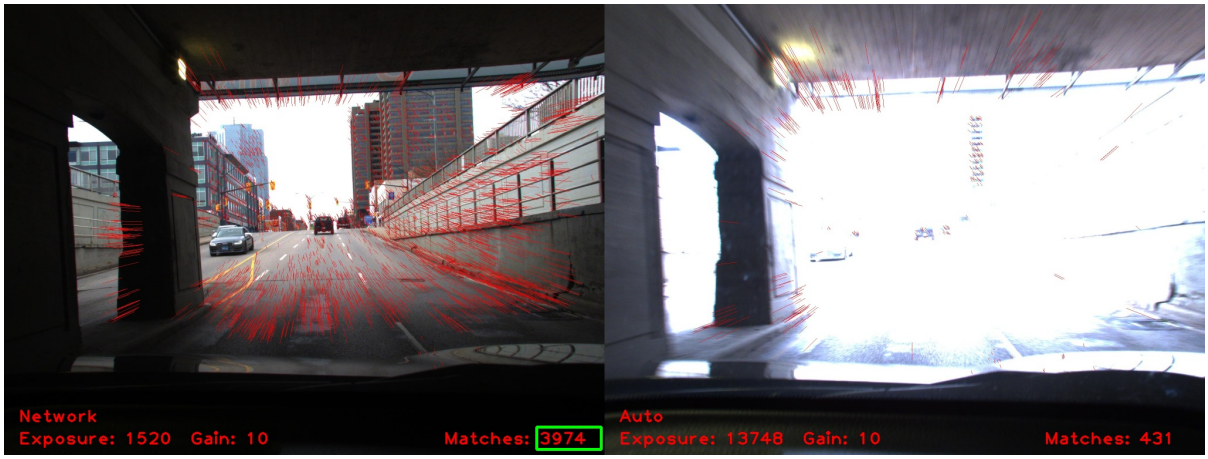


Figure 5.5: ORB inlier feature matches, `libviso2` inlier feature matches, and exposure times over a tunnel transition for our method and built-in AE with fixed-gain.

details within the car were reflected onto the inside of the windshield. An example of this glare can be seen in Figure 5.6a, on the left side of the left (network) image. Features in the environment outside of the vehicle were obscured by the glare, and spurious matches with reflected details from inside the vehicle further contributed to a reduction in inlier feature matches. We noticed that the quantity of glare was significantly increased with the higher exposure time of the network, perhaps attributing to the lower minimum inlier match counts. Conversely, the reduction in exposure time was significantly more gradual when exiting the tunnel, especially for the network, resulting in a higher number of inlier feature matches. Most notably, however, is that both the single-parameter network and the built-in AE (with fixed-gain) were limited in that they could only adjust the camera exposure time, which resulted in significant blur during the tunnel transitions, evident in Figures 5.6a (left) and 5.6b (right). Further, the network was not trained with images containing blur, so it did not learn the relation between increased exposure time and

(a) Tunnel entrance. Network (*left*) vs AE (*right*).



(b) Tunnel exit. Network (*left*) vs AE (*right*).

Figure 5.6: Examples of individual frames from a `libviso2` matching experiment comparing our proposed method to AE with fixed gain over a trajectory through a tunnel. (a) Shows that our method performs relatively poorly at the entrance of the tunnel while (b) demonstrates greatly improved performance at the exit of the tunnel.

image blur. The lack of gain control necessitated larger increases to exposure time, and resulted in blurry images with fewer matchable features, especially within the tunnel, and is likely why AE+AG outperformed our single-parameter network. The results obtained using the single-parameter network reinforce our motivation for using a learning-based approach. The network learned to compensate for changes in environmental lighting and behaved in an expected manner. These experimental results also indicate that the main limitation of using a learning-based approach is the method of collecting training data.

5.2.2 Dual-Parameter Case

In Section 5.2.1, we demonstrated that a data-driven approach to adjusting camera parameters can, in some cases, improve the quality of images for the VO front end. The shortcomings of the single-parameter experiments were mostly due to the training dataset and the lack of gain control. These shortcomings were most evident in the static experiments, where AE+AG outperformed the single-parameter network. Consequently, we trained a network using the dual-parameter dataset described in Section 4.2.2 to learn to adjust both exposure time and gain in a predictive fashion. We also investigated the effects and differences in training with variants of two metrics of interest (M_{feat} and M_{grid}). For a full comparison with the single-parameter controller, we conducted the same static and dynamic lighting experiments, recording the average median and minimum ORB and `libviso2` inlier feature match statistics.

Comparison of Network Training Methods

The dataset collection method used for the dual-parameter network, as previously discussed in Section 4.2.2, sampled the parameter space in a very sparse manner. Using two cameras allowed us to sample only two images (camera parameter settings) at each pose. For generating training targets, the number of available parameter samples was increased to eight using the windowed approach. Due to the sampling sparsity and the differences between the proposed image metrics, the training targets generated using our image metrics differed substantially from each other. We sought to determine which target generation method resulted in the best network performance. To determine which metric was optimal, we compared the performance of four networks trained using different target generation methods, 1) M_{feat}, 2) windowed M_{feat}, 3) windowed M_{grid}, and 4) windowed M_{hybrid}, over static and dynamic lighting conditions. We then measured both the median and minimum inlier feature counts for ORB and `libviso2` feature matches. Figure 5.7 shows a comparison of network performance in static lighting conditions while Figure 5.8 displays a comparison of the performance in dynamic conditions.

From Figures 5.7 & 5.8, is it clear that the windowed hybrid method of generating training targets resulted in a network that consistently outperformed all others in both static and dynamic lighting conditions. The M_{feat} method resulted in inconsistent median inliers matches and consistently low minimum inlier matches over the recorded trajectories. The poor performance of the M_{feat} method was due to the oscillatory behaviour introduced by the sparsity of the training samples generated at each pose and can be seen in Figure 5.9a. Obtaining targets from a window of poses greatly improved

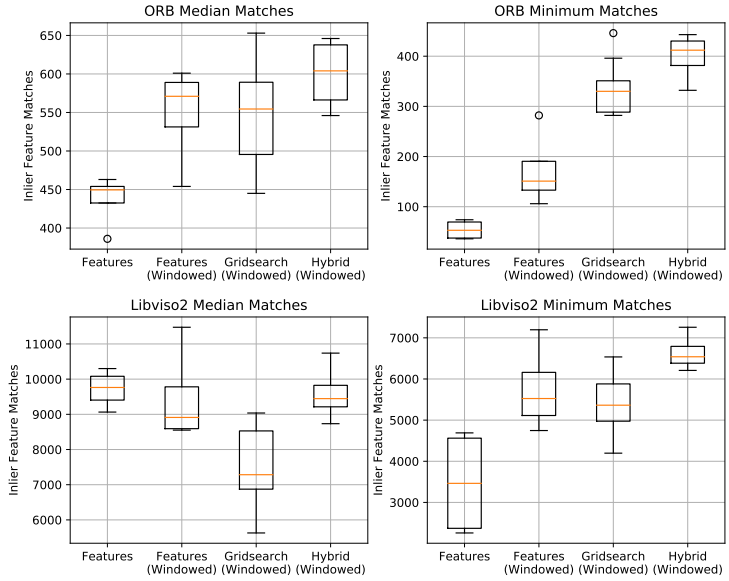


Figure 5.7: Median and minimum ORB and `libviso2` inlier feature match statistics for static lighting trajectories. These statistics were generated over at least four trajectories for each method. The hybrid training method was the most performant.

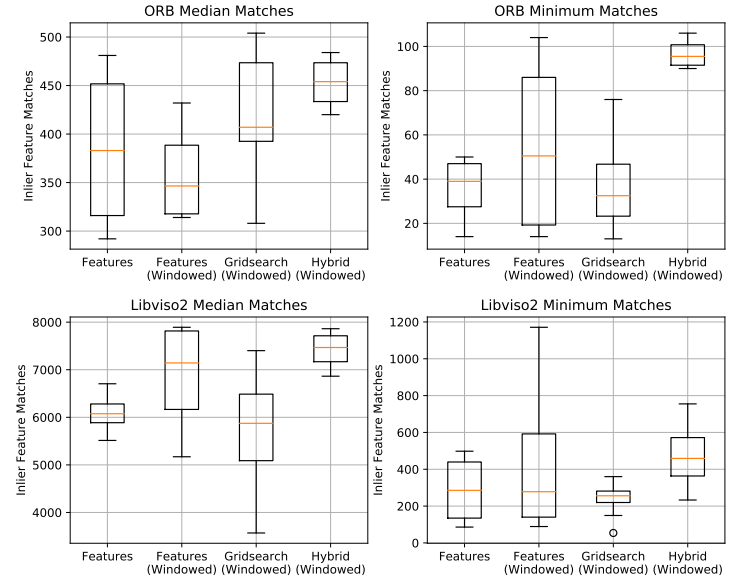
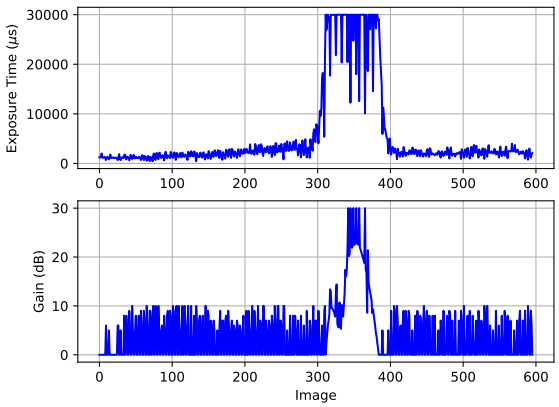
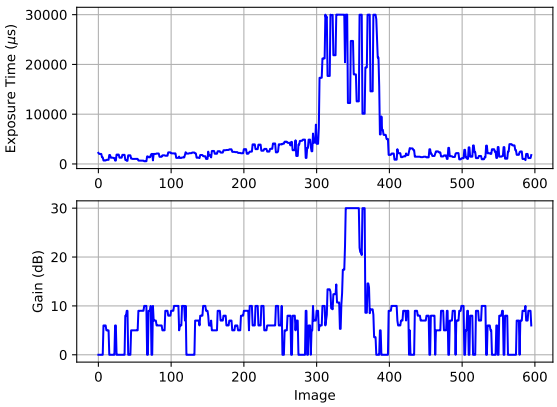


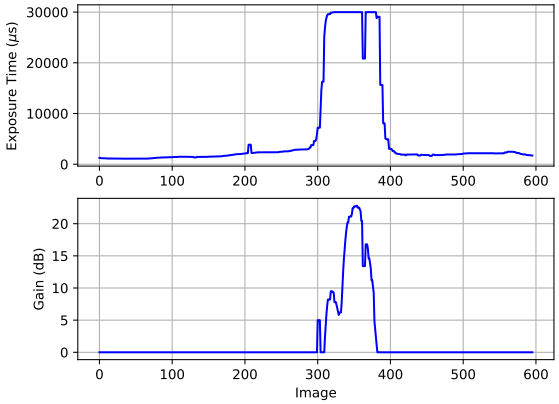
Figure 5.8: Median and minimum ORB and `libviso2` inlier feature match statistics for dynamic lighting (tunnel) trajectories. These statistics were generated over at least four trajectories for each method. Again, the hybrid training method was the most performant.



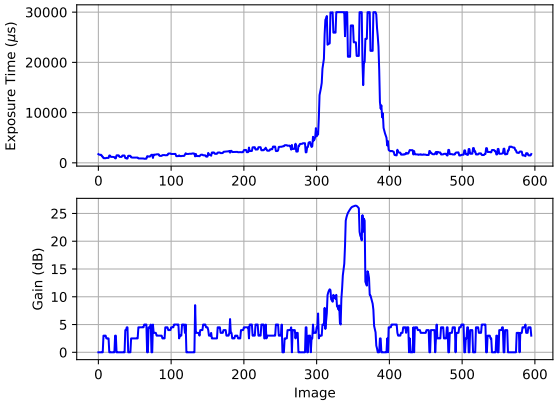
(a) Features method.



(b) Features (windowed) method.



(c) Gridsearch (windowed) method.



(d) Hybrid (windowed) method.

Figure 5.9: Examples of target profiles for the proposed target generation methods across a tunnel transition. The plots indicate the value of the targets over time generated using the respective target generation method.

the performance of the M_{feat} method through a reduction of oscillations in the training targets as shown in Figure 5.9b. The windowed gridsearch method, alternatively, had a much more stable training signal, as shown in Figure 5.9c, but this stability led to slower parameter adjustment speeds during transitions, again reinforcing the notion of a trade-off between reaction speed and output stability. The effects of the slower response can be seen in the plots of the minimum inlier feature matches for the dynamic experiments (Figure 5.8) where the windowed gridsearch method scored the lowest. The windowed hybrid method, which averages the parameters determined by the windowed M_{feat} and M_{grid} methods, yielded the best results in almost every case. The hybrid method com-

bined the relative stability of the windowed gridsearch targets with the quick response and higher feature count of the windowed features targets during transitions, shown in Figure 5.9d. For these reasons, we utilized the dual-parameter network trained with the windowed hybrid target generation method for all experiments in this section.

Static Lighting Experiments

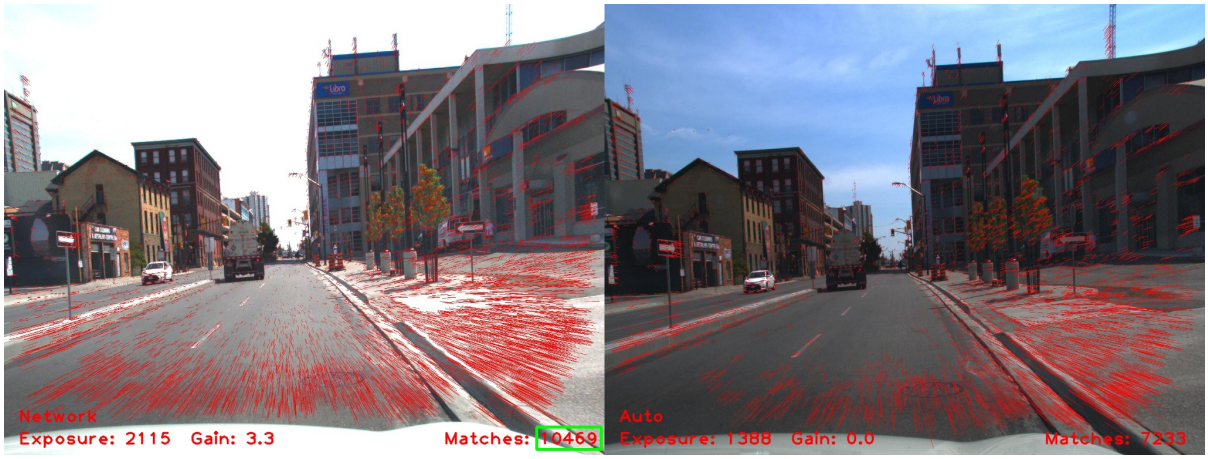
It was important that the dual-parameter network output consistent exposure time and gain values in environments where lighting was relatively static. The oscillation in the exposure value outputs of the single-parameter network contributed to a reduction in the number of inlier feature matches. Using the windowed hybrid training method to train the dual-parameter network effectively reduced oscillation of the network outputs. The hybrid training method also maintained a high number of features in captured images and made quick adjustments to exposure time and gain during transitions. We compared the performance of our network trained with the windowed hybrid targets with built-in AE+AG and the Shin method [41].

Table 5.3: Dual-parameter network performance in static lighting conditions. A comparison of average inlier feature match statistics across multiple (≥4) trajectories.

Method	Inlier Matches *Median*		Inlier Matches *Minimum*	
	`ORB`	`libviso2`	`ORB`	`libviso2`
AE+AG	593	8301	366	5840
Ours	**600**	**9591**	**401**	**6636**
Shin method [41] (*Mid-Tolerance*)	414	5295	183	3555
Ours	**439**	**5666**	**200**	**4032**

We see from Table 5.3 that our proposed dual-parameter network trained using the hybrid method consistently yielded higher median and minimum inlier feature matches for both ORB and `libviso2`. The higher number of median inlier matches can be attributed to brighter images resulting from the windowed feature component of the training targets. The higher number of minimum inlier matches can be attributed to the stability introduced by the windowed gridsearch component of the training targets. Overall performance improvements can also be attributed to the addition of gain control. Examples from the static experiments are displayed in Figure 5.10. Again, we can see that our network selected parameter values that yielded brighter images, with a slightly overexposed sky but containing more detail in the scene. The brighter images, consequently,

contained a higher number of identifiable features in each image and culminated in more inlier feature matches on average over the trajectory. An example of the inlier matches, exposure times, and gains over a static lighting trajectory is presented in Figure 5.11, where we can see that our network favoured a higher relative gain value while maintaining a lower relative exposure time across most of the trajectory. The combination of lower exposure time and higher gain led to brighter images with less blur and more inlier feature matches compared with those generated by AE+AG. We conclude that our network yields better images for feature matching in environments where lighting is relatively static.



(a) Dual-parameter network (*left*) vs AE+AG (*right*).

(b) Dual-parameter network (*left*) vs Shin method (mid tolerance) (*right*).

Figure 5.10: Examples of individual frames from `libviso2` matching experiments in static lighting conditions showing our proposed method compared with (a) AE+AG, and (b) the Shin method with medium tolerance for the thresholding parameter. Note that our proposed network, in general, favoured higher gain values, resulting in brighter images with more feature matches.

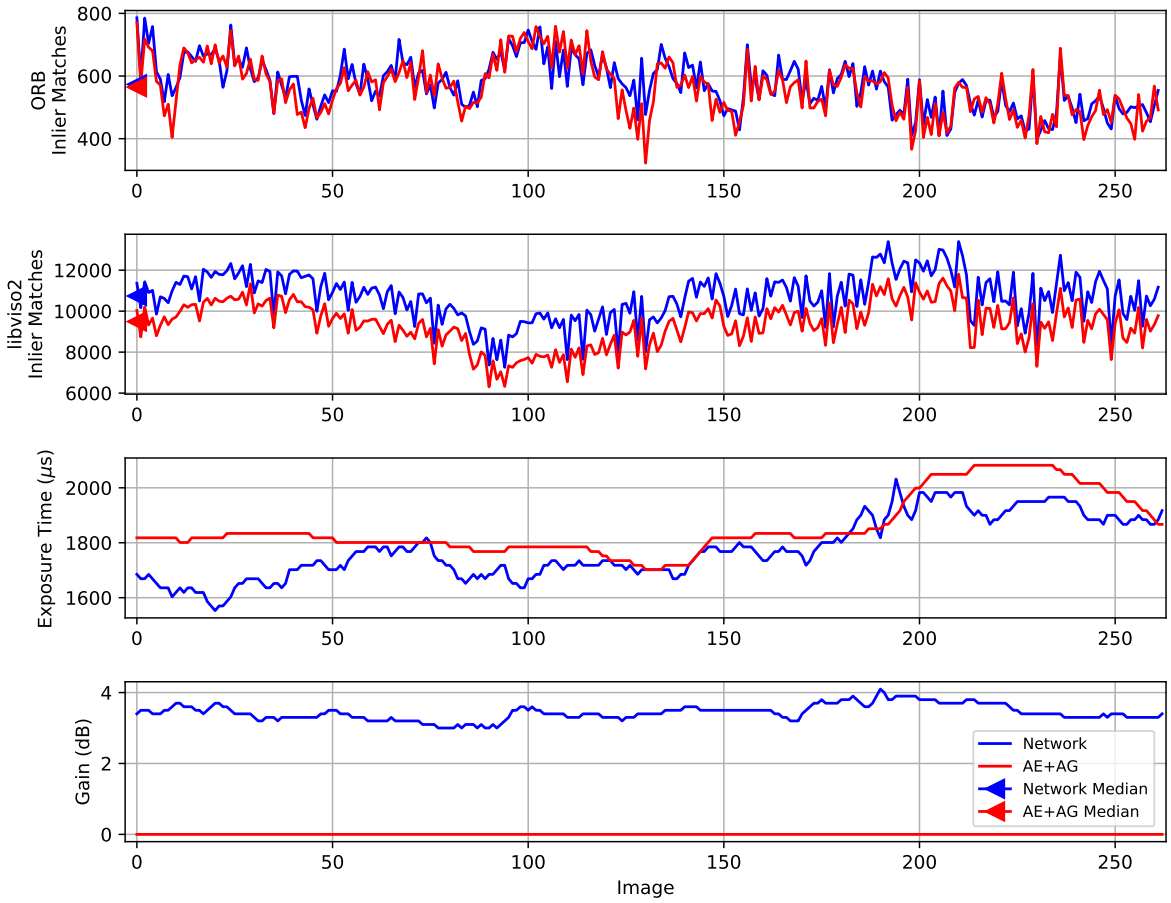


Figure 5.11: ORB and `libviso2` inlier feature matches along with exposure time and gain for our method and AE+AG across an entire static lighting trajectory.

Dynamic Lighting Experiments

Our single-parameter network did not outperform built-in AE or the Shin method [41] in the tunnel transition experiments. The limitations of the single-parameter approach were due to adjusting only one camera parameter and the lack of any motion blur contained in the training dataset. We removed these limitations using our dual-parameter network configuration and training regime using our dual-parameter dataset. The average results of eight tunnel transition experiments are summarized in Table 5.4.

From Table 5.4, we can clearly see that our network outperformed both built-in AE+AG and the Shin method in dynamic lighting conditions. In fact, the Shin method failed during a transition out of the tunnel, resulting in very low median inlier match values and in one case, zero minimum inlier match values. We suspect the optimization method used to search the parameter space reached a local minimum in a poor region and

Table 5.4: Dual-parameter network performance in dynamic lighting conditions. A comparison of average inlier feature match statistics across multiple tunnel transition trajectories.

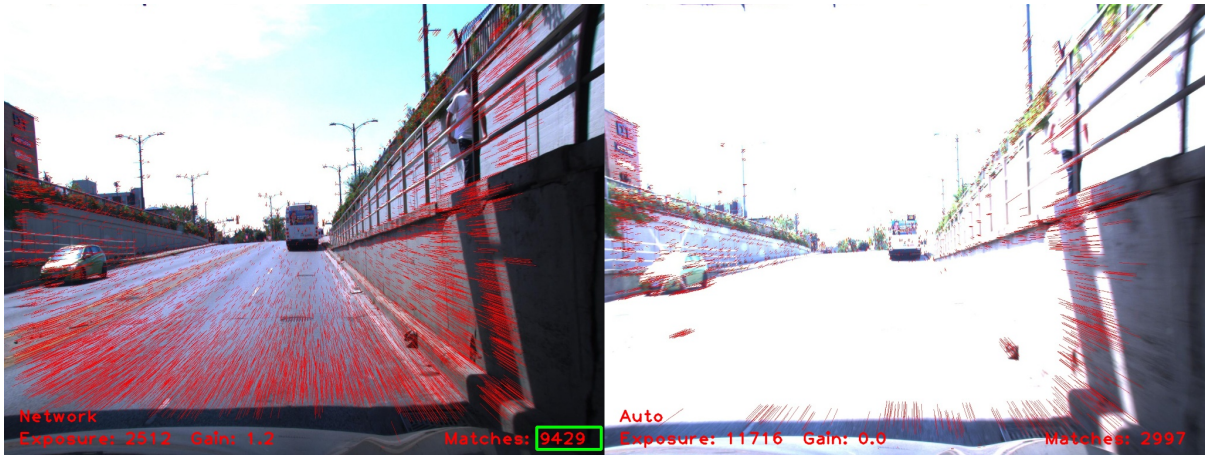
Method	Inlier Matches *Median*		Inlier Matches *Minimum*	
	`ORB`	`libviso2`	`ORB`	`libviso2`
AE+AG	433	5288	65	377
Ours	**434**	**6582**	**97**	**473**
Shin method [41] (*Mid-Tolerance*)	131	614	0	0
Ours	**202**	**3530**	**37**	**127**

could not recover. The authors of Shin et al. [41] informed us[1] that their method is not intended for use in dynamic lighting environments and can fail in such cases if not tuned correctly. We found the Shin method particularly difficult to correctly tune, which limits the practicality and effectiveness of their approach. The median number of inlier feature matches using our method was higher in all cases. More importantly, however, is that the minimum number of inlier features matches for ORB and `libviso2` was significantly higher in all cases as well. These results indicate that our method yields images that are well suited for the VO front end compared with built-in AE+AG and the Shin method in dynamic lighting conditions. We show examples of frames from these experiments in Figure 5.12.

Figure 5.12 demonstrates that our method produced images that are relatively well-exposed during fast transitions. In the dynamic lighting experiments, the network predicted that the camera was entering and exiting a tunnel and compensated faster than AE+AG. Additionally, our network was able to maintain useful images during the entire transition, unlike the Shin method [41].

We present an example of the inlier matches, exposure time, and gain values over time during a tunnel transition in Figure 5.13. We see that the network outputted similar exposure times but slightly higher gain values compared with AE+AG before and after the transition. The increased gain values resulted in brighter images containing a slightly overexposed sky, similar to those described in the static lighting experiments. These brighter images contained a higher number of features that were able to be identified and matched. We see that the gain was preemptively increased before entering the tunnel to account for the darkness inside and reduced earlier than the AE+AG controller to

[1] We discussed the applicability of the Shin method to dynamic lighting environments with the authors in a private communication.



(a) Dual-parameter network (*left*) vs AE+AG (*right*) exiting a road tunnel.

(b) Dual-parameter network (*left*) vs AE+AG (*right*) entering a road tunnel.

(c) Dual-parameter network (*left*) vs Shin method (mid tolerance) (*right*).

Figure 5.12: Examples of individual frames from `libviso2` matching experiments showing our proposed method compared with AE+AG during (a) a tunnel exit and (b) a tunnel entrance. We also show a comparison with (c) the Shin method with medium tolerance for the thresholding hyperparameter. The Shin method failed to recover during the transition out of the tunnel, resulting in severely overexposed images.

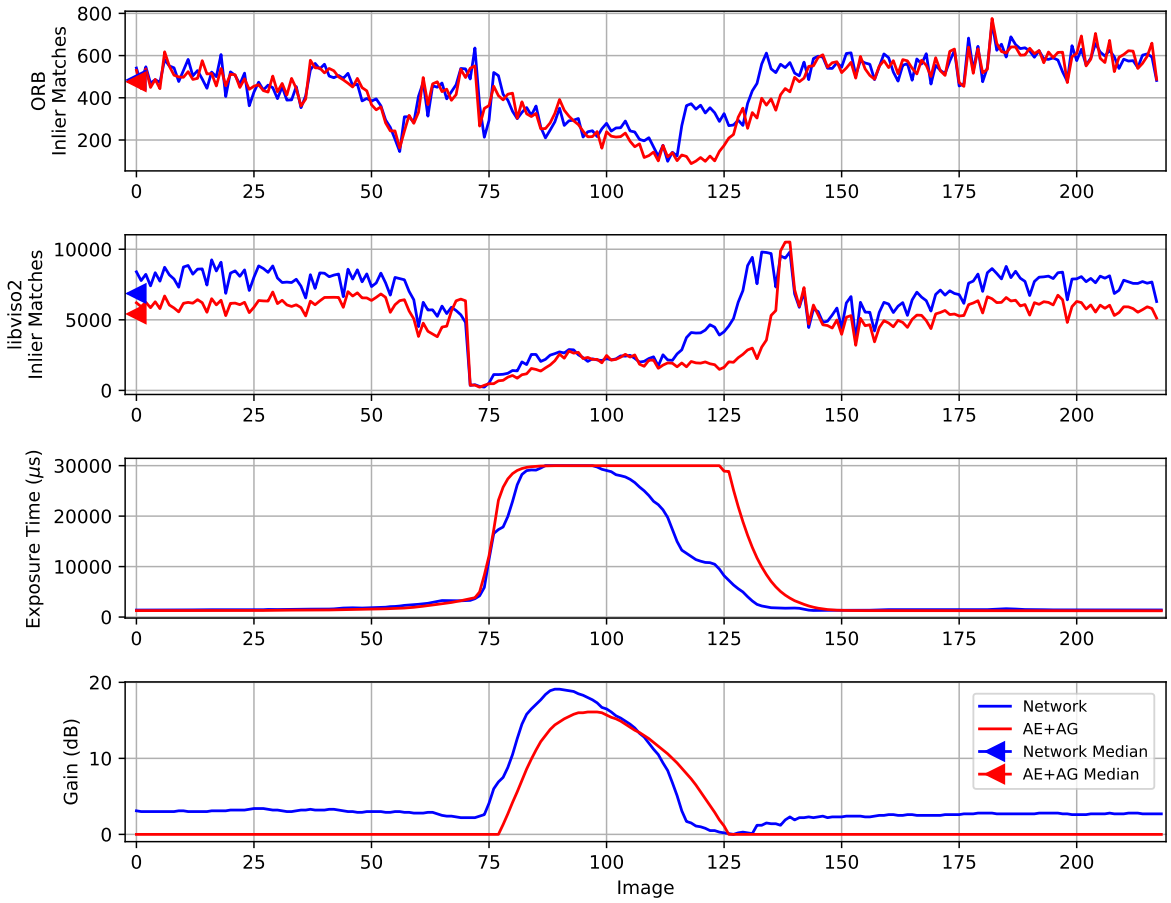


Figure 5.13: ORB inlier feature matches, `libviso2` inlier feature matches, exposure times, and gain values across a tunnel transition trajectory for our method and built-in AE+AG.

compensate for the brightness outside. The exposure time was increased similarly to AE+AG during the transition into the tunnel, however, it was reduced much earlier once the cameras began passing through the tunnel. The reduced exposure time within the tunnel produced images containing less blur and led to an increased number of inlier feature matches. Again, the largest increase in performance was found at the exit of the tunnel. The performance increase at the exit of the tunnel can be attributed to the earlier adjustments of the exposure time and gain relative to the built-in AE+AG algorithm. We note that although the gain was preemptively increased at the beginning of the tunnel transition relative to AE+AG, there was no discernible increase in inlier matches over the AE+AG algorithm during this transition. However, since the network did increase gain more preemptively, it can be expected that in other scenarios, the predictive nature would result in more inlier matches.

5.2.3 `Libviso2` Image Processing Experiment

To further support our approach, we investigated how our proposed network improved the actual performance of a VO pipeline. To measure if improved performance was obtained, we recorded the percentage of frames that were used to estimate pose change by monocular `libviso2` over the entire experimental route consisting of two tunnels described in section 5.1.2. Monocular `libviso2` reports a failure each time that it does not recover sufficient sequential inlier feature matches between frames. It is important to note that monocular `libviso2` also reports failures when there is little or no camera motion between properly exposed frames. Since the cameras captured images simultaneously, the number of failures related to a lack of motion of the cameras occurred for the same frames acquired by each approach. Although the failures caused by a lack of motion of the cameras lowered the overall number of processed frames, the comparison between the approaches is still valid. We tested monocular `libviso2` using two variants of the hybrid target generation method (Equation 4.13) where we selected differing α values. The first was with an equal weighting ($\alpha = 0.5$) between the windowed feature targets and windowed gridsearch targets denoted *equal*. The second was with weighting in favour of the gridsearch method ($\alpha = 0.33$), denoted *gridsearch-weighted*.

Table 5.5: A comparison between the number of frames processed by monocular `libviso2` using images captured by the dual-parameter network trained using the hybrid data generation method and built-in AE+AG. We also present a comparison between the mean number of inlier features relative to all identified features in the captured images.

Method	Frames Processed	Mean Inliers
AE+AG	65.17%	50.84%
Ours (Hybrid) (*Equal*)	**72.56%**	**58.36%**
AE+AG	52.43	40.48
Ours (Hybrid) (*Gridsearch-weighted*)	**56.69%**	**44.38%**

Table 5.5 shows that our method produced images that were better suited for use in a monocular VO system compared with built-in AE+AG algorithms. We found greater improvements over the automatic algorithms when using an equal weighting between training targets in Equation 4.13 for the hybrid target generation method. Improvements were likely due to the equal-weighted network variant favouring higher exposure time and gain values while still maintaining parameter stability. The higher exposure time and gain values produced images containing more features and yielding more inlier feature

matches.

5.3 Discussion

Our results indicate that predictively adjusting camera exposure time and gain parameters to produce consistently well-exposed images across lighting transitions can culminate in increased inlier feature matches between sequential frames. We found that it is important to adjust both exposure time and gain rather than simply adjusting just one parameter. Through appropriate adjustments of both parameters, we can ensure that acquired images are both bright and sharp.

Unlike the sampling-based strategies from the literature, our learning-based approach considers temporal information contained in the images and uses this information to predictively adjust the camera parameters. We demonstrated that the predictive capabilities of our networks resulted in large improvements in the quality of acquired images, particularly under dynamic lighting conditions. Although sampling-based strategies can produce high-quality images in controlled environments, it is apparent that these approaches are not suitable for use in dynamic situations. Furthermore, tuning the parameters of sampling-based methods can be difficult and time-consuming. Conversely, data-driven methods can account for environmental dynamics (e.g., transitioning from indoors to outdoors) and can improve over time with more data. Additionally, there are few parameters to tune once appropriate training hyperparameters have been identified. We note, however, that our approach is not entirely robust. Environmental changes or conditions that do not appear in the training data may yield nonsensical network outputs, and further experimentation is needed to determine how well the network generalizes. Further, the network was trained using images captured with a fixed lens aperture. If the lens aperture or camera system were changed, the network would likely no longer function correctly, as the relation between image intensity and the parameter settings would change. The network would need to be retrained with data captured using the new camera or using the new lens aperture setting.

The performance of our data-driven approaches is mainly limited by the quality and quantity of the training data. Since the single-parameter and dual-parameter networks are similar in size and architecture, we can attribute the difference in performance between networks to both the addition of gain targets generated with the dataset and the incorporation of blur in the training images. There are likely further improvements that can be obtained with an increase in the quantity and quality of training data, improved data collection methods, and careful tuning of the metric used.

One of the main challenges we faced was determining efficient and effective methods for collecting large amounts of useful training data. Collecting images for our initial single-parameter dataset proved to be an arduous task; we collected long trajectories composed of dozens of poses, each with fifty images, in varied environments. Determining an appropriate sampling strategy for the dual-parameter dataset also proved to be challenging, and further improvements can likely be made with better sampling.

Chapter 6

Conclusion

In this thesis, we examined how visual navigation algorithms, namely visual odometry, are heavily dependent on the quality of the captured input images. The image acquisition process is often overlooked, with users relying on built-in automatic camera parameter controllers or simply fixing the exposure time and gain values manually. These settings are normally not an issue under static lighting conditions, but in environments where the lighting can rapidly change, the parameter settings can produce severely overexposed or underexposed images that are missing important details useful to the VO pipeline.

6.1 Summary and Contributions

Our literature review revealed that the image acquisition process in the context of robot vision had not been thoroughly investigated. The relatively few papers related to camera parameter control have only focused on reactive control techniques (usually just exposure time). Additionally, the approaches outlined in these works typically measure image quality using a heuristic quality metric, without consideration for the final application or task. The reactive algorithms from the literature are often not suited for use in dynamic lighting conditions, and are instead designed to obtain "optimal" images, determined by a heuristic metric, in relatively static environments. The limitations of the techniques outlined in Chapter 3 presented an opportunity to improve the image acquisition process used by visual navigation algorithms by introducing a predictive camera parameter controller.

We presented an alternative to existing control algorithms (AE+AG, and methods from the literature) which are generally slow, reactive, and based on heuristics. Our data-driven approach used a CNN to predictively adjust camera gain and exposure time parameters to compensate for lighting changes expected to occur in future frames. We

trained the networks to learn the camera parameter values that resulted in images that are favourable for use in a VO system. We generated training targets for the networks by using the VO front end to inform the quality of training images. Making use of the VO front end to generate training targets for the networks allowed us to ensure that the learned camera parameter adjustments would produce improved images for feature matching. In the dual-parameter case, we found that the best way to generate training targets for the network was to use a combination of two image metrics derived from the VO front end. Namely, we combined camera parameter values that yielded images containing a high number of features with camera parameter values that yielded images containing a high number of inlier feature matches between two sequential frames.

We collected large datasets that contained images acquired with various parameter settings and used our proposed metrics to generate training targets. Our networks were trained using the collected datasets and training targets. We compared the performance of our networks with built-in AE+AG and the Shin et al. [41] approach from the literature by measuring the number of sequential inlier feature matches contained in the acquired images. Experiments with both single-parameter and dual-parameter networks were conducted in both static and dynamic lighting conditions. We found that our dual-parameter approach resulted in images with a higher number of inlier feature matches between frames.

We showed that both the single-parameter and dual-parameter networks can predict changes in lighting due to an approaching tunnel entrance or exit, and compensate for these changes by adjusting the camera parameters preemptively. The predictive behaviour of our networks resulted in images containing significantly more sequential ORB and `libviso2` inlier feature matches. Our networks favoured parameters that produced brighter images because there were more identifiable features in these images compared with AE+AG images. Unlike AE+AG, our networks learned that there are few important image features located in the sky and overexposed this region to ensure that the feature-rich regions of the scene remained bright. The behaviour of our networks relative to automatic algorithms supports the findings from Shim et al. [3] that built-in automatic algorithms generally attempt to compensate for the brightness of the entire scene, including the sky, and consequently reduce brightness in feature-rich regions of images. Finally, our dual-parameter method yielded a higher number of useful frames (i.e., image frames that contained sufficient information for estimation of inter-frame pose changes) using the `libviso2` monocular VO system compared with built-in AE+AG. Apart from the specifics of our results, this thesis further explored the notion proposed in Clement et al. [6] of using the VO pipeline to generate training targets for a neural network.

6.2 Potential Improvements

Although we were able to obtain convincing results in both the static and dynamic experiments, we believe that the quality of our results could be improved if some adjustments to the data collection and training methodologies are made. The main disadvantage, specific to the dual-parameter case, is that our training procedure samples the parameter space in a very sparse manner; only two images are captured at each pose. If a more refined sampling strategy could be employed, or better yet, if more images could be obtained at each pose (similar to the dataset used in the single-parameter case), we could generate better training targets. Furthermore, our training samples would likely not oscillate to the same extent as shown in Figure 5.9. Reduced oscillations may improve the network performance as targets generated under static lighting conditions would be relatively stable. Additionally, the more dense sampling of the parameter space would yield improved targets across lighting transitions.

Further improvements could be made if a larger and more diverse collection of training data could be collected. The main difficulty with this project was that the very nature of an image acquisition algorithm necessitates real-world experimentation and data collection. It was not feasible to simulate realistic environments or accurately model the effects of parameters changes on acquired images. It was difficult to manually collect a very large amount of training data and cumbersome to conduct multiple experiments. It was also difficult to iterate and test new ideas quickly as all testing needed to be conducted in real time during image capture, rather than offline using preexisting datasets. Exploring how to generate realistic data in simulation would help to increase the efficiency of data collection and experimental testing. Alternatively, automating real-world data collection through the use of algorithms such as visual teach and repeat could also improve the performance of our data-driven approach. Finally, further support for our method could be obtained through a more diverse set of experiments, for example, experiments onboard drones that transition between indoors and outdoors.

6.3 Future Work

Vision is likely to remain an integral component of mobile robotic perception systems. Novel localization, mapping, and perceptions algorithms continue to be developed for robotic systems. Consequently, the quality of acquired images remains an important consideration for future robotics research.

Specific to this thesis, there are several avenues of research that could be explored

in future work. One method we wish to explore is to iteratively train a model and use this model, in place of AE+AG outlined in Section 4.2.2, to collect additional training data. By replacing AE+AG with our initial trained model, the camera parameters used as a baseline for data collection would be closer to the 'optimal' values, and sampling a second time would explore the parameter space in more suitable regions. We could then iteratively train the network with the improved data and repeat this process. Iteratively training in this manner is essentially manual reinforcement learning (RL). An alternative avenue of research is the notion of formulating predictive camera parameter adjustments as an RL problem. The state and actions are relatively simple and clearly defined, and the reward function can be modelled to incorporate the specific robot vision task in a number of interesting ways. The downside of heuristically exploring the parameter space during data collection, as described in Section 4.2.2, would be replaced with strategies employed in RL. Further, an RL implementation could be automated and improved over time.

We also wish to investigate the incorporation of classical control techniques into the camera parameter control problem. The addition of classical control techniques, such as model predictive control, could provide predictive capabilities similar to our learned approach. Additionally, the well-defined theoretical framework of classical control could be leveraged in ways that are not possible with black box neural networks. Finally, we would like to explore the feasibility of using pose error as a training target for our networks, that is, determining the camera parameters used to acquire images that produce the most accurate pose estimates from a VO system.

In summary, while this thesis focused on improving the quality of images acquired in dynamic lighting conditions for use in VO, the problem of robust visual sensing is far from solved. We investigated how machine learning techniques can be leveraged to improve camera parameter control, but greater improvements can likely be achieved through the marriage of classical techniques and advances in machine learning.

Bibliography

[1] H. P. Moravec, *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford, CA, USA, 1980. AAI8024717.

[2] F. Fraundorfer and D. Scaramuzza, "Visual odometry: Part II: Matching, robustness, optimization, and applications," *IEEE Robotics and Automation Magazine*, vol. 19, no. 2, pp. 78–90, 2012.

[3] I. Shim, J. Lee, and I. S. Kweon, "Auto-adjusting camera exposure for outdoor robotics using gradient information," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1011–1017, 2014.

[4] H. Lu, H. Zhang, S. Yang, and Z. Zheng, "Camera parameters auto-adjusting technique for robust robot vision," in *2010 IEEE International Conference on Robotics and Automation*, pp. 1518–1523, 2010.

[5] J. Westerhoff, M. Meuter, and A. Kummert, "A generic parameter optimization workflow for camera control algorithms," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 944–949, 2015.

[6] L. Clement, M. Gridseth, J. Tomasi, and J. Kelly, "Learning Matchable Image Transformations for Long-Term Metric Visual Localization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1492–1499, 2020.

[7] R. Gomez-Ojeda, Z. Zhang, J. Gonzalez-Jimenez, and D. Scaramuzza, "Learning-Based Image Enhancement for Visual Odometry in Challenging HDR Environments," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 42, pp. 805–811, IEEE, 2018.

[8] H. Porav, W. Maddern, and P. Newman, "Adversarial training for adverse conditions: Robust metric localisation using appearance transfer," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1011–1018, 2018.

[9] M. Grundmann, C. McClanahan, S. B. Kang, and I. Essa, "Post-processing approach for radiometric self-calibration of video," in *IEEE International Conference on Computational Photography (ICCP)*, pp. 1–9, 2013.

[10] L. Clement and J. Kelly, "How to train a cat: Learning canonical appearance transformations for direct visual localization under illumination change," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2447–2454, 2018.

[11] S. Park, T. Schps, and M. Pollefeys, "Illumination change robustness in direct visual slam," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4523–4530, 2017.

[12] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[13] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[14] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 430–443, 2006.

[15] S. Leutenegger, M. Chli, and R. Siegwart, "BRISK: Binary Robust Invariant Scalable Keypoints," *2011 IEEE International Conference on Computer Vision (ICCV)*, pp. 1–8, 2011.

[16] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 778–792, Springer Berlin Heidelberg, 2010.

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2564–2571, IEEE, 2011.

[18] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "Lift: Learned invariant feature transform," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 467–483, Springer International Publishing, 2016.

[19] H. Yang, B. Wang, N. Vesdapunt, M. Guo, and S. B. Kang, "Personalized Exposure Control Using Adaptive Metering and Reinforcement Learning," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 8, pp. 1–17, 2018.

[20] B. K. Johnson, "Photographic exposure control system and method," U.S. Patent 4,423,936 3 Jan,. 1984.

[21] M. Muramatsu, "Photometry Device for a Camera," U.S. Patent 5,592,256, 7 Jan., 1997.

[22] N. Sampat, S. Venkataraman, T. Yeh, and R. Kremsens, "System implications of implementing auto-exposure on consumer digital cameras," *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 3650, pp. 100–107, 1999.

[23] I. Shim, T. H. Oh, J. Y. Lee, J. Choi, D. G. Choia, and I. S. Kweon, "Gradient-based Camera Exposure Control for Outdoor Mobile Platforms," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8215, no. c, pp. 1–14, 2018.

[24] L. Halodová, E. Dvořáková, F. Majer, J. Ulrich, T. Vintr, K. Kusumam, and T. Krajník, "Adaptive image processing methods for outdoor autonomous vehicles," in *Modelling and Simulation for Autonomous Systems*, pp. 456–476, Springer International Publishing, 2019.

[25] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

[26] R. Szeliski, *Computer Vision: Algorithms and Applications*. Texts in Computer Science, Springer London, 2010.

[27] P. E. Debevec and J. Malik, "Recovering high dynamic range radiance maps from photographs," in *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, (USA), pp. 369–378, ACM Press/Addison-Wesley Publishing Co., 1997.

[28] M. D. Grossberg and S. K. Nayar, "What is the space of camera response functions?," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 602–609, 2003.

[29] M. D. Grossberg and S. K. Nayar, "Determining the camera response from images: What is knowable?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 11, pp. 1455–1467, 2003.

[30] Yang Cheng, M. Maimone, and L. Matthies, "Visual Odometry on the Mars Exploration Rovers," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 903–910, 2005.

[31] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. I–I, 2004.

[32] D. Scaramuzza and F. Fraundorfer, "Visual odometry: Part I: The First 30 Years and Fundamentals," *IEEE Robotics and Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[33] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[34] H. C. Longuet-higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[35] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment - A Modern Synthesis," in *Vision Algorithms: Theory and Practice - International Workshop on Vision Algorithms, Proceedings*, pp. 298–372, 2000.

[36] N. Govender, "Evaluation of Feature Detection Algorithms for Structure from Motion," *Council for Scientific and Idustrial Research, Pretoria, Technical Report*, 2009.

[37] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI81, (San Francisco, CA, USA), pp. 674–679, Morgan Kaufmann Publishers Inc., 1981.

[38] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition CVPR-94*, pp. 593–600, IEEE Comput. Soc. Press, 1994.

[39] Z. Zhang, C. Forster, and D. Scaramuzza, "Active exposure control for robust visual odometry in hdr environments," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3894–3901, 2017.

[40] J. Kim, Y. Cho, and A. Kim, "Exposure control using bayesian optimization based on entropy weighted image gradient," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 857–864, 2018.

[41] U. Shin, J. Park, G. Shim, F. Rameau, and I. S. Kweon, "Camera exposure control for robust robot vision with noise-aware image quality assessment," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1165–1172, 2019.

[42] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[44] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, 2015.

[45] J. Kim and A. Kim, "Light condition invariant visual slam via entropy based image fusion," in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pp. 529–533, 2017.

[46] S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, "High dynamic range video," *ACM Transactions on Graphics*, vol. 22, pp. 319–325, July 2003.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[48] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[49] D. Nister, "An efficient solution to the five-point relative pose problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[50] I. Cvišić, J. Ćesić, I. Marković, and I. Petrović, "SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles," *Journal of Field Robotics*, vol. 35, no. 4, pp. 578–595, 2018.