On Learning Models of Appearance for Robust Long-term Visual Navigation

by

Lee Eric Clement

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

© Copyright 2020 by Lee Eric Clement

Abstract

On Learning Models of Appearance for Robust Long-term Visual Navigation

Lee Eric Clement
Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto
2020

Simultaneous localization and mapping (SLAM) is a class of techniques that allow robots to navigate unknown environments using onboard sensors. With inexpensive commercial cameras as the primary sensor, visual SLAM has become an important and widely used approach to enabling mobile robot autonomy. However, traditional visual SLAM algorithms use only a fraction of the information available from conventional cameras: in addition to the basic geometric cues typically used in visual SLAM, colour images encode information about the camera itself, environmental illumination, surface materials, vehicle motion, and other factors influencing the image formation process. Moreover, visual localization performance degrades quickly in long-term deployments due to environmental appearance changes caused by lighting, weather, or seasonal effects. This is especially problematic when continuous metric localization is required to drive vision-in-the-loop systems such as autonomous route following. This thesis explores several novel approaches to exploiting additional information from cameras to improve the accuracy and reliability of metric visual SLAM algorithms in short- and long-term deployments. First, we develop a technique for reducing drift error in visual odometry (VO) by estimating the position of a known light source such as the sun using indirect illumination cues available from existing image streams. We build and evaluate hand-engineered and learned models for single-image sun detection and achieve significant reductions in drift error over 30 km of driving in urban and planetary analogue environments. Second, we explore deep image-to-image translation as a means of improving metric visual localization under time-varying illumination. Using images captured under different illumination conditions in a common environment, we demonstrate that localization accuracy and reliability can be substantially improved by learning a many-to-one mapping to a user-selected canonical appearance condition. Finally, we develop a self-supervised method for learning a canonical appearance optimized for high-quality localization. By defining a differentiable surrogate loss function related to the performance of a non-differentiable localization pipeline, we train an optimal RGB-to-grayscale mapping for a given environment, sensor, and pipeline. Using synthetic and real-world long-term vision datasets, we demonstrate significant improvements in localization performance compared to standard grayscale images, enabling continuous metric localization over day-night cycles using a single mapping experience.

Acknowledgements

Research is rarely, if ever, the work of a single person, and this thesis is no exception: without the enduring support, encouragement, and insight of many people, you would not be reading these words. As a result, I have a long list of people to whom I want to express my gratitude. To my advisor, Jonathan Kelly, thank you not only for guiding and supporting the work presented in this thesis, but more importantly for encouraging me to grow as an independent researcher, communicator, and leader. It has been an incredible privilege to work with you as one of your first students and to build up the lab together. To my other committee members, Tim Barfoot and Angela Schoellig, thanks for all the helpful conversations and constructive feedback over the years; this thesis is stronger for it. To my long-suffering collaborator and frequent travel companion, Valentin Peretroukhin, thank you for joining me on this journey and so many others, and for sharing your ideas and insights so generously along the way. Thanks to the rest of the STARS Lab, past and present, especially Matt, Jacob, Brandon, Trevor, Oliver, Filip, Emmett, Olivier, Jordan, Adam, and Justin for helping build a fun and supportive group and for doing great work. To everyone I worked with in the ASA, GECoS, and elsewhere, thank you for your relentless energy, initiative, and commitment to making UTIAS and UofT a better place for everyone. Thanks also to NSERC and the Vector Institute for financially supporting my research. To my friends and family, your love and support over the years have meant the world to me. To my parents, Leslie and Armand, my sister Ashley, and my aunt Andrea, thank you for always cheering me on even if you couldn't make heads or tails of my work. To my cousin, Trish, even though we live half a country apart, I'm grateful for the time we spend together; you are every bit as much a role model as you are a friend. To David, Meredith, and Jen, thank you for making me a part of your family and helping me find my footing in a new city; Toronto wouldn't be the same without you. To Rebecca, your courage, independence, and integrity have been a continual source of inspiration, and I couldn't ask for a better roommate or friend. To Warren, thank you for always being someone I could turn to for advice and encouragement. To Karime, your strength, kindness, and creativity are something to aspire to; I can't imagine grad school without you. To June, our deep conversations and hiking adventures continue to be a source of endless joy; who would I be without knowing you? And to Paris, Dan, and Simon, thank you for being the family of overachieving queers I always needed, and for your unending love, generosity, and advice; home is you. Finally, thank you to the many many people I haven't named but who have nonetheless helped me along my path. All of you made this possible.

Contents

List of Tables

List of Figures

Notation

a	A real scalar quantity
$\mathbf{a}$	A real column vector
$\mathbf{A}$	A real matrix
$\mathbf{1}$	The identity matrix, sometimes denoted $\mathbf{1}_M$ to make its dimensions explicit
$\mathbf{0}$	The zero matrix, sometimes denoted $\mathbf{0}_{M\times N}$ to make its dimensions explicit
$\mathbb{R}^{M\times N}$	The vector space of real $M \times N$ matrices
$p(\mathbf{x})$	The probability density of a random variable $\mathbf{x}$
$p(\mathbf{x}\|\mathbf{y})$	The conditional probability density of $\mathbf{x}$ given $\mathbf{y}$
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	A Gaussian probability density with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$(\cdot)_k$	The value of a quantity at timestep k
$(\cdot)_{k_1:k_2}$	The value of a quantity from timestep k_1 to timestep k_2, inclusive
$\hat{(\cdot)}$	A posterior (estimated) quantity
$\check{(\cdot)}$	A prior quantity
$\underrightarrow{\mathcal{F}}_a$	A vectrix representing a reference frame in three dimensions
$\underrightarrow{a}$	A vector quantity in three dimensions
$\mathrm{SO}(3)$	The special orthogonal group, a matrix Lie group used to represent rotations
$\mathfrak{so}(3)$	The Lie algebra associated with SO(3)
$\mathrm{SE}(3)$	The special Euclidean group, a matrix Lie group used to represent poses
$\mathfrak{se}(3)$	The Lie algebra associated with SE(3)
$(\cdot)^\wedge$	An operator associated with the Lie algebra for rotations and poses
$(\cdot)^\vee$	The inverse operator associated with $(\cdot)^\wedge$
$\mathrm{Ad}(\cdot)$	An operator producing the adjoint of an element of a matrix Lie group
$\mathbf{C}_{ba}$	A 3×3 rotation matrix (member of SO(3)) that takes points expressed in $\underrightarrow{\mathcal{F}}_a$ and re-expresses them in $\underrightarrow{\mathcal{F}}_b$, which is rotated with respect to $\underrightarrow{\mathcal{F}}_a$
$\mathbf{T}_{ba}$	A 4×4 transformation matrix (member of SE(3)) that takes points expressed in $\underrightarrow{\mathcal{F}}_a$ and re-expresses them in $\underrightarrow{\mathcal{F}}_b$, which is rotated and/or translated with respect to $\underrightarrow{\mathcal{F}}_a$

Chapter 1

Introduction

1.1 Motivation

Simultaneous localization and mapping (SLAM) encompasses a range of techniques that allow robots to navigate unknown environments using onboard sensors, jointly solving the twin problems of mapping the environment and determining the robot's location (i.e., localizing itself) within it. In the last decade, robotics has entered the "robust perception age", and a major focus of modern SLAM research has been the design of perception systems capable of operating over extended periods of time in a broad range of environments (Kelly et al., 2012; Cadena et al., 2016). Visual SLAM in particular holds great promise in this area due to the wealth of information available from standard colour cameras, which, to date, has not been fully exploited. Traditional visual SLAM formulations are based on the prediction of visual measurements (e.g., the positions of landmarks or the intensity of pixels) from some underlying model, and the minimization of modelling errors between the predicted and observed measurements. While these methods are well studied and supported by a large body of literature, the models used in practice are typically hand-engineered approximations of the complex process of image formation. These models rely on simplifying assumptions such as brightness constancy or feature descriptor invariance in order to be analytically and computationally tractable, which limits their usefulness in situations where these assumptions are severely violated, for example under conditions of environmental appearance change in long-term autonomy applications. Moreover, these simplified models do not take full advantage of visual data, which encode information about the scene being imaged (e.g., geometry, materials, and illumination), the imaging sensor itself (e.g., optics and photometric response), and the motion of the vehicle (e.g., from motion blur).

When faced with the task of describing complex phenomena whose underlying physical processes are difficult to model analytically with sufficient fidelity for real-world applications, an appealing and increasingly popular approach is to learn such models from data. In the case of visual data, deep convolutional neural networks (CNNs) (LeCun, 1989; LeCun et al., 2015) have proven to be useful tools for a variety of perception tasks due to their ability to learn salient visual features or representations tailored to a specific task, as well as their ability to efficiently encode hierarchical relationships between low-level features such as edges and corners and high-level features such as buildings and people. Traditionally, the primary applications of CNNs have been classification tasks such as object recognition, but increasingly we are seeing the application of deep convolutional models to localization tasks that have

until recently been the exclusive domain of traditional geometric computer vision (Kendall et al., 2015; Costante et al., 2016; Flynn et al., 2016; Haarnoja et al., 2016; Handa et al., 2016). Combined with the production of new long-term vision datasets such as the University of Michigan North Campus dataset (Carlevaris-Bianco et al., 2015), the Oxford RobotCar dataset (Maddern et al., 2017), and the UTIAS Multi-Season dataset (Paton et al., 2017), among others, the time is ripe to apply the methods of deep learning to the problem of localization under appearance change.

Recent works by Kendall et al. (2015), Costante et al. (2016), Handa et al. (2016), Wang et al. (2018), and others exemplify one approach to applying deep learning to robot navigation, where the goal is to fully replace hand-engineered localization pipelines with end-to-end learning. These methods hearken back to early connectionist approaches to visual navigation (Pomerleau, 1989), and while they are interesting proofs of concept, it is unclear whether end-to-end learning will ultimately be able to achieve the accuracy, reliability, and generality of well-understood model-based SLAM methods without massive corpuses of training data. On the other hand, an integrative approach may prove more immediately fruitful. Rather than *replacing* model-based techniques wholesale, deep learning could be used as a *supplement* to deal with unmodelled complexities such as appearance change. This approach invites us to develop complementary, data-driven methods for coping with appearance change in visual SLAM while still retaining the familiar estimation machinery on which modern SLAM systems are typically based. As a result, we can preserve many desirable properties of model-based systems, including interpretability and generality deriving from broadly applicable first principles, while simultaneously improving their robustness outside the modelling regime. Moreover, models impose a strong prior that benefits the learning problem by increasing data efficiency and discouraging high-capacity models such as deep neural networks from "reinventing the wheel" by solving tasks for which good algorithmic solutions already exist (Sünderhauf et al., 2018).

In this thesis we are concerned with developing models of appearance suitable for use with standard metric visual SLAM techniques, and applying these models as enablers of long-term mobile autonomy. We explore several novel approaches to exploiting additional information from vision sensors for the purpose of improving the accuracy and reliability of metric visual SLAM in short- and long-term deployments. In the short term, we consider the problem of dead-reckoning drift error in visual odometry (VO), which is a subproblem of SLAM concerned principally with motion estimation. Drift error in VO is chiefly a result of accumulated orientation error over long trajectories, and many state-of-the-art SLAM systems rely on loop closure detection and pose-graph optimization to correct for orientation drift. Others exploit auxiliary sensors such as GNSS, sun sensors, and inclinometers to obtain orientation information in a global frame. Rather than relying on additional sensors or potentially non-existent loop closures, we are interested in modelling environmental illumination as a means of injecting global orientation information into the VO pipeline to correct for long-term drift. By estimating the position of a known light source such as the sun using indirect illumination cues available from existing image streams, we can achieve the benefits of sun sensing without the need for specialized sun sensor hardware or direct observations of the sun. We demonstrate through extensive experimentation on urban and planetary analogue datasets that a CNN trained to regress a three-dimensional unit vector measuring the direction of the sun from a single RGB (red-green-blue) image provides a sufficiently high quality global orientation signal to substantially correct accumulated orientation error when used as an auxiliary 'pseudo-sensor' in a conventional VO pipeline.

For long-term deployments, environmental appearance change presents a significant impediment to establishing correspondences for metric visual localization against an existing map, whether due to illumination variation over the course of a day, changes in weather conditions, or seasonal appearance variations. While other sensing modalities such as lidar[1] can provide illumination invariance in a SLAM context (McManus et al., 2013), these sensors are often heavy and prohibitively expensive compared to cameras, which limits their usefulness in many robotics applications.[2] In the context of feature-based (indirect) visual localization, approaches to compensating for appearance change have ranged from incorporating additional sources of information such as colour-constant images (Paton et al., 2017; Clement et al., 2017a), which must be tuned for a specific environment-sensor pair, to combining multiple experiences of an environment into a spatiotemporal map (Churchill and Newman, 2013; Linegar et al., 2015; Paton et al., 2016), which requires storing and searching a large database of visual experiences. Attempts to adapt photometric (direct) methods to these cases have largely relied on affine approximations of brightness change and other simple analytical image transformations (Engel et al., 2015; Park et al., 2017), which do not fully capture the effects of environmental appearance change. Others have attempted to negate the influence of appearance change by reframing the problem in an appearance-invariant manner, for example using additional sensing modalities such as 3D lidar and information-theoretic matching strategies (Wolcott and Eustice, 2014; Pascoe et al., 2015). In this thesis, we investigate two varieties of learned image transformations to improve the robustness of metric visual localization to long-term appearance change. First, we use deep image-to-image translation (Isola et al., 2017; Zhu et al., 2017; Liu et al., 2017; Choi et al., 2018) as a means of improving visual localization under time-varying illumination conditions. Using training images captured under different illumination conditions in a common environment, we demonstrate that localization accuracy and reliability can be substantially improved by learning a many-to-one mapping to a user-selected canonical appearance condition. Finally, we develop a self-supervised method for learning a canonical appearance optimized for high-quality localization, based loosely on colour-constancy theory (Ratnasingam and Collins, 2010). By defining a differentiable surrogate loss function related to the performance of a non-differentiable localization pipeline, our method learns an optimal RGB-to-grayscale mapping for a given environment, sensor, and localization pipeline. Through experiments on synthetic and real-world long-term vision datasets, we demonstrate significant improvements in localization performance compared to standard grayscale images, enabling continuous metric localization over a continuous 30-hour period using a single mapping experience, and presenting a viable strategy for reducing the data requirements of state-of-the-art experience-based localization pipelines such as that of Paton et al. (2016).

1.2 Thesis Structure

Chapter 2 provides a background discussion of concepts in 3D state estimation, visual localization and mapping, and deep learning that are critical to the understanding of this thesis. Chapter 3 develops a method for reducing drift error in visual odometry by learning to interpret image data in order to detect a known light source such as the sun. Chapter 4 investigates the use of deep image-to-image translation as a means of enabling visual localization across changing appearance conditions, while

[1] Lidar, sometimes used as an acronym of "light detection and ranging", measures the position and reflectivity of a target by illuminating the target with laser light and measuring the reflected light with a photodetector.

[2] Significant industrial research effort is presently being expended on the development of compact, low-cost 3D lidar sensors.

Chapter 5 develops a more restricted set of deep models to learn image transformation functions that are optimized for high-quality localization rather than visual consistency. Each chapter generally outlines the motivation for the work, reviews related work in the literature, describes the methodology and experimental results validating the proposed method, summarizes the novel contributions, and provides references to associated publications. Finally, Chapter 6 concludes the thesis by summarizing the novel contributions, proposing potential avenues for future work, and offering closing remarks.

1.3 Associated Publications

The following first-author papers have appeared for publication and comprise the technical content of this thesis:

- Clement, L., Peretroukhin, V., and Kelly, J. (2017b). Improving the accuracy of stereo visual odometry using visual illumination estimation. In *Proceedings of the 2016 International Symposium on Experimental Robotics*, Springer Proceedings in Advanced Robotics, pages 409–419, Tokyo, Japan. Springer International Publishing.

- Peretroukhin, V., Clement, L., and Kelly, J. (2017). Reducing drift in visual odometry by inferring sun direction using a bayesian convolutional neural network. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2035–2042, Singapore.
  Note: Peretroukhin and Clement contributed equally to this work and jointly claim first authorship.

- Peretroukhin, V., Clement, L., and Kelly, J. (2018). Inferring sun direction to improve visual odometry: A deep learning approach. *International Journal of Robotics Research*, 37(9):996–1016.
  Note: Peretroukhin and Clement contributed equally to this work and jointly claim first authorship.

- Clement, L. and Kelly, J. (2018). How to train a CAT: Learning canonical appearance transformations for direct visual localization under illumination change. *IEEE Robotics and Automation Letters*, 3(3):2447–2454.

- Clement, L., Gridseth, M., Tomasi, J., and Kelly, J. (2019a). Learning maximally matchable image transformations for long-term metric visual localization. arXiv:1904.01080.

- Clement, L., Gridseth, M., Tomasi, J., and Kelly, J. (2019b). Matchable colorspace transformations for long-term metric visual localization. In *2019 CVPR Workshop on Image Matching*.

Much of the inspiration for this thesis originated in earlier work not presented here, which extended the stereo-vision-based autonomous route following system of Furgale and Barfoot (2010) to monocular vision and improved its robustness to daytime illumination change using the colour-constant image transformations derived by Paton et al. (2015). The interested reader will find detailed descriptions of this work in the following publications:

- Clement, L., Kelly, J., and Barfoot, T. D. (2016). Monocular visual teach and repeat aided by local ground planarity. In Wettergreen, D. S. and Barfoot, T. D., editors, *Proceedings of Field and Service Robotics*, Springer Tracts in Advanced Robotics, pages 547–561. Springer International Publishing, Toronto, Canada

- Clement, L., Kelly, J., and Barfoot, T. D. (2017a). Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery. *Journal of Field Robotics*, 34(1):74–97

1.4 Associated Videos

- Video summary of Sun-BCNN (Chapter 3) as presented at the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore:
  https://www.youtube.com/watch?v=c5XTrq3a2tE
- Video summary of Canonical Appearance Transformations (Chapter 4) as presented at the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia:
  https://www.youtube.com/watch?v=ej6VNBq3dDE

Chapter 2

Background

This chapter provides background information on key concepts used in this thesis, namely three-dimensional geometry, nonlinear state estimation, the design of visual localization and mapping systems, and the fundamentals of deep learning. No novel contributions are presented in this background chapter.

2.1 State Estimation in Three Dimensions

In this thesis we are often concerned with estimating the six-degrees-of-freedom (6-dof) pose of a robot or sensor in three-dimensional Euclidean space. The *pose* of a rigid body encapsulates both the three-dimensional *position* and the 3-dof *orientation* of the body with respect to a chosen reference frame. This section discusses important concepts in three-dimensional geometry and state estimation as they pertain to the remainder of the thesis. We refer the reader to Barfoot (2017) for a more detailed treatment of this topic.

2.1.1 Matrix Lie Groups

We represent rotations and poses in this thesis using matrix Lie groups. A *group* is a set G that, taken together with a binary operation $\bullet$, satisfies the four group axioms:

$$\begin{aligned}
\text{Closure:} \quad & a \bullet b \in G, \ \forall a, b \in G; \\
\text{Associativity:} \quad & (a \bullet b) \bullet c = a \bullet (b \bullet c), \ \forall a, b, c \in G; \\
\text{Identity element:} \quad & \exists e \in G \text{ such that } e \bullet a = a \bullet e = a, \ \forall a \in G; \text{ and} \\
\text{Inverse element:} \quad & \forall a \in G, \ \exists b \in G \text{ such that } a \bullet b = b \bullet a = e.
\end{aligned}$$

In order for a group G to be considered a *Lie group*, it must also be a differentiable manifold (i.e., one that is locally similar enough to a linear space to allow one to do calculus) with the property that the group operations are smooth (i.e., having derivatives of all orders everywhere in its domain). To be a *matrix Lie group*, the elements of G must be matrices and the operation $\bullet$ must be matrix multiplication.

Rotations Rotations in three dimensions can be represented as elements of the *special orthogonal* matrix Lie group SO(3), which is defined as the set of 3×3 rotation matrices:

$$\mathrm{SO}(3) = \left\{ \mathbf{C} \in \mathbb{R}^{3\times 3} \;\middle|\; \mathbf{C}\mathbf{C}^T = \mathbf{1},\ \det(\mathbf{C}) = +1 \right\}. \tag{2.1}$$

The *orthogonality* condition $\mathbf{C}\mathbf{C}^T = \mathbf{1}$ imposes six constraints on the nine parameters of the matrix, reducing the number of degrees of freedom to three. Choosing $\det(\mathbf{C}) = +1$ ensures that $\mathbf{C}$ is a *proper rotation* as opposed to a rotary reflection (which would be the case for $\det(\mathbf{C}) = -1$).

Poses By combining a three-dimensional rotation $\mathbf{C}$ with a three-dimensional translation $\mathbf{r}$, we can describe the 6-dof *pose* of a rigid body. Poses can be represented as elements of the *special Euclidean* matrix Lie group SE(3), which is defined as the set of 4×4 transformation matrices:

$$\mathrm{SE}(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \;\middle|\; \mathbf{C} \in \mathrm{SO}(3),\ \mathbf{r} \in \mathbb{R}^3 \right\}. \tag{2.2}$$

The matrix $\mathbf{T}$ has sixteen parameters and ten constraints (six from the orthogonality condition on $\mathbf{C}$ and four from the entries of the bottom row), leaving six degrees of freedom (three for rotation and three for translation).

2.1.1.1 Lie Algebras

Every matrix Lie group has an associated *Lie algebra* consisting of a vector space $\mathbb{V}$ over some field $\mathbb{F}$ (which will be $\mathbb{R}$ for our purposes), together with a binary operation called the Lie bracket $[\cdot, \cdot]$ that satisfies the following four properties for all $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \in \mathbb{V}$ and $a, b \in \mathbb{F}$:

$$\begin{aligned}
\text{Closure:} \quad & [\mathbf{X}, \mathbf{Y}] \in \mathbb{V}; \\
\text{Bilinearity:} \quad & [a\mathbf{X} + b\mathbf{Y}, \mathbf{Z}] = a\,[\mathbf{Z}, \mathbf{X}] + b\,[\mathbf{Z}, \mathbf{Y}]\,; \\
\text{Alternating:} \quad & [\mathbf{X}, \mathbf{X}] = \mathbf{0};\ \text{and} \\
\text{Jacobi identity:} \quad & [\mathbf{X}, [\mathbf{Y}, \mathbf{Z}]] + [\mathbf{Z}, [\mathbf{Y}, \mathbf{X}]] + [\mathbf{Y}, [\mathbf{Z}, \mathbf{X}]] = \mathbf{0}.
\end{aligned}$$

The vector space of a Lie algebra is the *tangent space* of the associated Lie group at the identity element of the group, and completely captures the local structure of the group.

Rotations The Lie algebra associated with SO(3) is given by

$$\begin{aligned}
\text{Vector space:} \quad & \mathfrak{so}(3) = \left\{ \boldsymbol{\Phi} = \boldsymbol{\phi}^\wedge \in \mathbb{R}^{3\times 3} \;\middle|\; \boldsymbol{\phi} \in \mathbb{R}^3 \right\}, \\
\text{Field:} \quad & \mathbb{R}, \\
\text{Lie bracket:} \quad & [\boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2] = \boldsymbol{\Phi}_1\boldsymbol{\Phi}_2 - \boldsymbol{\Phi}_2\boldsymbol{\Phi}_1,
\end{aligned}$$

where

$$\boldsymbol{\phi}^\wedge = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 3}, \qquad \boldsymbol{\phi} \in \mathbb{R}^3. \tag{2.3}$$

The operator $(\cdot)^\wedge : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ defines a linear mapping from a coordinate vector $\boldsymbol{\phi}$ to its corresponding vector space element $\boldsymbol{\Phi}$. We will also make use of the inverse operator $(\cdot)^\vee : \mathbb{R}^{3\times3} \to \mathbb{R}^3$ to go the other way.

Poses The Lie algebra associated with SE(3) is given by

$$\begin{aligned} \text{Vector space:} \quad & \mathfrak{se}(3) = \left\{ \boldsymbol{\Xi} = \boldsymbol{\xi}^\wedge \in \mathbb{R}^{4\times4} \;\middle|\; \boldsymbol{\xi} \in \mathbb{R}^6 \right\}, \\ \text{Field:} \quad & \mathbb{R}, \\ \text{Lie bracket:} \quad & [\boldsymbol{\Xi}_1, \boldsymbol{\Xi}_2] = \boldsymbol{\Xi}_1\boldsymbol{\Xi}_2 - \boldsymbol{\Xi}_2\boldsymbol{\Xi}_1, \end{aligned}$$

where

$$\boldsymbol{\xi}^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{4\times4}, \qquad \boldsymbol{\rho}, \boldsymbol{\phi} \in \mathbb{R}^3. \tag{2.4}$$

Analogously to the SO(3) case, the operator $(\cdot)^\wedge : \mathbb{R}^6 \to \mathbb{R}^{4\times4}$ defines a linear mapping from a coordinate vector $\boldsymbol{\xi}$ to its corresponding vector space element $\boldsymbol{\Xi}$. We denote the inverse operator as $(\cdot)^\vee : \mathbb{R}^{4\times4} \to \mathbb{R}^6$. Note that $\boldsymbol{\phi}^\wedge$ in Equation (2.4) is still computed using Equation (2.3).

2.1.1.2 The Exponential Map

The *exponential map* relates a matrix Lie group to its associated Lie algebra. In general, for a square matrix $\mathbf{A} \in \mathbb{R}^{M\times M}$, the matrix exponential and matrix logarithm are given by

$$\exp(\mathbf{A}) = \sum_{n=0}^{\infty} \frac{1}{n!}\mathbf{A}^n, \tag{2.5}$$

$$\ln(\mathbf{A}) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n}(\mathbf{A} - \mathbf{1})^n. \tag{2.6}$$

Rotations We can relate elements of SO(3) to elements of $\mathfrak{so}(3)$ in closed form using the exponential map:

$$\begin{aligned} \mathbf{C} &= \exp(\boldsymbol{\phi}^\wedge) && (2.7)\\ &= \sum_{n=0}^{\infty} \frac{1}{n!}(\boldsymbol{\phi}^\wedge)^n && (2.8)\\ &= \cos\phi\ \mathbf{1} + (1 - \cos\phi)\ \mathbf{a}\mathbf{a}^T + \sin\phi\ \mathbf{a}^\wedge, && (2.9) \end{aligned}$$

where $\phi = \|\boldsymbol{\phi}\|$ is the rotation angle and $\mathbf{a} = \boldsymbol{\phi}/\phi$ is the unit-length axis of rotation. Note that this expression for the exponential map is precisely Rodriguez's rotation formula for computing the rotation matrix corresponding to a rotation by an angle ϕ about an axis $\mathbf{a}$.

An important property of the exponential map from $\mathfrak{so}(3)$ to SO(3) is that it is *surjective* (i.e., every element of SO(3) can be generated from an element of $\mathfrak{so}(3)$) but *non-injective* (i.e., multiple elements of $\mathfrak{so}(3)$ may correspond to a single element of SO(3)). This means that the inverse mapping (i.e., the logarithm) from SO(3) to $\mathfrak{so}(3)$ is *non-unique*, which can be seen by observing that adding a multiple of 2π to the rotation angle ϕ in Equation (2.9) will yield the same $\mathbf{C}$. If we choose to constrain $\phi \in (-\pi, \pi]$,

we can invert the exponential map in closed form as follows:

$$\boldsymbol{\phi} = \ln(\mathbf{C})^{\vee} \tag{2.10}$$

$$= \frac{\phi}{2\sin\phi}\left(\mathbf{C} - \mathbf{C}^{T}\right)^{\vee} \tag{2.11}$$

where

$$\phi = \cos^{-1}\left(\frac{\operatorname{tr}(\mathbf{C}) - 1}{2}\right). \tag{2.12}$$

Note that in the limit of $\phi \to 0$, Equation (2.11) takes on an indeterminate form, which can be a source of numerical instability in software implementations when ϕ is small. For small values of ϕ, we can obtain a first order approximation of the exponential map by taking the first two terms of Equation (2.8):

$$\mathbf{C} = \exp(\boldsymbol{\phi}^{\wedge}) \approx \mathbf{1} + \boldsymbol{\phi}^{\wedge}, \qquad |\phi| \ll 1. \tag{2.13}$$

We can then invert Equation (2.13) to approximate the logarithmic map to first order:

$$\boldsymbol{\phi} = \ln(\mathbf{C})^{\vee} \approx (\mathbf{C} - \mathbf{1})^{\vee}, \qquad |\phi| \ll 1. \tag{2.14}$$

Poses Similarly, we can relate elements of SE(3) to elements of $\mathfrak{se}(3)$ in closed form using the exponential map:

$$\mathbf{T} = \exp(\boldsymbol{\xi}^{\wedge}) \tag{2.15}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!}\left(\boldsymbol{\xi}^{\wedge}\right)^{n} \tag{2.16}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!}\left(\begin{bmatrix}\boldsymbol{\rho}\\ \boldsymbol{\phi}\end{bmatrix}^{\wedge}\right)^{n} \tag{2.17}$$

$$= \sum_{n=0}^{\infty} \frac{1}{n!}\begin{bmatrix}\boldsymbol{\phi}^{\wedge} & \boldsymbol{\rho}\\ \mathbf{0} & 0\end{bmatrix}^{n} \tag{2.18}$$

$$= \begin{bmatrix}\sum_{n=0}^{\infty} \frac{1}{n!}\left(\boldsymbol{\phi}^{\wedge}\right)^{n} & \left(\sum_{n=0}^{\infty} \frac{1}{(n+1)!}\left(\boldsymbol{\phi}^{\wedge}\right)^{n}\right)\boldsymbol{\rho}\\ \mathbf{0} & 1\end{bmatrix} \tag{2.19}$$

$$= \begin{bmatrix}\mathbf{C} & \mathbf{r}\\ \mathbf{0} & 1\end{bmatrix}, \tag{2.20}$$

where

$$\mathbf{r} = \mathbf{J}\boldsymbol{\rho} \in \mathbb{R}^{3}, \tag{2.21}$$

with

$$\mathbf{J} = \sum_{n=0}^{\infty} \frac{1}{(n+1)!}\left(\boldsymbol{\phi}^{\wedge}\right)^{n}. \tag{2.22}$$

$\mathbf{J}$ is called the *left Jacobian* of SO(3), and it relates the translational component of $\mathfrak{se}(3)$ to the translational component of SE(3). Both $\mathbf{J}$ and its inverse can be written in closed form:

$$\mathbf{J} = \frac{\sin\phi}{\phi}\,\mathbf{1} + \left(1 - \frac{\sin\phi}{\phi}\right)\,\mathbf{a}\mathbf{a}^T + \frac{1-\cos\phi}{\phi}\,\mathbf{a}^\wedge, \tag{2.23}$$

$$\mathbf{J}^{-1} = \frac{\phi}{2}\cot\frac{\phi}{2}\,\mathbf{1} + \left(1 - \frac{\phi}{2}\cot\frac{\phi}{2}\right)\,\mathbf{a}\mathbf{a}^T - \frac{\phi}{2}\,\mathbf{a}^\wedge. \tag{2.24}$$

We can use $\mathbf{J}^{-1}$ to invert the exponential map for SE(3) as follows:

$$\boldsymbol{\xi} = \ln(\mathbf{T})^\vee \tag{2.25}$$

$$= \ln\left(\begin{bmatrix}\mathbf{C} & \mathbf{r}\\ \mathbf{0} & 1\end{bmatrix}\right)^\vee \tag{2.26}$$

$$= \begin{bmatrix}\mathbf{J}^{-1}\mathbf{r}\\ \ln(\mathbf{C})^\vee\end{bmatrix} \tag{2.27}$$

$$= \begin{bmatrix}\boldsymbol{\rho}\\ \boldsymbol{\phi}\end{bmatrix}. \tag{2.28}$$

Due to the embedded rotation, the exponential map from $\mathfrak{se}(3)$ to SE(3) is also surjective and non-injective and therefore the inverse mapping is non-unique as well.

In the limit of $\phi \to 0$, Equations (2.23) and (2.24) both take on indeterminate forms, which can also be a source of numerical instability for small ϕ. We can approximate Equation (2.23) to first order by taking the first two terms of Equation (2.22) to obtain

$$\mathbf{J} \approx \mathbf{1} + \frac{1}{2}\boldsymbol{\phi}^\wedge, \qquad |\phi| \ll 1. \tag{2.29}$$

The first-order approximation of Equation (2.24) is correspondingly given by

$$\mathbf{J}^{-1} \approx \mathbf{1} - \frac{1}{2}\boldsymbol{\phi}^\wedge, \qquad |\phi| \ll 1. \tag{2.30}$$

2.1.1.3 Adjoints

When working with matrix Lie groups, it is often necessary to transform a tangent vector from the tangent space of one element to the tangent space of another. Every element $\mathbf{X}$ of a matrix Lie group has an associated *adjoint*, denoted $\mathrm{Ad}\,(\mathbf{X})$, that transforms a tangent vector from the tangent space on the right side of the element to the tangent space on the left.

Rotations For SO(3), we have the relationship

$$\mathbf{C}\ \exp(\boldsymbol{\phi}^\wedge) = \exp\left((\mathrm{Ad}\,(\mathbf{C})\ \boldsymbol{\phi})^\wedge\right)\ \mathbf{C}, \tag{2.31}$$

and the adjoint is given by

$$\mathrm{Ad}\,(\mathbf{C}) = \mathbf{C}. \tag{2.32}$$

Poses Analogously for SE(3), we have

$$\mathbf{T}\ \exp(\boldsymbol{\xi}^\wedge) = \exp\left((\mathrm{Ad}\left(\mathbf{T}\right)\ \boldsymbol{\xi})^\wedge\right)\ \mathbf{T}, \tag{2.33}$$

and the adjoint can be constructed directly from the components of $\mathbf{T}$:

$$\mathrm{Ad}\left(\mathbf{T}\right) = \begin{bmatrix} \mathbf{C} & \mathbf{r}^\wedge\mathbf{C} \\ \mathbf{0} & \mathbf{C} \end{bmatrix} \in \mathbb{R}^{6\times 6}. \tag{2.34}$$

2.1.2 Batch Discrete-Time Nonlinear Estimation

We are often concerned with estimating the motion of a robot or sensor platform as the robot's state evolves (nonlinearly) over time and as the robot makes (nonlinear) measurements of the environment. Classically, recursive techniques such as the Extended Kalman Filter (EKF) and its variants have been employed to estimate the robot state as new measurements become available. However, because past states are marginalized out, filter-based methods scale poorly to long trajectories and cannot straightforwardly incorporate auxiliary sources of long-range information such as loop closures.

Many modern robotic systems instead rely on batch methods to jointly estimate all the state parameters of interest over the entire trajectory or over a subset of it (e.g., a sliding window). These methods typically make use of iterative nonlinear optimization techniques such as the Gauss-Newton or Levenberg-Marquardt algorithms to compute the *maximum a posteriori* (MAP) solution yielding the most likely posterior state based on the information available (initial state, measurements, and inputs) and knowledge or assumptions about process and measurement noise.

In this section we derive the general batch MAP solution for a discrete-time, nonlinear, time-varying system, and discuss how this solution can be adapted for estimation on matrix Lie groups under the assumption of Gaussian noise models.

2.1.2.1 Problem Setup

We define the following quantities:

$$\text{System state:} \quad \mathbf{x}_k \in \mathbb{R}^N \tag{2.35}$$

$$\text{Initial state:} \quad \mathbf{x}_0 \in \mathbb{R}^N \sim \mathcal{N}\left(\check{\mathbf{x}}_0, \check{\mathbf{P}}_0\right) \tag{2.36}$$

$$\text{Input:} \quad \mathbf{v}_k \in \mathbb{R}^N \tag{2.37}$$

$$\text{Process noise:} \quad \mathbf{w}_k \in \mathbb{R}^N \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_k\right) \tag{2.38}$$

$$\text{Measurement:} \quad \mathbf{y}_k \in \mathbb{R}^M \tag{2.39}$$

$$\text{Measurement noise:} \quad \mathbf{n}_k \in \mathbb{R}^M \sim \mathcal{N}\left(\mathbf{0}, \mathbf{R}_k\right) \tag{2.40}$$

where k is the discrete time index. From these we can define nonlinear motion and observation models:

$$\text{Motion model:} \quad \mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k), \qquad k = 1 \ldots K \tag{2.41}$$

$$\text{Measurement model:} \quad \mathbf{y}_k = \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k), \qquad k = 0 \ldots K \tag{2.42}$$

.

2.1.2.2 Maximum A Posteriori (MAP) Estimation

In batch estimation, our goal is to find the 'best' single estimate of the system state at all timesteps based on our knowledge of the initial state, inputs, and measurements. Mathematically, we want to solve the following MAP problem:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|\mathbf{v}, \mathbf{y}) \tag{2.43}$$

where $\mathbf{x} = \mathbf{x}_{0:K} = (\mathbf{x}_0, \ldots, \mathbf{x}_K)$, $\mathbf{v} = (\check{\mathbf{x}}_0, \mathbf{v}_{1:K}) = (\check{\mathbf{x}}_0, \mathbf{v}_1, \ldots, \mathbf{v}_K)$ and $\mathbf{y} = \mathbf{y}_{0:K} = (\mathbf{y}_0, \ldots, \mathbf{y}_K)$. We have included the initial state with the inputs because these terms together define our prior over the full set of states (i.e., we could integrate the motion model in Equation (2.42) using only these quantities to obtain a prior estimate of the state). The purpose of the measurements is to improve on this prior information.

Using Bayes' rule we can rewrite the MAP estimate as

$$\begin{aligned}
\hat{\mathbf{x}} &= \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|\mathbf{v}, \mathbf{y}) && (2.44)\\
&= \underset{\mathbf{x}}{\operatorname{argmax}}\, \frac{p(\mathbf{y}|\mathbf{x}, \mathbf{v})p(\mathbf{x}|\mathbf{v})}{p(\mathbf{y}|\mathbf{v})} && (2.45)\\
&= \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x})p(\mathbf{x}|\mathbf{v}) && (2.46)
\end{aligned}$$

where we have dropped the denominator because it does not depend on $\mathbf{x}$ and we have also dropped $\mathbf{v}$ in $p(\mathbf{y}|\mathbf{x}, \mathbf{v})$ since it does not affect $\mathbf{y}$ if $\mathbf{x}$ is known.

Under the assumption that all noise variables $\mathbf{w}_k$ and $\mathbf{n}_k$ for $k = 0 \ldots K$ are uncorrelated, we can use Bayes' rule to factor $p(\mathbf{y}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{v})$ as

$$p(\mathbf{y}|\mathbf{x}) = \prod_{k=0}^{K} p(\mathbf{y}_k|\mathbf{x}_k), \tag{2.47}$$

$$p(\mathbf{x}|\mathbf{v}) = p(\mathbf{x}_0|\check{\mathbf{x}}_0) \prod_{k=1}^{K} p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k). \tag{2.48}$$

We can simplify the optimization by working with the *log likelihood* rather than the original probability distributions, which has the effect of turning the products into sums:

$$\ln\left(p(\mathbf{y}|\mathbf{x})p(\mathbf{x}|\mathbf{v})\right) = \ln p(\mathbf{x}_0|\check{\mathbf{x}}_0) + \sum_{k=1}^{K} \ln p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k) + \sum_{k=0}^{K} \ln p(\mathbf{y}_k|\mathbf{x}_k). \tag{2.49}$$

If we further assume that the noise variables $\mathbf{w}_k$ and $\mathbf{n}_k$ follow zero-mean Gaussian distributions with covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively, then the right-hand-side terms of Equation (2.49) take the

form

$$\ln p(\mathbf{x}_0|\check{\mathbf{x}}_0) = -\frac{1}{2}(\mathbf{x}_0 - \check{\mathbf{x}}_0)^T \check{\mathbf{P}}_0^{-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0) - \underbrace{\frac{1}{2}\ln\left((2\pi)^N \det \check{\mathbf{P}}_0\right)}_{\text{independent of } \mathbf{x}} \tag{2.50}$$

$$\ln p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{v}_k) = -\frac{1}{2}(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{0}))^T \mathbf{Q}_k^{-1}(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{0})) - \underbrace{\frac{1}{2}\ln\left((2\pi)^N \det \mathbf{Q}_k\right)}_{\text{independent of } \mathbf{x}} \tag{2.51}$$

$$\ln p(\mathbf{y}_k|\mathbf{x}_k) = -\frac{1}{2}(\mathbf{y}_k - \mathbf{g}(\mathbf{x}_k, \mathbf{0}))^T \mathbf{R}_k^{-1}(\mathbf{y}_k - \mathbf{g}(\mathbf{x}_k, \mathbf{0})) - \underbrace{\frac{1}{2}\ln\left((2\pi)^M \det \mathbf{R}_k\right)}_{\text{independent of } \mathbf{x}} \tag{2.52}$$

Ignoring the terms that do not depend on $\mathbf{x}$, we can define the following quantities:

$$\mathcal{J}_{v,k}(\mathbf{x}) = \begin{cases} \frac{1}{2}(\mathbf{x}_0 - \check{\mathbf{x}}_0)^T \check{\mathbf{P}}_0^{-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0), & k = 0 \\ \frac{1}{2}(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{0}))^T \mathbf{Q}_k^{-1}(\mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{0})), & k = 1 \dots K \end{cases} \tag{2.53}$$

$$\mathcal{J}_{y,k}(\mathbf{x}) = \frac{1}{2}(\mathbf{y}_k - \mathbf{g}(\mathbf{x}_k, \mathbf{0}))^T \mathbf{R}_k^{-1}(\mathbf{y}_k - \mathbf{g}(\mathbf{x}_k, \mathbf{0})), \qquad k = 0 \dots K \tag{2.54}$$

which are all squared Mahalanobis distances. We can then define an overall *objective function* $\mathcal{J}(\mathbf{x})$ that we will seek to minimize with respect to the *design parameter* $\mathbf{x}$:

$$\mathcal{J}(\mathbf{x}) = \sum_{k=0}^{K} \left(\mathcal{J}_{v,k}(\mathbf{x}) + \mathcal{J}_{y,k}(\mathbf{x})\right) \tag{2.55}$$

Minimizing $\mathcal{J}(\mathbf{x})$ is equivalent to minimizing the negative log-likelihood of $\mathbf{x}$, which in turn is equivalent to maximizing the likelihood of $\mathbf{x}$ since the logarithm is a monotonically increasing function. The maximum likelihood estimate of $\mathbf{x}$ can therefore be obtained by solving the unconstrained optimization problem

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathcal{J}(\mathbf{x}) \tag{2.56}$$

which will yield the same solution for the best estimate $\hat{\mathbf{x}}$ as Equation (2.43).

2.1.2.3 Gauss-Newton Optimization

We can reformulate Equations (2.53) and (2.54) in terms of *errors* with respect to the prior information and measurements:

$$\mathcal{J}_{v,k}(\mathbf{x}) = \frac{1}{2}\mathbf{e}_{v,k}(\mathbf{x})^T \mathbf{W}_{v,k}^{-1}\mathbf{e}_{v,k}(\mathbf{x}) \tag{2.57}$$

$$\mathcal{J}_{y,k}(\mathbf{x}) = \frac{1}{2}\mathbf{e}_{y,k}(\mathbf{x})^T \mathbf{W}_{y,k}^{-1}\mathbf{e}_{y,k}(\mathbf{x}) \tag{2.58}$$

where

$$\mathbf{e}_{v,k}(\mathbf{x}) = \begin{cases} \mathbf{x}_0 - \check{\mathbf{x}}_0, & k = 0 \\ \mathbf{x}_k - \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{0}), & k = 1 \dots K \end{cases} \tag{2.59}$$

$$\mathbf{e}_{y,k}(\mathbf{x}) = \mathbf{y}_k - \mathbf{g}(\mathbf{x}_k, \mathbf{0}) \tag{2.60}$$

and $\mathbf{W}_{v,k}, \mathbf{W}_{y,k}$ are symmetric positive-definite weight matrices. We can further rewrite Equation (2.55) in lifted form[1] as

$$\mathcal{J}(\mathbf{x}) = \frac{1}{2}\mathbf{e}(\mathbf{x})^T \mathbf{W}^{-1} \mathbf{e}(\mathbf{x}) \tag{2.61}$$

where

$$\mathbf{e}(\mathbf{x}) = \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}) \\ \hline \mathbf{e}_{y,0}(\mathbf{x}) \\ \vdots \\ \mathbf{e}_{y,K}(\mathbf{x}) \end{bmatrix} \tag{2.62}$$

and

$$\mathbf{W} = \operatorname{diag}\left\{\mathbf{W}_{v,0}, \dots, \mathbf{W}_{v,K}, \mathbf{W}_{y,0}, \dots, \mathbf{W}_{y,K}\right\}. \tag{2.63}$$

Note that Equation (2.61) is quadratic in $\mathbf{e}(\mathbf{x})$, but not in $\mathbf{x}$ since our motion and measurement models may be nonlinear. While there are many possible algorithms for iteratively solving nonlinear least-squares problems such as this, a common technique that takes advantage of the quadratic form of the cost function is Newton's method or its derivatives, Gauss-Newton and Levenberg-Marquardt, which approximate Newton's method to avoid computing second-order derivatives. We can apply the Gauss-Newton optimization method to our batch nonlinear estimation problem by first approximating our error expressions using a first-order Taylor expansion about an operating point $\mathbf{x}_{\text{op}}$:

$$\mathbf{e}_{v,k}(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \approx \begin{cases} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) + \delta\mathbf{x}_0, & k = 0 \\ \mathbf{e}_{v,k}(\mathbf{x}_{\text{op}}) + \delta\mathbf{x}_k - \mathbf{F}_{k-1}\delta\mathbf{x}_{k-1}, & k = 1 \dots K \end{cases} \tag{2.64}$$

$$\mathbf{e}_{y,k}(\mathbf{x}_{\text{op}} + \delta\mathbf{x}) \approx \mathbf{e}_{y,k}(\mathbf{x}_{\text{op}}) - \mathbf{G}_k\delta\mathbf{x}_k, \qquad k = 0 \dots K, \tag{2.65}$$

where the Jacobians of the nonlinear motion and measurement models are given by

$$\mathbf{F}_{k-1} = \left.\frac{\partial \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{v}_k, \mathbf{w}_k)}{\partial \mathbf{x}_{k-1}}\right|_{\mathbf{x}_{\text{op},k-1}, \mathbf{v}_k, \mathbf{0}}, \tag{2.66}$$

$$\mathbf{G}_k = \left.\frac{\partial \mathbf{g}(\mathbf{x}_k, \mathbf{n}_k)}{\partial \mathbf{x}_k}\right|_{\mathbf{x}_{\text{op},k}, \mathbf{0}}. \tag{2.67}$$

[1] The term 'lifted' here refers to the fact that we are considering the entire trajectory at once.

If we make the common assumption that the noise terms in the motion and measurement models are additive, we can choose $\mathbf{W}_{v,0} = \check{\mathbf{P}}_0$, $\mathbf{W}_{v,k\in[1,K]} = \mathbf{Q}_k$ and $\mathbf{W}_{y,k} = \mathbf{R}_k$ to recover the maximum likelihood problem. Then we can define

$$\delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{x}_0 \\ \delta\mathbf{x}_1 \\ \delta\mathbf{x}_2 \\ \vdots \\ \delta\mathbf{x}_K \end{bmatrix} \tag{2.68}$$

$$\mathbf{H} = \left[\begin{array}{ccccccc} \mathbf{1} & & & & & & \\ -\mathbf{F}_0 & \mathbf{1} & & & & & \\ & -\mathbf{F}_1 & \ddots & & & & \\ & & \ddots & \mathbf{1} & & & \\ & & & -\mathbf{F}_{K-1} & \mathbf{1} & & \\ \hline -\mathbf{G}_0 & & & & & & \\ & -\mathbf{G}_1 & & & & & \\ & & -\mathbf{G}_2 & & & & \\ & & & & \ddots & & \\ & & & & & -\mathbf{G}_K & \end{array}\right] \tag{2.69}$$

$$\mathbf{e}(\mathbf{x}_{\text{op}}) = \begin{bmatrix} \mathbf{e}_{v,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{v,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{v,K}(\mathbf{x}_{\text{op}}) \\ \hline \mathbf{e}_{y,0}(\mathbf{x}_{\text{op}}) \\ \mathbf{e}_{y,1}(\mathbf{x}_{\text{op}}) \\ \vdots \\ \mathbf{e}_{y,K}(\mathbf{x}_{\text{op}}) \end{bmatrix} \in \mathbb{R}^{(K+1)(N+M)} \tag{2.70}$$

$$\mathbf{W} = \text{diag}\left\{\check{\mathbf{P}}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_K, \mathbf{R}_0, \mathbf{R}_1, \ldots, \mathbf{R}_K\right\}, \tag{2.71}$$

and compute the optimal Gauss-Newton update $\delta\mathbf{x}^*$ to the operating point by solving a linear system of equations:

$$\left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)\,\delta\mathbf{x}^* = \mathbf{H}^T\mathbf{W}^{-1}\mathbf{e}(\mathbf{x}_{\text{op}}). \tag{2.72}$$

For vector- or scalar-valued state variables, we update the operating point using

$$\mathbf{x}_{\text{op}} \leftarrow \mathbf{x}_{\text{op}} + \alpha\delta\mathbf{x}^*, \tag{2.73}$$

where $\alpha \in [0, 1]$ is a user-defined parameter controlling the step size along the update direction $\delta\mathbf{x}^*$, which is typically set using a line search. By providing a sufficiently good initial guess for $\mathbf{x}_{\text{op}}$ and iterating Equations (2.72) and (2.73) until convergence, we arrive at an optimal estimate for $\hat{\mathbf{x}}$.

It can be shown that the term $\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}$ in Equation (2.72) is precisely the *precision matrix* (i.e., the inverse covariance matrix) associated with a Gaussian estimate of $\delta\mathbf{x}$ whose mean is $\delta\mathbf{x}^*$. This allows us to judge our confidence in $\delta\mathbf{x}^*$, in a Bayesian sense, by computing the covariance matrix $\mathbf{P}^*$ associated with the update solution as

$$\mathbf{P}^* = \left(\mathbf{H}^T\mathbf{W}^{-1}\mathbf{H}\right)^{-1}. \tag{2.74}$$

2.1.2.4 Optimization on Matrix Lie Groups

We can adapt the Gauss-Newton optimization algorithm to handle rotations and poses without introducing constraints into the optimization problem. The key is to choose an appropriate perturbation scheme such that the operating point always remains in the group, while the update itself is unconstrained.

Rotations For rotations, we can use the exponential map to define an appropriate update rule by perturbing our operating point on the left by $\boldsymbol{\psi} \in \mathbb{R}^3$:

$$\mathbf{C}_{\text{op}} \leftarrow \exp\left(\boldsymbol{\psi}^\wedge\right)\mathbf{C}_{\text{op}}. \tag{2.75}$$

The goal of our Gauss-Newton optimization is then to find the optimal perturbation $\boldsymbol{\psi}^*$ such that iteratively modifying the operating point using this update rule minimizes the objective function.

For an arbitrary three-dimensional point $\mathbf{v} \in \mathbb{R}^3$, the Jacobian of the rotated point $\mathbf{Cv}$ with respect to $\boldsymbol{\psi}$ is given by

$$\frac{\partial(\mathbf{Cv})}{\partial\boldsymbol{\psi}} = -(\mathbf{Cv})^\wedge. \tag{2.76}$$

For a compound rotation $\mathbf{C} = \mathbf{C}_1\mathbf{C}_0$, we can consider the Jacobians with respect to perturbations in each rotation matrix, $\exp\left(\boldsymbol{\psi}^\wedge\right)\mathbf{C} = \exp\left(\boldsymbol{\psi}_1^\wedge\right)\mathbf{C}_1 \;\exp\left(\boldsymbol{\psi}_0^\wedge\right)\mathbf{C}_0$. These are given by

$$\frac{\partial\boldsymbol{\psi}}{\partial\boldsymbol{\psi}_1} = \mathbf{1} \qquad\qquad \frac{\partial\boldsymbol{\psi}}{\partial\boldsymbol{\psi}_0} = \text{Ad}\left(\mathbf{C}_1\right) = \mathbf{C}_1. \tag{2.77}$$

Poses Similarly for poses, we can use the exponential map to define an appropriate update rule by perturbing our operating point on the left by $\boldsymbol{\epsilon} \in \mathbb{R}^6$:

$$\mathbf{T}_{\text{op}} \leftarrow \exp\left(\boldsymbol{\epsilon}^\wedge\right)\mathbf{T}_{\text{op}}. \tag{2.78}$$

Again, our goal is now to find the optimal perturbation $\boldsymbol{\epsilon}^*$ such that iteratively modifying the operating point using this update rule minimizes the objective function.

For an arbitrary three-dimensional point $\mathbf{p} \in \mathbb{R}^4$ expressed in homogeneous coordinates, the Jacobian of the transformed point $\mathbf{Tp}$ with respect to $\boldsymbol{\epsilon}$ is given by

$$\frac{\partial(\mathbf{Tp})}{\partial\boldsymbol{\epsilon}} = (\mathbf{Tp})^\odot, \tag{2.79}$$

where

$$\mathbf{p}^{\odot} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}^{\odot} = \begin{bmatrix} \eta\mathbf{1} & -\boldsymbol{\varepsilon}^{\wedge} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{4\times 6}, \tag{2.80}$$

with $\boldsymbol{\varepsilon} \in \mathbb{R}^3$ and $\eta \in \mathbb{R}$. In most cases $\eta = 1$ and we are interested only in the 3×6 Jacobian of the Cartesian coordinates $\boldsymbol{\varepsilon}$ of $\mathbf{p}$ with respect to $\boldsymbol{\epsilon}$.

For a compound pose $\mathbf{T} = \mathbf{T}_1\mathbf{T}_0$, we can consider the Jacobians with respect to perturbations in each transformation matrix, $\exp(\boldsymbol{\epsilon}^{\wedge})\,\mathbf{T} = \exp(\boldsymbol{\epsilon}_1^{\wedge})\,\mathbf{T}_1 \;\exp(\boldsymbol{\epsilon}_0^{\wedge})\,\mathbf{T}_0$. These are given by

$$\frac{\partial \boldsymbol{\epsilon}}{\partial \boldsymbol{\epsilon}_1} = \mathbf{1} \qquad\qquad \frac{\partial \boldsymbol{\epsilon}}{\partial \boldsymbol{\epsilon}_0} = \mathrm{Ad}(\mathbf{T}_1), \tag{2.81}$$

with $\mathrm{Ad}(\mathbf{T}_1)$ given by Equation (2.34).

2.1.2.5 Robust Estimation

It is often the case in robotics that the set of measurements used to define our nonlinear least-squares problem contains a number of *outliers*, which do not originate from the same data-generating process as the rest of the measurements (i.e., the *inliers*). Outlier measurements are often produced by sensor artifacts, incorrect data association, or other effects, and can cause least-squares estimation to become biased. This is because outliers tend to produce large measurement errors, which are given proportionately more weight in the estimator, thereby causing the solution to be 'dragged' towards the outliers.

There are a number of techniques for mitigating the effect of outliers, which can be thought of as proactive or reactive. Proactive approaches such as Random Sample Consensus or RANSAC (Fischler and Bolles, 1981) focus on removing outliers from the measurement set before setting up the least-squares problem. In contrast, reactive approaches attempt to minimize the influence of outliers in the measurement set by modifying the least-squares objective function. This is referred to as *robust estimation*.

M-Estimation A popular approach to robust estimation is *M-estimation*, or maximum likelihood-type estimation, which generalizes the maximum likelihood least-squares problem (Huber et al., 1981). To use M-estimation in our problem, we generalize the objective function to be of the form

$$\mathcal{J}(\mathbf{x}) = \rho(\mathbf{e}(\mathbf{x})) \tag{2.82}$$

where $\rho(\cdot)$ is referred to as a *M-estimator* or *robust loss function*. There are many possible choices for $\rho(\cdot)$, of which the squared error is one. Choosing $\rho(\mathbf{e}(\mathbf{x})) = \frac{1}{2}\mathbf{e}(\mathbf{x})^T\mathbf{e}(\mathbf{x})$ recovers the original least-squares formulation of Equation (2.61), where we have made the substitution $\mathbf{e}(\mathbf{x}) \leftarrow \mathbf{W}^{-\frac{1}{2}}\mathbf{e}(\mathbf{x})$.[2]

A common choice for $\rho(\cdot)$ is the Huber loss, which behaves like the squared error near zero and like the absolute error far from zero:

$$\rho(e;\kappa) = \begin{cases} \frac{1}{2}e^2 & |e| \le \kappa \\ \kappa\left(|e| - \frac{\kappa}{2}\right) & |e| \ge \kappa, \end{cases} \tag{2.83}$$

[2] $\mathbf{W}^{-\frac{1}{2}}$ is sometimes referred to as the *stiffness matrix*. We know $\mathbf{W}^{-\frac{1}{2}}$ exists because $\mathbf{W}$ is symmetric and positive-definite by construction (cf. Equation (2.71)).

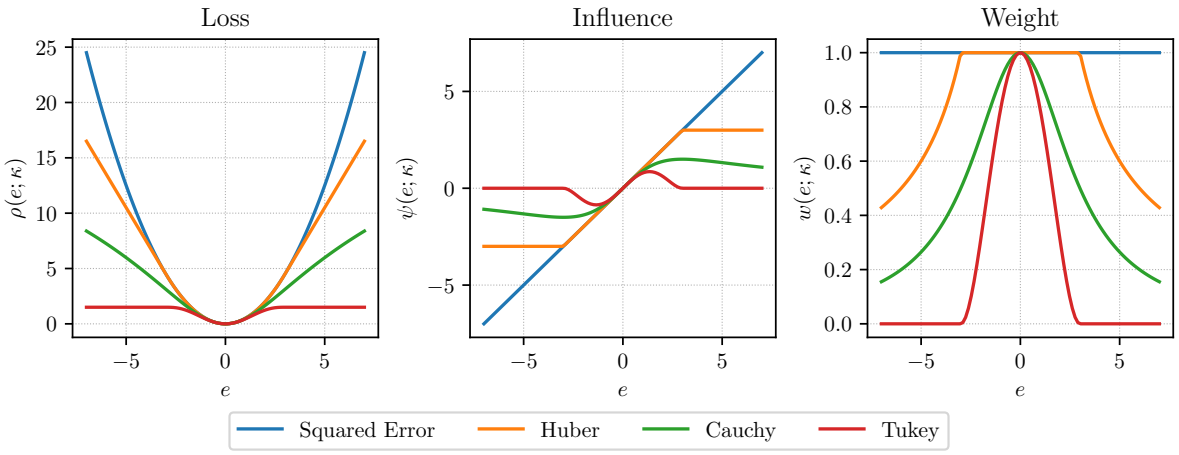


Figure 2.1: Comparison of squared error, Huber, Cauchy, and Tukey robust loss functions ($\kappa = 3$).

where κ is a tunable parameter and $\rho(\mathbf{e}(\mathbf{x})) = \sum_i \rho(e_i;\kappa)$ with e_i the components of $\mathbf{e}(\mathbf{x})$. Other commonly used M-estimators include the Cauchy loss, given by

$$\rho(e;\kappa) = \frac{\kappa^2}{2}\log\left(1 + \left(\frac{e}{\kappa}\right)^2\right), \tag{2.84}$$

and the Tukey loss, given by

$$\rho(e;\kappa) = \begin{cases} \frac{\kappa^2}{6}\left(1 - \left(1 - \left(\frac{e}{\kappa}\right)^2\right)^3\right) & |e| \leq \kappa \\ \frac{\kappa^2}{6} & |e| \geq \kappa. \end{cases} \tag{2.85}$$

Figure 2.1 illustrates the Huber, Cauchy, and Tukey losses for the case $\kappa = 3$. The intuition behind all three functions is that large measurement errors, which are likely induced by outliers, should contribute proportionately less to the objective function than small measurement errors, which likely correspond to inliers. The parameter κ controls the extent of the region in which inlier measurement errors are expected to fall.

Iteratively Re-weighted Least Squares While there are closed-form solutions to the linear least-squares problem, no closed-form solution exists for robust losses such as the Huber, Cauchy, and Tukey losses described above. In order to solve robust estimation problems using M-estimators, we can rely on the technique of Iteratively Re-weighted Least Squares (IRLS). IRLS operates by assigning a weight w_i to each scalar error term e_i, such that $\rho(\mathbf{e}(\mathbf{x}))$ can be minimized iteratively by updating w_i and solving a linear least-squares problem at each iteration. One of the main advantages of IRLS is that it can be easily used with Gauss-Newton optimization to solve nonlinear M-estimation problems.

We can derive the values of the weights w_i by making the identification

$$\mathcal{J}(\mathbf{x}) = \rho(\mathbf{e}(\mathbf{x})) = \sum_i \rho(e_i) = \frac{1}{2}\sum_i w_i e_i^2. \tag{2.86}$$

Differentiating with respect to the design parameter $\mathbf{x}$ and setting to zero for an optimum, we obtain

$$\frac{\partial \mathcal{J}}{\partial \mathbf{x}} = \sum_i \frac{\partial \rho(e_i)}{\partial e_i} \frac{\partial e_i}{\partial \mathbf{x}} = \sum_i w_i e_i \frac{\partial e_i}{\partial \mathbf{x}} = \mathbf{0}. \tag{2.87}$$

Identifying terms in the two sums, we see that the weights must be given by

$$w_i = \frac{1}{e_i} \frac{\partial \rho(e_i)}{\partial e_i} = \frac{1}{e_i} \psi(e_i). \tag{2.88}$$

The function $\psi(e_i) = \dfrac{\partial \rho(e_i)}{\partial e_i}$ is often referred to as the *influence function* because it describes the rate of growth of the objective function with respect to the measurement errors. Figure 2.1 illustrates the influence and weight functions associated with the Huber, Cauchy, and Tukey robust loss functions.

2.2 Visual Mapping and Localization

Passive cameras are an appealing sensor choice for many robotics applications because they are inexpensive, widely available, and can be used to infer both the motion of a robot and the geometry of the operating environment. This section provides an overview of the visual mapping and localization pipelines used in this thesis, beginning with a discussion of common camera models used in robotics and concluding with an overview of feature-based and photometric approaches to mapping and localization using cameras.

2.2.1 Camera Models

We first discuss sensor models associated with several classes of camera commonly used in robotics. Section 2.2.1.1 discusses the basic frontal projection model for a single (monocular) perspective camera, while Section 2.2.1.2 extends this model to the two-camera fronto-parallel stereo case, and Section 2.2.1.3 describes the model associated with RGB-D sensors composed of a monocular camera and a depth sensor.

2.2.1.1 Monocular Camera

Figure 2.2 illustrates the basic frontal projection model for a single (monocular) perspective camera, which is based on an idealized pinhole camera and is commonly used in computer vision. The sensor frame $\underrightarrow{\mathcal{F}}_s$ is defined such that its z-axis coincides with the *optical axis* of the camera, which is orthogonal to the image plane and intersects it at the *principal point* (c_u, c_v). The origin of $\underrightarrow{\mathcal{F}}_s$ is located at a focal distance f away from the image plane along the optical axis. Image coordinates (u, v) are typically measured from the top-left corner of the image plane, although other conventions exist.

Forward Measurement Model Ignoring lens distortion, which is typically compensated for in a pre-processing step using an appropriate calibration, we can establish the following relationship between the coordinates of a 3D point $\mathbf{p} = [\, x \quad y \quad z \,]^T$ in the sensor frame and its corresponding 2D projection

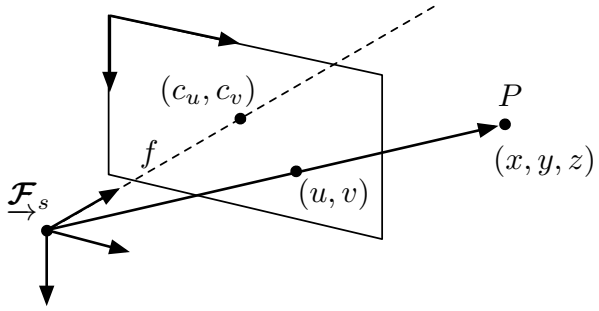


Figure 2.2: Frontal projection model for a perspective camera (Barfoot, 2017).

onto the image plane $\mathbf{y} = \begin{bmatrix} u & v \end{bmatrix}^T$:

$$\mathbf{y} = \mathbf{g}(\mathbf{p}) = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{\mathbf{P}} \underbrace{\begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \frac{1}{z}\mathbf{p} \tag{2.89}$$

where $\mathbf{K}$ is the *intrinsic parameter matrix* of the camera containing the camera focal length, expressed in units of horizontal pixels f_u and vertical pixels f_v, and the principal point (c_u, c_v), while $\mathbf{P}$ is a projection matrix that removes the bottom row from the homogeneous point representation. From Equation (2.89) we can clearly see that the forward measurement model for a monocular camera is not invertible due to the loss of information through the projective division by the depth z. Put another way, any 3D point situated along the ray passing through the origin of $\mathcal{F}_s$ and the point $\mathbf{p}$ will map onto to the same $\mathbf{y}$ in image space. In order to invert the measurement model, it is necessary to determine the depth z of the point in 3D space, which can be done using the two views of a stereo camera or a dedicated depth sensor as discussed in the following sections.

Jacobians For optimization tasks such as pose estimation, it is often necessary to derive the Jacobians of our measurement models. The Jacobian of the forward measurement model for a monocular camera is given by

$$\frac{\partial \mathbf{g}(\mathbf{p})}{\partial \mathbf{p}} = \begin{bmatrix} f_u/z & 0 & -f_u x/z^2 \\ 0 & f_v/z & -f_v y/z^2 \end{bmatrix} \tag{2.90}$$

2.2.1.2 Stereo Camera

Three-dimensional information is commonly recovered from perspective cameras by combining observations from multiple views. A typical technique for constructing such a three-dimensional vision sensor is to combine two perspective cameras into a stereo camera, where the two cameras are rigidly connected to one another and the transformation between them is known. One of the most common stereo configurations, the *fronto-parallel* configuration, is shown in Figure 2.3. In this configuration, the two cameras are aligned such that corresponding coordinate axes are parallel to each other, and the pair is separated by a known *baseline* b along the x-axis. Note that in this model we have chosen to identify the sensor frame $\mathcal{F}_s$ with the frame of the left camera $\mathcal{F}_\ell$, however other conventions exist as well.

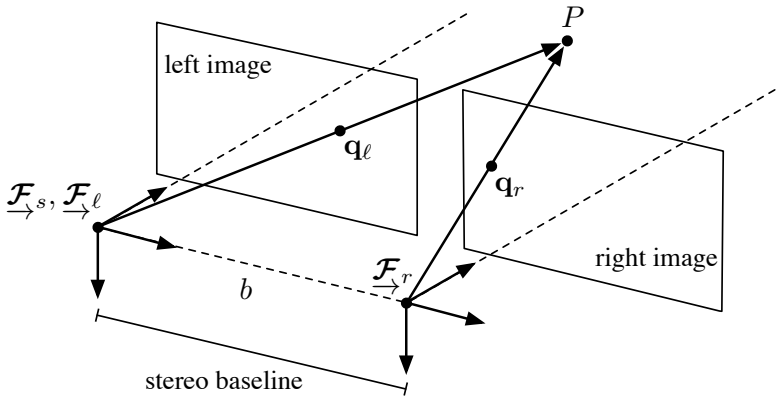


Figure 2.3: Stereo camera model with the sensor frame located at the left camera (Barfoot, 2017).

Forward Measurement Model The forward measurement model for a stereo camera is given by

$$\mathbf{y} = \begin{bmatrix} \mathbf{q}_\ell \\ \mathbf{q}_r \end{bmatrix} = \begin{bmatrix} u_\ell \\ v_\ell \\ u_r \\ v_r \end{bmatrix} = \mathbf{g}(\mathbf{p}) = \underbrace{\begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u b \\ 0 & f_v & c_v & 0 \end{bmatrix}}_{\mathbf{M}} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.91}$$

which relates the coordinates of $\mathbf{p}$ to the corresponding image coordinates in the left image (u_ℓ, v_ℓ) and the right image (u_r, v_r). Often we drop the equation for v_r and replace the equation for u_r with one for the *disparity*, which is given by

$$d = u_\ell - u_r = \frac{1}{z} f_u b \tag{2.92}$$

Then we can rewrite Equation (2.91) as

$$\mathbf{y} = \begin{bmatrix} u \\ v \\ d \end{bmatrix} = \mathbf{g}(\mathbf{p}) = \underbrace{\begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 0 & f_u b \end{bmatrix}}_{\mathbf{M}} \frac{1}{z} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.93}$$

Inverse Measurement Model Unlike the measurement model for the single perspective camera, the stereo camera measurement model is invertible and we can recover three-dimensional information from a single measurement. The inverse models are given by

$$\mathbf{p} = \mathbf{g}^{-1}(\mathbf{y}) = \frac{b}{u_\ell - u_r} \begin{bmatrix} u_\ell - c_u \\ (\overline{v} - c_v)(f_u / f_v) \\ f_u \end{bmatrix} \qquad \text{for Equation (2.91)} \tag{2.94}$$

$$\mathbf{p} = \mathbf{g}^{-1}(\mathbf{y}) = \frac{b}{d} \begin{bmatrix} u - c_u \\ (v - c_v)(f_u / f_v) \\ f_u \end{bmatrix} \qquad \text{for Equation (2.93)} \tag{2.95}$$

where $\overline{v} = \frac{1}{2}(v_\ell + v_r)$.

Jacobians The Jacobians of the forward measurement models are given by

$$\frac{\partial \mathbf{g}(\mathbf{p})}{\partial \mathbf{p}} = \frac{1}{z}\begin{bmatrix} f_u & 0 & -f_u x/z \\ 0 & f_y & -f_v y/z \\ f_u & 0 & -f_u(x-b)/z \\ 0 & f_y & -f_v y/z \end{bmatrix} \quad \text{for Equation (2.91)} \tag{2.96}$$

$$\frac{\partial \mathbf{g}(\mathbf{p})}{\partial \mathbf{p}} = \frac{1}{z}\begin{bmatrix} f_u & 0 & -f_u x/z \\ 0 & f_v & -f_v y/z \\ 0 & 0 & -f_u b/z \end{bmatrix} \quad \text{for Equation (2.93)} \tag{2.97}$$

For the inverse measurement model Jacobians we have

$$\frac{\partial \mathbf{g}^{-1}(\mathbf{y})}{\partial \mathbf{y}} = \frac{b}{(u_\ell - u_r)^2}\begin{bmatrix} -(u_r - c_u) & 0 & u_\ell - c_u & 0 \\ -(\overline{v} - c_v)(f_u/f_v) & \frac{1}{2}(u_\ell - u_r)(f_u/f_v) & (\overline{v} - c_v)(f_u/f_v) & \frac{1}{2}(u_\ell - u_r)(f_u/f_v) \\ -f_u & 0 & f_u & 0 \end{bmatrix} \tag{2.98}$$

for Equation (2.94), and

$$\frac{\partial \mathbf{g}^{-1}(\mathbf{y})}{\partial \mathbf{y}} = \frac{b}{d}\begin{bmatrix} 1 & 0 & -(u - c_u)/d \\ 0 & f_u/f_v & -(v - c_v)(f_u/f_v)/d \\ 0 & 0 & -f_u/d \end{bmatrix} \tag{2.99}$$

for Equation (2.95).

2.2.1.3 Depth Camera

Depth cameras have become increasingly prevalent in robotics as inexpensive sources of three-dimensional vision data. In contrast to stereo cameras, which are passive and require solving a data association problem to obtain depth information, depth cameras operate on an active sensing principle: by projecting infrared light onto a scene and analyzing the return signal, a *depth image* can be computed and associated with an image from a standard perspective camera. This is typically accomplished using a structured projection pattern or time-of-flight ranging techniques. The combination of a depth camera and RGB (red-green-blue) colour camera is referred to as an RGB-D sensor. RGB-D sensors are most commonly used for indoor robotics applications due to their limited sensing range and sensitivity to environmental sources of infrared light such as the sun.

Forward Measurement Model The forward measurement model for an RGB-D sensor mirrors the model for the monocular perspective camera (cf. Equation (2.89)) with the exception that the depth z

is preserved in the measurement:

$$\mathbf{y} = \begin{bmatrix} u \\ v \\ z \end{bmatrix} = \mathbf{g}(\mathbf{p}) = \begin{bmatrix} f_u x/z + c_u \\ f_v y/z + c_v \\ z \end{bmatrix} \tag{2.100}$$

Inverse Measurement Model Since the depth z is known directly from the depth camera, the inverse measurement model is straightforward to derive:

$$\mathbf{p} = \mathbf{g}^{-1}(\mathbf{y}) = \begin{bmatrix} (u - c_u)z/f_u \\ (v - c_v)z/f_v \\ z \end{bmatrix} \tag{2.101}$$

Jacobians The Jacobians of the forward and inverse measurement models are given by

$$\frac{\partial \mathbf{g}(\mathbf{p})}{\partial \mathbf{p}} = \begin{bmatrix} f_u x/z & 0 & -f_u x/z^2 \\ 0 & f_v y/z & -f_v y/z^2 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.102}$$

$$\frac{\partial \mathbf{g}^{-1}(\mathbf{y})}{\partial \mathbf{y}} = \begin{bmatrix} z/f_u & 0 & (u - c_u)/f_u \\ 0 & z/f_v & (v - c_v)/f_v \\ 0 & 0 & 1 \end{bmatrix} \tag{2.103}$$

2.2.2 Feature-based (Indirect) Methods

Many visual mapping and localization pipelines used in robotics applications are based on the detection and tracking of salient *features* or *keypoints* to establish correspondences between images. Collectively, these are referred to as *feature-based* or *indirect* methods. There are a wide variety of feature types in use in modern robotics applications, with popular choices being SIFT (Lowe, 2004), SURF (Bay et al., 2008), ORB (Rublee et al., 2011), and `libviso2` features (Geiger et al., 2011), among others. All of these methods have in common three basic operations: i) *feature detection*, usually as a response to convolutional corner or edge detection filters; ii) *feature description*, which summarizes the local appearance of the image in a neighbourhood around each feature point, typically as a high-dimensional feature vector; and iii) *feature matching*, which is usually accomplished by comparing feature descriptors using a distance metric such as Euclidean distance or cosine similarity for vector representations.

Figure 2.4 illustrates the major processing blocks in a common feature-based mapping and localization pipeline, originally developed by Moravec (1980) and later extended to stereo vision by Matthies (1989). Broadly speaking, the pipeline can be separated into two stages: image processing, in which features are detected and matched to establish a set of point correspondences; and pose estimation, in which the vehicle state is estimated using nonlinear optimization.

2.2.2.1 Image Processing

The pipeline begins by pre-processing images from the left and right cameras to remove the effects of lens distortion and correct for minor camera misalignment using an appropriate camera calibration. This process is referred to as rectification, and the resulting stereo images correspond to a fronto-parallel view. These 'rectified' images are further processed by a feature detector, which produces two sets of

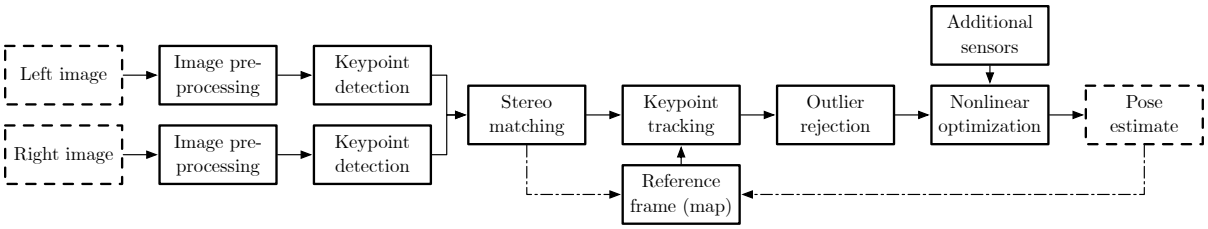


Figure 2.4: Processing blocks in a feature-based (indirect) stereo mapping and localization pipeline.

two-dimensional keypoints in the left and right images. These 2D keypoints are matched together to obtain a set of stereo point correspondences, which can be used to compute the position of environmental landmarks in three dimensions, relative to the camera frame. This set of three-dimensional landmarks can then be tracked against landmarks stored in a map, if one exists, or against landmarks in a previous frame in the case of visual odometry.

2.2.2.2 Pose Estimation

The set of temporal keypoint matches can be used to determine the relative pose of the robot by solving a *bundle adjustment* problem using the nonlinear optimization tools described in Section 2.1.2. We can simultaneously reject spurious matches and obtain an initial guess for the nonlinear optimization using the three-point method of Horn (1987) and Umeyama (1991) within the random sample consensus (RANSAC) algorithm developed by Fischler and Bolles (1981).

The optimization problem to be solved in a feature-based localization pipeline is typically expressed in terms of the *reprojection error* of keypoints in the reference image or map when projected onto the live or tracking image using the current estimate of the relative pose between them:

$$\hat{\mathbf{T}}_{tr} = \underset{\mathbf{T}_{tr}}{\operatorname{argmin}} \sum_{j=0}^{J} \mathbf{e}_j^T \mathbf{R}_j^{-1} \mathbf{e}_j, \tag{2.104}$$

where $\mathbf{e}_j$ is the reprojection error for keypoint j given by

$$\mathbf{e}_j = \mathbf{g}(\mathbf{T}_{tr}\mathbf{g}^{-1}(\mathbf{y}_{r,j})) - \mathbf{y}_{t,j}, \tag{2.105}$$

$\mathbf{R}_j$ is the covariance matrix of the measurement noise at timestep j, $\mathbf{T}_{tr}$ represents the relative pose between the reference frame $\underrightarrow{\mathcal{F}}_r$ and tracking frame $\underrightarrow{\mathcal{F}}_t$, and $\mathbf{g}(\cdot)$ is the stereo camera measurement model (or any invertible camera model). In practice, $\mathbf{R}_j$ is often tuned to a constant value that performs well for the target application. Note that Equation (2.104) can be easily adapted to include data terms from additional sensors and/or to simultaneously estimate multiple poses in, for example, a sliding window bundle adjustment. The 3D coordinates of keypoints in the map may also be included in the state and simultaneously optimized along with the camera poses, however in our applications we are typically more interested in estimating the motion of the vehicle than the precise shape of the world, and the additional computational cost of optimizing keypoint positions may not be worthwhile.

Assuming that the nonlinear optimization converges, we are left with a set of stereo keypoints and the maximum likelihood estimate of the vehicle pose from which they were observed. At this stage, we can integrate these observations into the map. This is commonly done by constructing a *pose*

graph consisting of vehicle states or *keyframes* as nodes connected by vertices encoding the relative transformations between them. In many modern systems (e.g., PTAM (Klein and Murray, 2007), ORB-SLAM (Mur-Artal et al., 2015) and LSD-SLAM (Engel et al., 2014)) the pose graph optimization is carried out as a background task at a lower rate than the foreground tracking procedure. However, for many navigation tasks such as autonomous route following (Furgale and Barfoot, 2010), a globally consistent map is not necessary and the pose graph optimization can be neglected or carried out in a limited window around the vehicle's current pose.

2.2.3 Photometric (Direct) Methods

In contrast to feature-based methods, *direct* or *photometric* methods do not rely on feature detection and matching but operate directly on pixel intensities, minimizing a *photometric error* rather than a reprojection error. Two companion papers by Irani and Anandan (2000) and Torr and Zisserman (2000) summarize the key distinction between direct and feature-based methods: direct methods *simultaneously* estimate the motion of the camera and the correspondence of every pixel, whereas feature-based methods separate the two problems, first finding correspondences and then using those correspondences to estimate the motion of the camera. While feature-based methods have been the dominant paradigm for the past 20 years, largely due to their relative computational efficiency and the invariance of feature descriptors to certain types of limited appearance change, direct methods have enjoyed a resurgence in popularity since 2010 with Newcombe et al. (2011), Engel et al. (2014), Forster et al. (2014) and Omari et al. (2015), among others, demonstrating relatively robust and computationally efficient approaches to direct mapping and localization.

Figure 2.5 illustrates the major processing blocks in a generic optimization-based photometric mapping and localization pipeline analogous to the feature-based pipeline depicted in Figure 2.4.

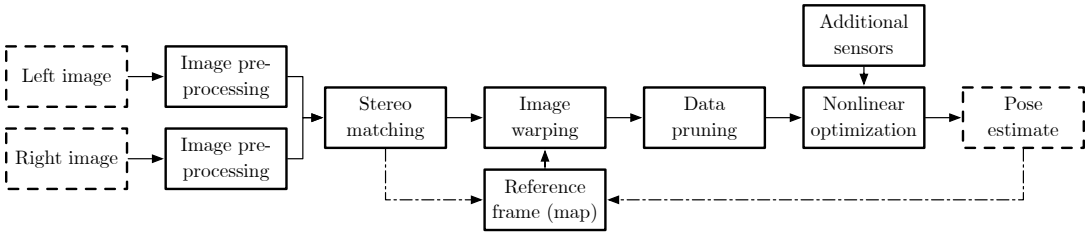


Figure 2.5: Processing blocks in a photometric (direct) stereo mapping and localization pipeline.

2.2.3.1 Image Processing

Similarly to the feature-based pipeline, the photometric pipeline begins by un-distorting and rectifying the stereo images to convert them to a fronto-parallel view. We can obtain a dense *disparity map* for the entire image using a stereo block matching procedure.[3] Stereo block matching operates by comparing windows of pixels along each row in the rectified stereo images and finds correspondences between pixels by optimizing a measure of similarity such as normalized cross-correlation (NCC), sum of absolute differences (SAD) or sum of squared distances (SSD).

[3]In practice, there are typically pixels for which a reliable disparity value cannot be computed, often due to depth discontinuities and occlusions.

2.2.3.2 Pose Estimation

Pose estimation in direct methods conventionally relies on the *photometric consistency* or *brightness constancy* assumption, which states that pixels corresponding to the same point in space should have the same intensity value over time. Given a tracking image $\mathbf{I}_t$, a reference image $\mathbf{I}_r$ and a disparity map $\mathbf{D}_t$ or $\mathbf{D}_r$ corresponding to either image, we can iteratively solve for the relative pose of the camera by minimizing a *photometric error* for each correspondence. For reasons of computational efficiency, we often assume that the disparity map is associated with the reference image or keyframe, and compute the photometric error as follows.

For notational convenience, we decompose our stereo measurement $\mathbf{y}$ as

$$\mathbf{y} = \begin{bmatrix} \mathbf{u} \\ \mathbf{D}(\mathbf{u}) \end{bmatrix} \tag{2.106}$$

where $\mathbf{u} = [u\ v]^T$ denotes image coordinates and $d = \mathbf{D}(\mathbf{u})$ the value of the disparity map at $\mathbf{u}$.

We first map image coordinates $\mathbf{u}_r$ in the reference image onto their corresponding image coordinates $\mathbf{u}'_t$ in the tracking image using the camera model and the current estimate of the relative transformation:

$$\mathbf{y}'_t = \begin{bmatrix} \mathbf{u}'_t \\ \mathbf{D}'_t(\mathbf{u}') \end{bmatrix} = \mathbf{g}(\mathbf{T}_{tr}\mathbf{g}^{-1}(\mathbf{y}_r)) \tag{2.107}$$

where the notation $(\cdot)'$ denotes that the quantity has been computed by applying a reprojection operation, which is sometimes referred to as a warping operation. We can then compute a reconstruction $\mathbf{I}'_r$ of the reference image by sampling the tracking image at the reprojected image coordinates $\mathbf{u}'_t$:

$$\mathbf{I}'_r(\mathbf{u}_r) = \mathbf{I}_t(\mathbf{u}'_t). \tag{2.108}$$

This image reconstruction procedure is referred to as *inverse compositional* warping because the reference image is reconstructed by sampling intensity values from the tracking image at the reprojected coordinates in the tracking image (Baker and Matthews, 2004). In contrast, a *forward compositional* warping operation would reconstruct the tracking image by transferring pixel values from the reference image to the reprojected coordinates in the tracking image. Forward compositional warping suffers from collisions where multiple pixels in the reference image map onto a single pixel in the tracking image (e.g., due to occlusions induced by perspective shift), which must be resolved using depth buffering to correctly predict the observed intensity value. Inverse compositional warping naturally resolves these collisions by reframing this many-to-one mapping of pixels as a one-to-many mapping.

After reconstructing the reference image by warping the tracking image, the photometric error at each pixel can then be computed as

$$e_{\mathbf{u}} = \mathbf{I}'_r(\mathbf{u}) - \mathbf{I}_r(\mathbf{u}) \tag{2.109}$$

and the optimization problem to be solved can be formulated as

$$\hat{\mathbf{T}}_{tr} = \underset{\mathbf{T}_{tr}}{\operatorname{argmin}} \sum_{\mathbf{u}\in\boldsymbol{\Omega}} \frac{1}{\sigma^2_{\mathbf{u}}} e^2_{\mathbf{u}} \tag{2.110}$$

where $\mathbf{\Omega}$ is the domain of valid pixels and $\sigma_{\mathbf{u}}^2$ is the variance of the photometric errors. As $\sigma_{\mathbf{u}}^2$ encapsulates the noise properties of both the imaging sensor and the disparity map computation, it varies per pixel and can be estimated as

$$\sigma_{\mathbf{u}}^2 = \sigma_{\mathbf{I},\mathbf{u}}^2 + \left(\frac{\partial e_{\mathbf{u}}}{\partial \mathbf{D}_r} \bigg|_{\mathbf{u}} \right)^2 \sigma_{\mathbf{D},\mathbf{u}}^2. \tag{2.111}$$

Note that in order to carry out the optimization in Equation (2.110), we require the Jacobians of the images with respect to the image coordinates. These are typically obtained as *gradient images* via smoothing and finite differencing using separable convolutional Sobel-Feldman operators,

$$\frac{\partial \mathbf{I}}{\partial u} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \mathbf{I} \qquad \frac{\partial \mathbf{I}}{\partial v} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \mathbf{I}, \tag{2.112}$$

which compute an approximation to the image derivative with smoothing.[4]

Direct methods can achieve localization accuracy on par with state of the art feature-based methods while providing dense geometric information at no additional computational cost, and enjoying a degree of robustness to common failure cases of feature-based methods such as camera defocus and motion blur. However, they are more sensitive to initialization, computationally more expensive, and do not lend themselves well to joint optimization of motion and scene structure. These disadvantages can be mitigated in several ways. First, we can reduce the algorithm's computational cost and sensitivity to initialization by using a coarse-to-fine optimization scheme in which we iteratively re-solve Equation (2.110) at multiple scales in a Gaussian image pyramid. We traverse the pyramid from lowest to highest resolution, using the solution at each scale as the initial guess to the next. Second, we can use a *data pruning* scheme to reduce the dimensionality of the measurement error. In practice, this takes the form of discarding pixels that project to invalid image coordinates (i.e., outside the field of view of the camera) and pixels without disparity information, as well as pixels with low gradient magnitude as these contribute less information. Finally, if we assume that the reconstructed reference image is sufficiently similar to the actual reference image, we can make the approximations

$$\frac{\partial \mathbf{I}'_r}{\partial u} \approx \frac{\partial \mathbf{I}_r}{\partial u} \qquad \text{and} \qquad \frac{\partial \mathbf{I}'_r}{\partial v} \approx \frac{\partial \mathbf{I}_r}{\partial v}, \tag{2.113}$$

which allows us to compute the image gradients for keyframes only.

2.3 Deep Learning

Mobile robots collect and store massive amounts of sensor data which can, in principle, be analyzed and used to improve the performance of our autonomy systems. Data-driven approaches are particularly valuable when the models used to design algorithms for perception, planning and control do not describe the environment or robot with sufficient fidelity for the target application. Over the past several years, machine learning, and especially deep learning (LeCun et al., 2015), has become an increasingly important tool for leveraging large datasets to expand the capabilities of robots beyond the limitations

[4] There exist a number of alternatives to the Sobel-Feldman operator for differentiating discrete multi-dimensional signals, however these are beyond the scope of this thesis and we do not discuss them here.

of classical model-based algorithms. This section provides a brief overview of core concepts in deep learning as it is used in this thesis. The interested reader may refer to Goodfellow et al. (2016) for a more complete discussion of deep learning theory and practice.

2.3.1 Deep Feedforward Networks

Deep feedforward networks (also referred to as feedforward neural networks) are an extremely common type of deep learning model, where the goal is to approximate an arbitrary function $\mathbf{f}^* : \mathbb{R}^M \to \mathbb{R}^N$ by defining a parameterized mapping $\mathbf{y} = \mathbf{f}(\mathbf{x}; \boldsymbol{\theta})$ and learning from data the values of the parameters $\boldsymbol{\theta}$ that yield the 'best' approximation of $\mathbf{f}^*$. Feedforward networks are called *feedforward* because information flows from the input $\mathbf{x} \in \mathbb{R}^M$ through intermediate computations to the output $\mathbf{y} \in \mathbb{R}^N$, without any feedback connections through which the model may operate on its own outputs. Feedforward networks are called *networks* because they are designed by chaining together several simpler functions; the *depth* of the network is determined by the length of the chain. For example, a network of depth four might consist of three functions composed together so that $\mathbf{y} = \mathbf{f}_3 \circ \mathbf{f}_2 \circ \mathbf{f}_1(\mathbf{x})$. The four layers are the *input layer* $\mathbf{x}$, the *output layer* $\mathbf{y}$ and the two intermediate *hidden layers* $\mathbf{h}_1 = \mathbf{f}_1(\mathbf{x})$ and $\mathbf{h}_2 = \mathbf{f}_2(\mathbf{h}_1)$.

During training, we are typically provided with training examples $(\mathbf{x}_i, \mathbf{y}_i^*)$ representing noisy samples $\mathbf{y}_i^*$ from $\mathbf{f}^*$ evaluated at different points $\mathbf{x}_i$. The goal of the training procedure is to drive $\mathbf{f}$ to match $\mathbf{f}^*$ by manipulating the parameters $\boldsymbol{\theta}$ such that the output layer gives $\mathbf{y}_i \approx \mathbf{y}_i^*$, $\forall i$. Note that only the behaviour of the output layer is prescribed by the training data; the learning algorithm is free to choose how to use the hidden layers to best approximate $\mathbf{f}^*$. It is for this reason that the intermediate layers of the network are referred to as "hidden" layers. Hidden layers are typically vector-valued even if the final output of the network is scalar-valued; the dimensionality of the widest hidden layer determines the model's *width*.

A deep feedforward network can be represented as a directed acyclic graph describing how information flows through the layers of the network. Figure 2.6 shows the graph corresponding to our example four-layer network. Nodes correspond to input layer $\mathbf{x}$, output layer $\mathbf{y}$, and hidden layers $\mathbf{h}_i$, while edges correspond to the operations $\mathbf{f}_i$ mapping each layer onto the next.

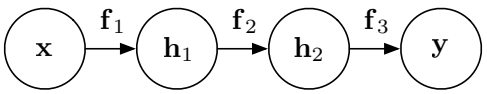


Figure 2.6: Directed acyclic graph describing the mapping $\mathbf{y} = \mathbf{f}_3 \circ \mathbf{f}_2 \circ \mathbf{f}_1(\mathbf{x})$. Nodes correspond to input layer $\mathbf{x}$, output layer $\mathbf{y}$ and hidden layers $\mathbf{h}_i$, while edges correspond to the operations $\mathbf{f}_i$ mapping each layer onto the next.

2.3.1.1 Multilayer Perceptrons (MLPs)

The classic example of a deep feedforward network is the *multilayer perceptron* (MLP), which has at least one hidden layer. In the MLP model, each layer is *fully connected* to its neighbours in the sense that each component of each layer is a function of every component of the preceding layer. For a MLP with only one hidden layer $\mathbf{h}$, we can specify the network as

$$\mathbf{h} = \mathbf{f}_1(\mathbf{x}; \boldsymbol{\theta}_1) \tag{2.114}$$

$$\mathbf{y} = \mathbf{f}_2(\mathbf{h}; \boldsymbol{\theta}_2) \tag{2.115}$$

If we assume $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{h} \in \mathbb{R}^3$ and $\mathbf{y} \in \mathbb{R}$, the graph associated with the MLP would be as depicted in Figure 2.7.

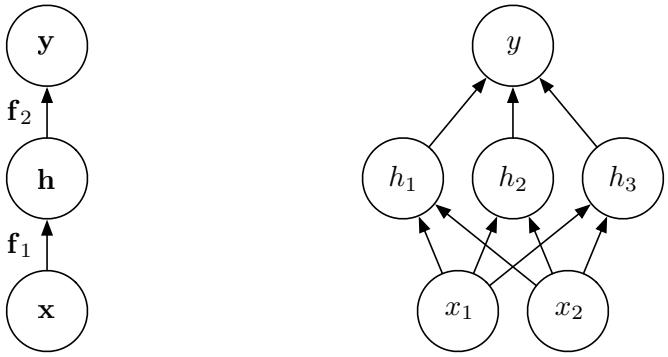


Figure 2.7: Equivalent graphs describing the structure of a multilayer perceptron (MLP) with one hidden layer for the case $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{h} \in \mathbb{R}^3$ and $\mathbf{y} \in \mathbb{R}$.

The question is then what form the functions $\mathbf{f}_1$ and $\mathbf{f}_2$ should take. If both $\mathbf{f}_1$ and $\mathbf{f}_2$ are linear functions then their composition $\mathbf{f} = \mathbf{f}_1 \circ \mathbf{f}_2$ is also linear, which means that our model cannot represent nonlinear functions. Instead, we can choose $\mathbf{f}_2$ to be linear and $\mathbf{f}_1$ to be the composition of a linear transformation and a nonlinear *activation function* $\mathbf{g}$ so that our model becomes

$$\mathbf{h} = \mathbf{f}_1(\mathbf{x}; \mathbf{W}_1, \mathbf{b}_1) = \mathbf{g}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) \tag{2.116}$$

$$\mathbf{y} = \mathbf{f}_2(\mathbf{h}; \mathbf{W}_2, \mathbf{b}_2) = \mathbf{W}_2\mathbf{h} + \mathbf{b}_2 \tag{2.117}$$

where $\mathbf{W}_1 \in \mathbb{R}^{3\times 2}, \mathbf{W}_2 \in \mathbb{R}^{1\times 3}$ are weight matrices and $\mathbf{b}_1 \in \mathbb{R}^3, \mathbf{b}_2 \in \mathbb{R}$ are biases. Because $\mathbf{g}$ is nonlinear, the mapping from $\mathbf{x}$ to $\mathbf{y}$ is nonlinear as well.

While there are a number of possible choices for the activation function, most modern feedforward networks employ the *rectified linear unit* or ReLU (Nair and Hinton, 2010), applied element-wise to the components of $\mathbf{h}$:

$$g(z) = \max\{0, z\} \tag{2.118}$$

Some networks use generalizations of the rectified linear unit such as the leaky ReLU (Maas et al., 2013) or parametric ReLU (PReLU) (He et al., 2015), both given by

$$g(z; \alpha) = \max\{0, z\} + \alpha \min\{0, z\} \tag{2.119}$$

where the parameter α controls the slope of the activation for negative values of z. The difference between the leaky ReLU and the PReLU is simply that α is a learnable parameter of the network for PReLU activations but is manually specified for the leaky ReLU. When $\alpha = 1$ the mapping is exactly linear and when $\alpha = 0$ we recover the basic ReLU activation. Figure 2.8 compares the ReLU and leaky ReLU activation functions. Rectified linear activations are extremely cheap to compute and have the property of being linear everywhere except at zero. Because of this, they preserve many of the properties that make linear models easy to optimize with gradient-based methods. Notably, Krizhevsky et al. (2012) showed that non-saturating ReLU activations enable deep networks to learn several times faster than equivalent networks with saturating activations such as the $\tanh(\cdot)$ function.

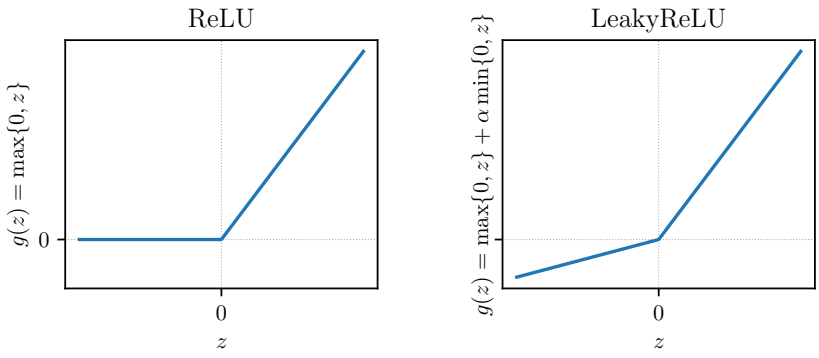


Figure 2.8: ReLU and leaky ReLU ($\alpha = 0.2$) activation functions.

2.3.1.2 Universal Approximation

Feedforward networks with at least one hidden layer (e.g., MLPs) are *universal function approximators* (Hornik et al., 1989; Leshno and Schocken, 1993). That is, for any given function of interest, there exists a MLP that can approximate it with any desired nonzero amount of error. However, there is no guarantee that a given training algorithm will be able to learn that function. For example, the optimization algorithm used for training might not converge to the right set of parameters, or the algorithm may find the wrong function as a result of overfitting to the training data. Moreover, the universal approximation theorem gives no bounds on the network width (or depth) necessary to approximate a target function; establishing bounds to guide model architecture remains an active area of research.

2.3.1.3 Gradient-based Learning

Deep networks are most commonly trained using a variation on *gradient descent* to minimize a chosen loss function $\mathcal{L}(\boldsymbol{\theta})$, which is typically tied to a likelihood maximization objective and may include additional regularization terms. If we can compute the gradient $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ of the loss function with respect to the model parameters, we can take a small step along the gradient direction to move to a point in parameter space with a lower value of the loss function:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \cdot \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}), \tag{2.120}$$

where ϵ is the step size, also referred to as the *learning rate*. After each step, $\mathcal{L}(\boldsymbol{\theta})$ and its gradient can be recomputed and Equation (2.120) applied in an iterative fashion until the algorithm converges to a critical point where $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) = \mathbf{0}$, or another heuristic stopping criterion is satisfied (e.g., an early stopping criterion as discussed in Section 2.3.1.4).

Back-propagation Computing $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$ involves traversing the graph of the network in reverse and using the chain rule of calculus to compute the gradients with respect to each parameter in the graph. If we have, for example, $\mathbf{y} = \mathbf{g}(\mathbf{x})$ and $\mathbf{z} = \mathbf{f}(\mathbf{g}(\mathbf{x})) = \mathbf{f}(\mathbf{y})$, then the chain rule states that

$$\nabla_{\mathbf{x}}\mathbf{z} = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^T \nabla_{\mathbf{y}}\mathbf{z}, \tag{2.121}$$

which implies that the gradient of $\mathbf{z}$ with respect to $\mathbf{x}$ can be obtained simply by multiplying a Jacobian matrix by a gradient vector. The *back-propagation* or *backprop* algorithm (Rumelhart et al., 1986) provides a means of efficiently computing gradients by recursively applying the chain rule for each parameter in the graph. One important advantage of back-propagation is that the practitioner need only specify how to compute the Jacobian-gradient product for each component function in the graph, which is usually a straightforward operation to derive. Because of its efficiency and simplicity as a tool for automatic differentiation, back-propagation is used ubiquitously in deep learning.

Stochastic Optimization In general, the goal of the optimization problem is to minimize the loss function over the training set. Most loss functions used in practice (e.g., log likelihood) decompose as a sum over the training examples so that the overall loss function can be written as

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{J}\sum_{j=1}^{J}\mathcal{L}(\boldsymbol{\theta}; \mathbf{x}_j, \mathbf{y}_j^*) \tag{2.122}$$

where $\boldsymbol{\theta}$ is the set of network parameters, $\mathbf{x}_j$ is an input to the network, $\mathbf{y}_j^*$ is the corresponding training label, and j indexes the J training examples in the training set. In practice, training sets for deep learning models can be very large, containing anywhere from several thousand to several million training examples, which makes computing the loss function and gradients over the entire training set extremely expensive. Because of this, deep learning relies on *stochastic* optimization techniques to estimate the loss function and gradients at each iteration using a subset or *minibatch* of training examples. The classic example of such a technique is *stochastic gradient descent* (SGD), which performs gradient descent based on gradients computed from a minibatch of training examples sampled randomly from the training set without replacement. After the training set has been fully sampled, we say that one *epoch* of training has elapsed; deep networks are typically trained for multiple epochs. Stochastic methods have the crucial property that, because the minibatch size is fixed, each iteration of training incurs constant computational cost, allowing the learning algorithm to scale to very large datasets. In some cases the stochastic gradient estimates are sufficiently close to the full gradient that training will converge in less than one epoch.

Stochastically estimated gradients are necessarily noisy, so the estimated gradient may be non-zero even when the true gradient over the training set is itself zero. As a result, the SGD update may pull the solution away from its optimum. For this reason it is common to gradually decrease the learning rate at each iteration to aid convergence. More sophisticated optimization algorithms such as AdaGrad (Duchi et al., 2011) and Adam (Kingma and Ba, 2015) adapt the learning rate for each parameter by scaling it as a function of the gradient magnitudes and are widely used in practice.

Batch Normalization Batch normalization (Ioffe and Szegedy, 2015) is an adaptive reparameterization technique motivated by the difficulty of training very deep models using gradient descent. This difficulty stems from the fact that the gradients computed by back-propagation provide an update direction for each parameter under the assumption that the other parameters are held constant. In practice, we update all the parameters simultaneously at each iteration, and unexpected results may arise from simultaneously changing many interdependent functions. Batch normalization reduces the difficulty of coordinating updates across many layers by reparameterizing the outputs of each layer to have zero mean and unit variance. For a minibatch $\mathbf{H} \in \mathbb{R}^{M\times N}$ of hidden units arranged such that each row of $\mathbf{H}$

corresponds to a different training example, we normalize $\mathbf{H}$ by making the replacement

$$H_{ij} \leftarrow \frac{H_{ij} - \mu_j}{\sigma_j} \tag{2.123}$$

At training time μ_j is the mean of the j^{th} column of $\mathbf{H}$ and σ_j its standard deviation, given by

$$\mu_j = \frac{1}{M} \sum_{i=1}^{M} H_{ij} \tag{2.124}$$

$$\sigma_j = \sqrt{\delta + \frac{1}{M} \sum_{i=1}^{M} (H_{ij} - \mu_j)^2} \tag{2.125}$$

where δ is a small positive number (e.g., 10^{-8}) imposed to avoid the undefined gradient of $\sqrt{z}$ at $z = 0$. During training, we back-propagate through the normalization operations, which has the effect that the gradient will never propose an operation that simply changes the mean or standard deviation of the outputs. During training we also record running averages of μ_j and σ_j, which are used at test time so that the model can be evaluated at individual test points.

Restricting the outputs of a layer to have zero mean and unit standard deviation can reduce the expressive power of the network. To compensate, it is common to introduce an additional linear transformation so that

$$H_{ij} \leftarrow \gamma_j \left(\frac{H_{ij} - \mu_j}{\sigma_j} \right) + \beta_j \tag{2.126}$$

where γ_j and β_j are learnable parameters that allow the activation statistics to have any mean and standard deviation. In practice, batch normalization has the effect of stabilizing the learning dynamics by decoupling the first- and second-order activation statistics from the preceding layers – the mean and standard deviation are determined entirely by γ_j and β_j.

2.3.1.4 Generalization, Overfitting and Regularization

While the goal of the optimization task is to minimize a loss function over the training set, the goal of the *learning* task is to drive $\boldsymbol{\theta}$ to values such that our learned model $\mathbf{f}$ accurately approximates the true data-generating function $\mathbf{f}^*$. In other words, we want $\mathbf{f}$ to *generalize* to points outside the training set. We can assess how well our model generalizes by evaluating the loss function on a held out validation set and monitoring its progression over successive iterations of the optimization algorithm. Typically the validation loss will decrease along with the training loss for a certain number of iterations before beginning to increase again. At this stage the model is said to have begun *overfitting* to the training data.

Early stopping Overfitting can be prevented using an *early stopping* criterion where training is deemed to have converged once the validation loss reaches its minimum. It is important to note that training often halts while the training loss still has large gradients (i.e., it has not reached a local minimum). This highlights the key difference between learning and pure optimization: we are interested in finding a model that generalizes outside of the training set rather than fitting the training set perfectly.

Weight decay Early stopping is one technique for *regularizing* the model to avoid overfitting to the training set. Other approaches to regularization involve limiting the model capacity. *Weight decay* is a common regularization technique where the loss function is modified to include an additional term $\Omega(\boldsymbol{\theta})$ that penalizes the norm of the model parameters:

$$\mathcal{L}'(\boldsymbol{\theta}) = \frac{1}{J}\sum_{j=1}^{J}\mathcal{L}(\boldsymbol{\theta}; \mathbf{x}_j, \mathbf{y}_j^*) + \alpha\Omega(\boldsymbol{\theta}) \tag{2.127}$$

The variable α is a scalar hyperparameter that controls the relative weight of the regularization term in the loss function. The most common choice of norm penalty is the L_2 norm penalty (also referred to as *ridge regression*), which drives the weights closer to zero by imposing a penalty of the form

$$\Omega(\boldsymbol{\theta}) = \frac{1}{2}\left\|\boldsymbol{\theta}\right\|_2^2 = \frac{1}{2}\sum_i \theta_i^2. \tag{2.128}$$

The L_1 norm penalty is also commonly used and takes the form

$$\Omega(\boldsymbol{\theta}) = \left\|\boldsymbol{\theta}\right\|_1 = \sum_i \left|\theta_i\right|. \tag{2.129}$$

L_1 regularization has the property of encouraging sparsity in the parameters, in the sense that some parameters may have an optimal value of zero.

Dropout Another common approach to regularization is *dropout* (Srivastava et al., 2014). The basic idea of dropout is simple: at each training iteration, we zero out components of one or more layers, each with some probability p, and apply back-propagation and gradient descent through the modified outputs. This operation stochastically reduces the capacity of the model during training so that we are updating a different subnetwork at every iteration. At test time we do not zero out any of the outputs.

Dropout is related to a type of ensemble method known as *bootstrap aggregation* or *bagging*, where the outputs of multiple models are combined to produce a more robust estimate of the underlying function. Like bagging, a model trained using dropout will correspond to an ensemble of exponentially many subnetworks. Unlike bagging, however, the subnetworks share parameters and are therefore not independent. Nevertheless, dropout provides an effective, computationally efficient, and widely applicable approach to regularization that can be easily combined with other regularization techniques such as weight decay.

In addition to the ensemble interpretation, dropout can also be thought of as a form of *noise injection*, which has the effect of forcing the model to be robust to the absence of certain inputs or extracted features. Batch normalization, discussed in Section 2.3.1.3, can also be thought of as a form of noise injection where additive and multiplicative noise are applied to the hidden units during training. Although the primary purpose of batch normalization is to improve optimization, it can also have a regularizing effect that sometimes makes dropout unnecessary.

Transfer learning While regularization and early stopping are useful tools for mitigating overfitting, the best way to improve model generalization is to increase the size of the training set. However, the cost of obtaining large-scale labeled training sets can be prohibitive, and we must often make do with small datasets. In these cases, a common strategy is to "pre-train" a model on a proxy task for which a large

labeled training set exists, and then "fine-tune" the model through a secondary training stage using a smaller, specialized training set. This is a form of *transfer learning*, where a model developed for one task is re-used as a starting point for a model intended for another task. For example, a model intended for image segmentation might require densely labeled image data for training, which can be expensive to obtain. To make use of a small training set and avoid overfitting, we might pre-train part of the model on a related task such as image classification, using a very large dataset such as ImageNet (Deng et al., 2009). After pre-training, we can then replace the parts of the model specific to image classification with parts specific to image segmentation, and fine-tune the model through further training using the smaller densely labeled training set. In this way we can take advantage of large datasets to learn general low-level features in our networks, which can provide an advantageous starting point for further training.

2.3.2 Convolutional Networks

Convolutional networks (LeCun, 1989) or convolutional neural networks (CNNs) are a specialized kind of deep feedforward network designed for processing data that has a known grid-like topology, such as image or time-series data. CNNs have proven to be powerful tools for solving a wide range of robot vision tasks including object recognition, stereo matching, and vision-assisted grasping, among others. The key distinction between convolutional networks and other types of feedforward networks is that they replace general matrix multiplication with a *convolution* operation in at least one of their layers.

2.3.2.1 The Convolution Operation

The convolution of two functions f and g is denoted as $f * g$ and is defined as the integral of the product of the two functions after one is reversed and shifted:

$$s(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau \tag{2.130}$$

In convolutional network terminology, the function f is often referred to as the *input* and the function g as the *kernel*. The output s of the convolution operation is referred to as the *feature map*.

Typically in machine learning applications we are dealing with discretized data such as images, which are represented as multidimensional arrays. For the case of two-dimensional discrete data, such as a grayscale image $\mathbf{I}$, we would use a discrete two-dimensional kernel $\mathbf{K}$ and write the convolution operation as

$$\mathbf{S}(i,j) = (\mathbf{I} * \mathbf{K})(i,j) = \sum_m \sum_n \mathbf{I}(m,n)\mathbf{K}(i-m, j-n) \tag{2.131}$$

$$= (\mathbf{K} * \mathbf{I})(i,j) = \sum_m \sum_n \mathbf{I}(i-m, j-n)\mathbf{K}(m,n) \tag{2.132}$$

since convolution is commutative. The goal of the learning algorithm is then to find the entries of the kernel $\mathbf{K}$ (and perhaps an additional bias term) that minimize the target loss function.

Many neural network libraries implement a related operation called the *cross-correlation* which does not reverse the kernel, but nevertheless call it convolution:

$$\mathbf{S}(i,j) = (\mathbf{I} * \mathbf{K})(i,j) = \sum_m \sum_n \mathbf{I}(i+m, j+n)\mathbf{K}(m,n) \tag{2.133}$$

Figure 2.9 graphically depicts the output of a "valid" 2D convolution where the output is restricted to positions where the kernel lies entirely within the input image. In practice, we often use a *padding* strategy such as zero padding or reflection padding to extend the range of valid convolutions and ensure that the dimensions of the output match the dimensions of the input.

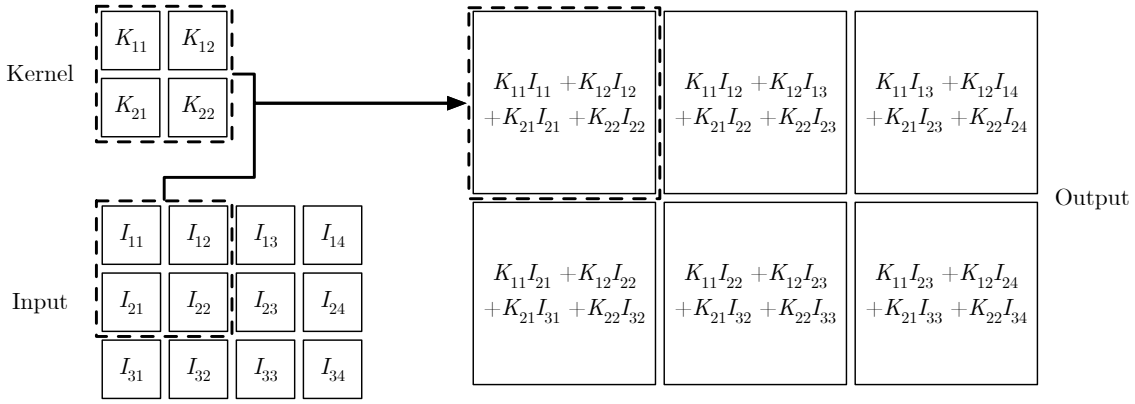


Figure 2.9: Output of a "valid" 2D convolution where the output is restricted to positions where the kernel lies entirely within the input image. In practice, we often use *padding* to ensure that the dimensions of the output match the dimensions of the input.

Convolution is a linear operation, which means that a model composed entirely of convolutions will be a linear model. As in the MLP case, we introduce nonlinearity into the model via a nonlinear activation function such as the ReLU, which is applied to the output feature map. Regularization functions such as dropout or batch normalization are typically applied to the feature map before applying the activation function.

2.3.2.2 Advantages of Convolution for Machine Learning

Convolutional layers have three important advantages for machine learning. First, they provide a means of *sparsifying* the interactions between input and output layers. Whereas a fully-connected layer using matrix multiplication must store a separate parameter for each input-output-connection, convolutional networks take advantage of small (e.g., 3×3) kernels to detect meaningful features such as edges. This property drastically reduces the number of parameters needed to operate on large images with millions of pixels, reducing the computational cost of the model in terms of both memory and processing time. Second, because a single convolutional kernel is used to compute the entire output feature map, parameters are *shared* across spatial locations in the input image, which further reduces the memory requirements of the model. Finally, because parameters are shared across spatial locations, convolutional kernels are *equivarient* to translation, that is, if we shift the input image to the left by one pixel, the output feature map will also be shifted to the left by one pixel. This means that the locations of features in an image (e.g., edges and corners) are preserved in the output feature map.

The number of input units connected to a given output unit is referred to as the *receptive field* of the output unit. When multiple convolutions are chained together in a deep network, feature maps at deeper layers are computed (sparsely) from the feature maps at shallower layers so that the effective receptive field for a given output unit grows as a function of its depth, even though each convolutional layer may use the same small kernel size. This property allows CNNs to efficiently represent hierarchical structure in the data. For example, in an image of a tree, the initial convolutional kernels may detect

green edges, while deeper kernels may combine green edge detections to detect leaves and deeper kernels still may combine leaf detections to detect the tree itself.

2.3.2.3 Pooling and Strided Convolutions

Pooling is a common operation in many convolutional network architectures. Its function is to replace the activations of a convolutional layer with a summary statistic of nearby activations such as the maximum (max pooling) or the average (average pooling). When applied along spatial dimensions, pooling helps to make the representation approximately invariant to small translations. Analogously, if pooling is applied over the outputs of multiple independent convolutional filters, the network can learn to be invariant to other types of input transformations. Moreover, pooling is a common technique for reducing the dimensionality of feature maps in CNNs, which in turn reduces the memory and processing load of the model. For example, by performing spatial max pooling over non-overlapping 2×2 windows, the spatial dimensions of the feature map can be halved as depicted in Figure 2.10.

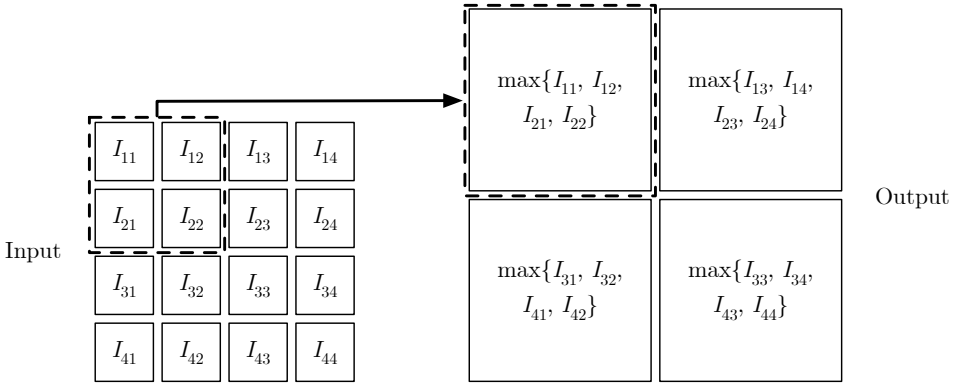


Figure 2.10: Output of a spatial max pooling operation with stride 2, which downsamples a 4×4 input to a 2×2 output.

Pooling is most useful when we are more interested in learning whether a particular feature is present rather than determining its precise location. For many robotic vision tasks, we are interested in preserving the locations of features in the image rather than merely detecting their presence or absence, in which case spatial pooling would be detrimental. An alternative to pooling that can reduce the dimensions of feature maps while preserving spatial information is the *strided convolution*. The stride k of a discrete convolution operation controls the spacing between points in the input at which the product with the kernel is evaluated:

$$\mathbf{S}(i,j) = (\mathbf{I} * \mathbf{K})(i,j) = \sum_m \sum_n \mathbf{I}(ik+m, jk+n)\mathbf{K}(m,n) \tag{2.134}$$

This has the effect of reducing the dimension of the output feature map with respect to the input. For example, a convolution with stride $k = 2$ will result in an output feature map with half the spatial resolution of the input.

It is also possible to use convolutions to *increase* the dimension of the output feature map relative to the input. Such an operation is referred to as *transposed convolution*, *fractionally-strided convolution*, or, less commonly, *deconvolution*. Intuitively, transposed convolution can be thought of as performing regular convolution in reverse: rather than mapping a window of points in the input onto a single point

in the output, a single point in the input is mapped onto a window of points in the output. Figure 2.11 graphically depicts of the output of a 2D transposed convolution. If the kernel size and stride are chosen such that the mappings overlap (e.g., a 3×3 kernel with stride 2), then the final values at each output point become the superposition of two or more mappings. Transposed convolutions are often used to learn upsampling algorithms in image reconstruction tasks such as super-resolution as they can produce more visually appealing images than nearest-neighbour or bilinear interpolation.

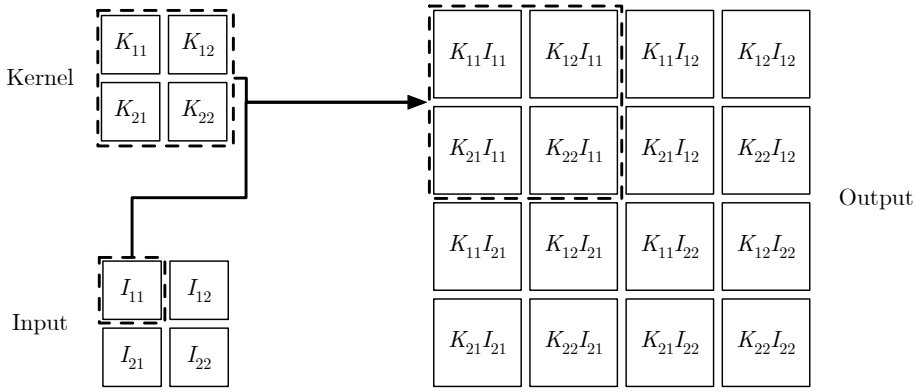


Figure 2.11: Output of a 2D transposed convolution with stride 2, which upsamples a 2×2 input to a 4×4 output.

Chapter 3

Visual Odometry Aided by Indirect Visual Sun Sensing

This chapter presents a method to incorporate global orientation information from the sun (or another known light source) into a visual odometry (VO) pipeline using only the existing image stream, in which the sun is typically not visible. We leverage recent advances in Bayesian Convolutional Neural Networks (BCNNs) to train and implement a deep feedforward network, dubbed Sun-BCNN, that infers a three-dimensional sun direction vector from a single RGB image. Crucially, our method also computes a principled uncertainty associated with each prediction, using a Monte Carlo dropout scheme (Gal and Ghahramani, 2016). We incorporate this uncertainty into a sliding window stereo visual odometry pipeline where accurate uncertainty estimates are critical for optimal data fusion. We evaluate our method on 21.6 km of urban driving data from the KITTI odometry (Geiger et al., 2012) benchmark where the method achieves a median error of approximately 12 degrees and yields improvements of up to 42% in translational ARMSE and 32% in rotational ARMSE compared to an equivalent VO pipeline without orientation correction. We further evaluate our method on an additional 10 km of visual navigation data from the Devon Island Rover Navigation dataset (Furgale et al., 2012), achieving a median error of less than 8 degrees and yielding similar improvements in estimation error. In each case we compare Sun-BCNN to contemporary hand-engineered and learned models for single-image sun detection, and find that Sun-BCNN consistently yields the greatest improvements in VO accuracy. Under clear-sky conditions, these improvements are comparable to, but generally smaller than, those achieved using a specialized hardware sun sensor making direct observations of the sun. In addition to reporting on the accuracy of Sun-BCNN and its impact on VO, we analyze the sensitivity of our model to cloud cover, investigate the possibility of model transfer between urban and planetary analogue environments, and examine the impact of different methods for computing the mean and covariance of a norm-constrained vector on the accuracy and consistency of the estimated sun directions. An open-source implementation of Sun-BCNN using Caffe (Jia et al., 2014) is available at `https://github.com/utiasSTARS/sun-bcnn`.

This work was done jointly with Valentin Peretroukhin, who focused on formulating the Bayesian learning problem, while I focused on the overall localization pipeline and comparisons against competing approaches to single-image sun detection including hand-engineered methods.

3.1 Motivation

A crucial competency for any autonomous mobile robot is the ability to estimate its own motion through an operating environment. While there exists a rich body of literature on the topic of motion estimation using a variety of techniques such as lidar-based point cloud matching (Zhang and Singh, 2015) and visual-inertial odometry (Leutenegger et al., 2015), egomotion estimation is fundamentally a process of dead-reckoning and will accumulate unbounded error over time. This accumulated error, or drift, can be limited by incorporating global information into the motion estimation problem. This frequently takes the form of a globally consistent map, loop closure detection, or reliance on additional sensors such as GNSS receivers to make corrections to the estimated trajectory. In many situations, however, a globally consistent map may be unavailable or prohibitively expensive to compute, loop closures may not occur, or GNSS may be unavailable or inaccurate. In such cases, it can be advantageous to rely on celestial navigation using the sun or other celestial bodies that are readily detectable (e.g., using a sun sensor) and whose apparent motion in the sky is well described by ephemeris models.

For visual odometry (VO) in particular, the addition of global orientation information can limit the growth of drift error to be linear rather than superlinear with distance traveled (Olson et al., 2003). Sun-based orientation corrections have been successfully used in planetary analogue environments (Furgale et al., 2011; Lambert et al., 2012) as well as on board the Mars Exploration Rovers (MERs) (Eisenman et al., 2002; Maimone et al., 2007). In particular, Lambert et al. (2012) showed that incorporating sun sensor and inclinometer measurements directly into the estimation pipeline (as opposed to periodically updating the vehicle heading, as in earlier work) can significantly reduce VO drift over long trajectories.

In this chapter, we seek to answer the question of whether similar reductions in VO drift can be obtained solely from the image stream already being used to compute VO. The main idea here is that by reasoning over more than just the geometric information available from a standard RGB camera, we can improve existing VO techniques without needing to rely on a dedicated sun sensor or specially oriented camera. In particular, we leverage recent advances in Bayesian Convolutional Neural Networks (BCNNs) to demonstrate how we can build and train a deep model capable of inferring the direction of the sun from a single RGB image. Moreover, we show that our network, Sun-BCNN, can produce a covariance estimate for each observation that obviates the need for a hand-tuned or empirically computed static covariance typically used for data fusion in a motion estimation pipeline.

The remainder of this chapter begins with a discussion of related work, followed by an overview of the theory underlying BCNNs and a discussion of our model architecture, implementation, and training procedure. We then outline our chosen visual odometry pipeline, which is based on a two-frame bundle adjustment optimization, and describe how observations of the sun can be incorporated directly into the motion estimation problem following the technique of Lambert et al. (2012). Finally, we present several sets of experiments designed to test and validate both Sun-BCNN and our sun-aided VO pipeline in a variety of environments. These include experiments on 21.6 km of urban driving data from the KITTI odometry benchmark (Geiger et al., 2012), as well as a further 10 km traverse from the Devon Island Rover Navigation Dataset collected in a planetary analogue site in the Canadian High Arctic (Furgale et al., 2012). We investigate the possibility of model generalization between different cameras and environments, and further explore the sensitivity of Sun-BCNN to cloud cover during training and testing, using data from the Oxford RobotCar Dataset (Maddern et al., 2017). We also examine the impact of different methods for computing the mean and covariance of a norm-constrained vector on the accuracy and consistency of the estimated sun directions.

3.2 Related Work

Visual odometry (VO), a technique to estimate the motion of a moving platform equipped with one or more cameras, has a rich research history including a notable implementation onboard the Mars Exploration Rovers (MERs) (Matthies, 1989; Maimone et al., 2007; Scaramuzza and Fraundorfer, 2011). Modern approaches to VO can achieve estimation errors below 1% of total distance traveled over kilometer-scale trajectories (Geiger et al., 2013). To achieve such accurate and robust estimates, modern techniques use careful visual feature pruning (Cvišić and Petrović, 2015), adaptive robust methods (Alcantarilla and Woodford, 2016; Peretroukhin et al., 2016), or operate directly on pixel intensities (Engel et al., 2015).

Independent of the estimator, VO exhibits superlinear error growth, and is particularly sensitive to errors in orientation (Olson et al., 2003; Cvišić and Petrović, 2015). One way to reduce orientation error is to incorporate observations of a landmark whose position or direction in the navigation frame is known *a priori*. The sun is an example of such a known directional landmark. Accordingly, hardware sun sensors have been used to improve the accuracy of VO in planetary analogue environments (e.g., the Sinclair Interplanetary SS-411 sun sensor used by Furgale et al. (2011) and Lambert et al. (2012)), while the MERs articulated their Pancam apparatus to directly image the sun (Maimone et al., 2007; Eisenman et al., 2002). More recently, software-based alternatives have been developed that can estimate the direction of the sun from a single image, making sun-aided navigation possible without additional sensors or a specially-oriented camera (Clement et al., 2017b). Some of these methods have been based on hand-crafted illumination cues such as shadows and variation in sky brightness (Lalonde et al., 2012; Clement et al., 2017b), while others have attempted to learn such cues from data using deep Convolutional Neural Networks (CNNs) (Ma et al., 2017).

Convolutional Neural Networks (CNNs) have been applied to a wide range of classification, segmentation, and learning tasks in computer vision (LeCun et al., 2015). Recent work has shown that CNNs can learn orientation information directly from images by modifying the loss functions of existing discrete classification-based CNN architectures into continuous regression losses (Ma et al., 2017; Kendall et al., 2015; Kendall and Cipolla, 2016). Despite their success in improving prediction accuracy, most existing CNN-based models do not report uncertainty estimates, which are important in the context of data fusion.

For classification, it is possible to restrict CNN model outputs to a certain range (e.g., using a softmax function) and interpret these values as the model's confidence in its output. As Gal (2016) noted, however, this can be misleading because these values can be unjustifiably large for test points far away from training data. To address this, Gal and Ghahramani (2016) showed that it is possible to achieve *principled* covariance outputs that better quantify model uncertainty for classification and regression tasks, with only minor modifications to existing CNN architectures. An early application of this uncertainty quantification was presented by Kendall and Cipolla (2016) who used it to improve their prior work on camera pose regression (Kendall et al., 2015).

Our method is similar in spirit to the work of Ma et al. (2017), who built a CNN-based sun sensor as part of a relocalization pipeline, however we make two important improvements:

1. In addition to a point estimate of the sun direction, we output a principled covariance estimate that is incorporated into our VO pipeline; and

2. We produce a full 3D sun direction estimate with azimuth and zenith angles that is better suited

to 6-dof robot pose estimation problems, as opposed to only the azimuth angle and 3-dof estimator used by Ma et al. (2017).

Furthermore, our Bayesian CNN includes a dropout layer after every convolutional and fully connected layer (as outlined by Gal and Ghahramani (2016) but not done by Kendall and Cipolla (2016)), which produces more principled covariance outputs.

3.3 Methodology

Our goal in this work is to train a deep neural network to estimate a unit-norm vector corresponding to the direction of the sun, as well as an associated uncertainty, from a single RGB image. We incorporate measurements from this 'virtual sun sensor' as additional observations in a feature-based stereo visual odometry (VO) pipeline based on the modelling and estimation tools discussed in Chapter 2, with the intent of correcting estimation errors induced by orientation drift over long trajectories. Figure 3.1 illustrates our method at a high level. We compare the performance of our proposed method against contemporary state-of-the-art techniques for single-image sun detection based both on physically motivated hand-engineered features (Lalonde et al., 2012; Clement et al., 2017b) and deep learning (Ma et al., 2017).

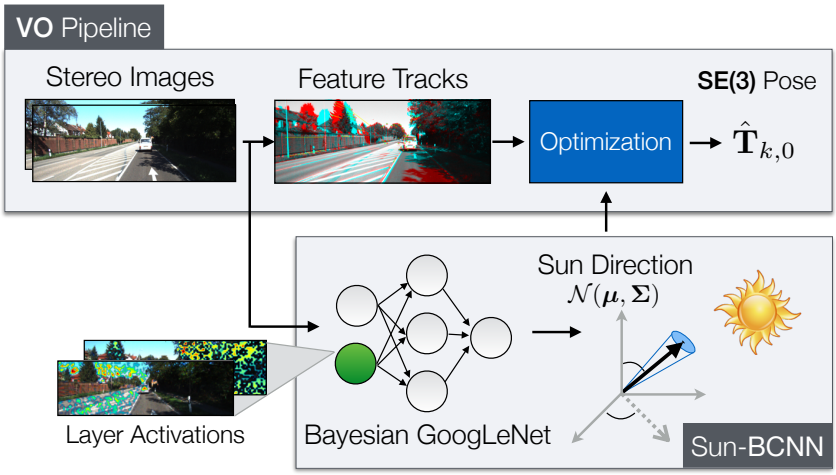


Figure 3.1: Our method uses a Bayesian Convolutional Neural Network (BCNN) to estimate the direction of the sun and to produce a principled uncertainty estimate for each prediction. We incorporate this 'virtual sun sensor' into a stereo visual odometry pipeline to reduce estimation error.

Section 3.4 summarizes experimental procedures and provides an analysis of the results, while this section discusses the component pieces of our system, including the base VO pipeline and our approach to fusing sun measurements with sparse feature tracking (Section 3.3.1), methods for single-image sun detection using hand-engineered features (Section 3.3.2), and our approach to learning a single-image sun detection model using a Bayesian Convolutional Neural Network (Section 3.3.3).

3.3.1 Sun-Aided Stereo Visual Odometry

We adopt a sliding window sparse stereo VO technique that has been used in a number of successful mobile robotics applications (Cheng et al., 2006; Furgale and Barfoot, 2010; Geiger et al., 2011; Kelly

et al., 2008). Our task is to estimate a window of SE(3) poses $\{\mathbf{T}_{k_1,0}, \mathbf{T}_{k_1+1,0}, \ldots, \mathbf{T}_{k_2-1,0}, \mathbf{T}_{k_2,0}\}$ expressed in a base coordinate frame $\underrightarrow{\mathcal{F}}_0$, given a prior estimate of the transformation $\mathbf{T}_{k_1,0}$. We accomplish this by tracking keypoints across pairs of stereo images and computing an initial guess for each pose in the window using frame-to-frame point cloud alignment, which we then refine by solving a local bundle adjustment problem over the window. In our experiments we choose a window size of two, which we observed to provide good VO accuracy at low computational cost. We select the initial pose $\mathbf{T}_{1,0}$ to be the first GNSS ground truth pose such that $\underrightarrow{\mathcal{F}}_0$ is a local Easting-Northing-Up (ENU) coordinate system with its origin at the first GNSS position.

3.3.1.1 Observation Model

We assume that incoming stereo images have been undistorted and rectified in a pre-processing step, and model the stereo camera as a pair of perfect pinhole cameras with focal lengths f_u, f_v, and principal points (c_u, c_v), separated by a fixed and known baseline b (see Section 2.2.1.2). If we take $\mathbf{p}_0^j$ to be the homogeneous 3D coordinates of keypoint j, expressed in our chosen base frame $\underrightarrow{\mathcal{F}}_0$, we can transform the keypoint into the camera frame at pose k to obtain $\mathbf{p}_k^j = \mathbf{T}_{k,0}\mathbf{p}_0^j = \begin{bmatrix} p_{k,x}^j & p_{k,y}^j & p_{k,z}^j & 1 \end{bmatrix}^T$. Our observation model $\mathbf{g}(\cdot)$ can then be formulated as

$$\mathbf{y}_{k,j} = \mathbf{g}(\mathbf{p}_k^j) = \begin{bmatrix} u \\ v \\ d \end{bmatrix} = \begin{bmatrix} f_u p_{k,x}^j / p_{k,z}^j + c_u \\ f_v p_{k,y}^j / p_{k,z}^j + c_v \\ f_u b / p_{k,z}^j \end{bmatrix}, \tag{3.1}$$

where (u, v) are the keypoint coordinates in the left image and d is the disparity in pixels.

3.3.1.2 Sliding Window Bundle Adjustment

We use the open-source `libviso2` library (Geiger et al., 2011) to detect and track keypoints between stereo image pairs. Based on these keypoint tracks, a three-point Random Sample Consensus (RANSAC) algorithm (Fischler and Bolles, 1981) generates an initial guess of the inter-frame motion and rejects outlier keypoint tracks by thresholding their reprojection error. We compound these pose-to-pose transformation estimates through our chosen window and refine them using a local bundle adjustment, which we solve via the nonlinear least-squares solver Ceres (Agarwal et al., 2016). The objective function to be minimized can be written as

$$\mathcal{J} = \mathcal{J}_{\text{reprojection}} + \mathcal{J}_{\text{prior}}, \tag{3.2}$$

where

$$\mathcal{J}_{\text{reprojection}} = \sum_{k=k_1}^{k_2} \sum_{j=1}^{J} \mathbf{e}_{\mathbf{y}_{k,j}}^T \mathbf{R}_{\mathbf{y}_{k,j}}^{-1} \mathbf{e}_{\mathbf{y}_{k,j}} \tag{3.3}$$

and

$$\mathcal{J}_{\text{prior}} = \mathbf{e}_{\check{\mathbf{T}}_{k_1,0}}^T \mathbf{R}_{\check{\mathbf{T}}_{k_1,0}}^{-1} \mathbf{e}_{\check{\mathbf{T}}_{k_1,0}}. \tag{3.4}$$

The quantity $\mathbf{e}_{\mathbf{y}_{k,j}} = \hat{\mathbf{y}}_{k,j} - \mathbf{y}_{k,j}$ represents the reprojection error of keypoint j for camera pose k, with $\mathbf{R}_{\mathbf{y}_{k,j}}$ being the covariance of these errors. The predicted measurements are given by $\hat{\mathbf{y}}_{k,j} = \mathbf{g}(\hat{\mathbf{T}}_{k,0}\hat{\mathbf{p}}_0^j)$, where $\hat{\mathbf{T}}_{k,0}$ and $\hat{\mathbf{p}}_0^j$ are the estimated poses and keypoint positions in base frame $\underrightarrow{\mathcal{F}}_0$.

The cost term $\mathcal{J}_{\text{prior}}$ imposes a normally distributed prior $\check{\mathbf{T}}_{k_1,0}$ on the first pose in the current window, based on the estimate of this pose in the previous window. The error in the current estimate

$\hat{\mathbf{T}}_{k_1,0}$ of this pose compared to the prior can be computed via the SE(3) matrix logarithm as $\mathbf{e}_{\check{\mathbf{T}}_{k_1,0}} = \log\left(\check{\mathbf{T}}_{k_1,0}^{-1}\hat{\mathbf{T}}_{k_1,0}\right)^{\vee} \in \mathbb{R}^6$, using the $\log(\cdot)$ and $(\cdot)^{\vee}$ operators defined in Section 2.1.1. The 6×6 matrix $\mathbf{R}_{\check{\mathbf{T}}_{k_1,0}}$ is the covariance associated with $\check{\mathbf{T}}_{k_1,0}$ in its local tangent space, and is obtained as part of the previous window's bundle adjustment solution (see Section 2.1.2.3). This prior term allows consecutive windows of pose estimates to be combined in a principled way that appropriately propagates global pose uncertainty from window to window, which is essential in the context of optimal data fusion.

3.3.1.3 Orientation Correction

In order to combat drift in the VO estimate produced by accumulated orientation error, we adopt the technique of Lambert et al. (2012) to incorporate absolute orientation information from the sun directly into the estimation problem. We assume the initial camera pose and its timestamp are available from a GNSS system and use them to determine the global direction of the sun $\mathbf{s}_0$, expressed as a 3D unit vector, from ephemeris data. [1] We define the world frame $\underrightarrow{\boldsymbol{\mathcal{F}}}_0$ to be a local Easting-Northing-Up (ENU) coordinate system with the initial GNSS position as its origin. At each timestep we update $\mathbf{s}_0$ by querying the ephemeris model using the current timestamp and the initial camera pose, allowing our model to account for the apparent motion of the sun over long trajectories.

By transforming the global sun direction into each camera frame $\underrightarrow{\boldsymbol{\mathcal{F}}}_k$ in the window, we obtain predicted sun directions $\hat{\mathbf{s}}_k = \hat{\mathbf{T}}_{k,0}\mathbf{s}_0$, where $\hat{\mathbf{T}}_{k,0}$ is the current estimate of camera pose k in the base frame. We compare the predicted and estimated sun directions to introduce an additional error term into the bundle adjustment cost function (cf. Equation (3.2)):

$$\mathcal{J} = \mathcal{J}_{\text{reprojection}} + \mathcal{J}_{\text{prior}} + \mathcal{J}_{\text{sun}}, \tag{3.5}$$

where

$$\mathcal{J}_{\text{sun}} = \sum_{k=k_1}^{k_2} \mathbf{e}_{\mathbf{s}_k}^T \mathbf{R}_{\mathbf{s}_k}^{-1} \mathbf{e}_{\mathbf{s}_k}, \tag{3.6}$$

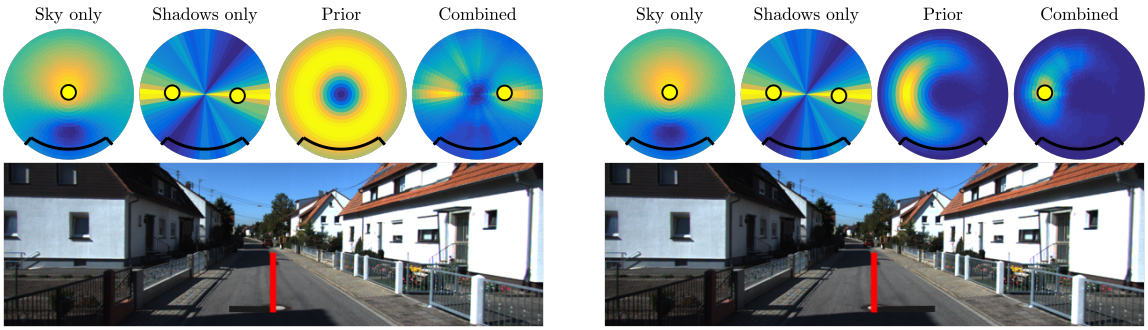
and $\mathcal{J}_{\text{reprojection}}$ and $\mathcal{J}_{\text{prior}}$ are defined in Equations (3.3) and (3.4), respectively. This additional term constrains the orientation of the camera, which helps limit drift in the VO result due to orientation error (Lambert et al., 2012).

Since $\mathbf{s}_k$ is constrained to be unit length, there are only two underlying degrees of freedom. We therefore re-express each 3D unit vector in camera frame $\underrightarrow{\boldsymbol{\mathcal{F}}}_k$ using a zenith-azimuth parameterization:

$$\mathbf{y}_k = \begin{bmatrix} \theta_k \\ \phi_k \end{bmatrix} = \begin{bmatrix} \operatorname{acos}\left(-s_{k,y}\right) \\ \operatorname{atan2}\left(s_{k,x}, s_{k,z}\right) \end{bmatrix}, \tag{3.7}$$

where $\mathbf{s}_k = \begin{bmatrix} s_{k,x} & s_{k,y} & s_{k,z} \end{bmatrix}^T$. We can then re-define the error term $\mathbf{e}_{\mathbf{s}_k} = \hat{\mathbf{y}}_k - \mathbf{y}_k$ to be the error in the predicted sun direction, expressed in zenith-azimuth coordinates, and $\mathbf{R}_{\mathbf{s}_k}$ to be the covariance of these errors. In practice, we also attempt to mitigate the effect of outlier sun predictions by applying a robust Huber loss to the sun measurements in our optimizer (see Section 2.1.2.5).

[1] Specifically, we use ephemeris data available through the MATLAB Aerospace Toolbox, which are based on databases provided by the NASA Jet Propulsion Laboratory.



(a) An ambiguous detection resulting in an incorrect maximum likelihood solution, using the prior term of Lalonde et al. (2012).

(b) The ambiguity is resolved using a VO-informed prior, which constrains the distribution over sun positions.

Figure 3.2: Sample frame from KITTI sequence `2011_09_30_drive_0018` and associated sun detection results using the "Lalonde" (Lalonde et al., 2012) and "Lalonde-VO" methods. *Top row*: Probability distributions over sun positions are shown for each visual cue independently, and for the combined result, with blue representing regions of low probability and yellow representing regions of high probability. The maximum likelihood solution(s) are represented as yellow circles, and the camera's field of view is shown in black. *Bottom row*: A virtual sundial (red line) is inserted into the image and casts a virtual shadow (black line) using the detected sun position.

3.3.2 Indirect Sun Detection Using Physically Motivated Visual Cues

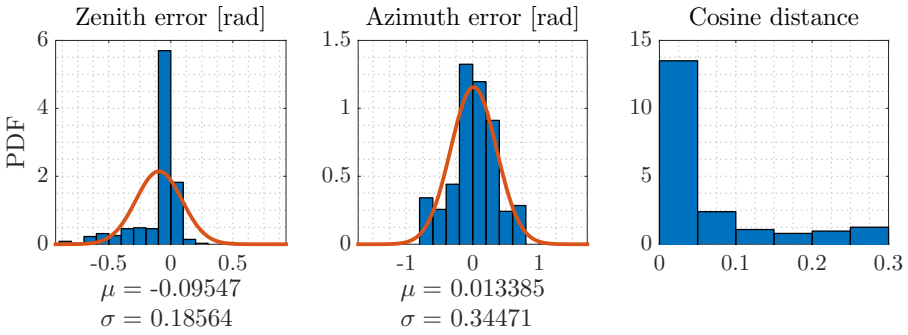
Lalonde et al. (2012) demonstrated that the likely direction of the sun can be estimated from a single RGB image using a combination of weak visual cues such as shadows and a model of the sky (Perez et al., 1993). We improve the accuracy and reliability of this technique by incorporating information from the VO estimate itself, and incorporate it into the sun-augmented VO pipeline discussed in Section 3.3.1 to evaluate its effect on VO drift.

3.3.2.1 "Lalonde"

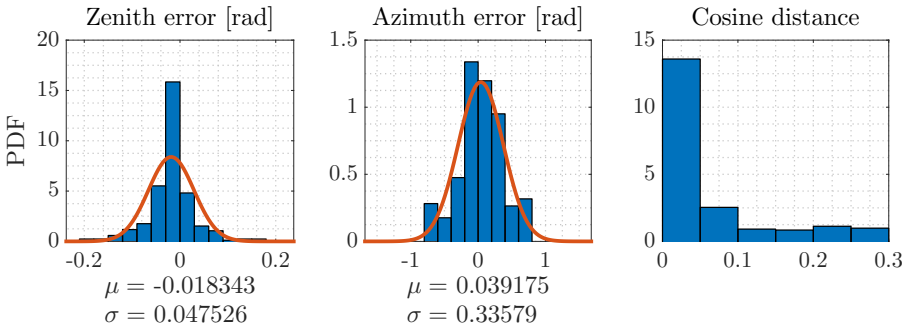
The "Lalonde" method (Lalonde et al., 2012) estimates the maximum likelihood zenith-azimuth sun direction in a single RGB image by combining relatively weak information from a physically based sky model (Perez et al., 1993), shadow detection, pedestrian detection, and vertical surface detection routines, as well as a data-driven prior term that captures the distribution of typical sun zeniths in photographs. An implementation of this technique is freely available as open-source software.[2] For our purposes, we use only a subset of these visual cues since the others tended to produce erroneous or null results in our experiments. Specifically, we use the sky model, shadow detection, and prior term described by Lalonde et al. (2012). Figure 3.2a shows an example of the results we obtained using this method. Note that in this case the algorithm produced an incorrect sun detection due to the bimodal ambiguity in the shadow cue and the symmetry of the sky model and prior term.

Since the "Lalonde" method tends to fail in the presence of ambiguous shadows and saturated sky pixels, we reject obvious outliers in our VO pipeline by thresholding the cosine distance between the observed and predicted sun directions based on the current pose estimate. In practice, we found a cosine distance threshold of 0.3 to be a reasonable choice. However, as shown in Figure 3.3a, the distribution of zenith errors is skewed. This is due to the bias introduced by the prior term of this method, which

[2] https://github.com/jflalonde/illuminationSingleImage



(a) Azimuth errors are approximately zero mean and Gaussian, but zenith errors are skewed due to failures in the sky cue and the biased nature of the prior term introduced by Lalonde et al. (2012) (Figure 3.2a), which fails to correctly capture the distribution of sun positions in the KITTI dataset.



(b) Thresholding the zenith error to exclude the skewed portion of the distribution yields a more Gaussian-like distribution of zenith errors.

Figure 3.3: Distribution of estimation errors for "Lalonde" relative to the ground truth sun vector transformed through the chain of ground truth camera poses. We use a cosine distance threshold of 0.3 to reject outlier estimates.

fails to correctly capture the distribution of sun zeniths in the KITTI dataset. We resolve this issue by thresholding the zenith error (or, equivalently, the y-component error in the camera frame) to exclude the skewed portion of the distribution, yielding a more Gaussian-like distribution over zenith errors.

3.3.2.2 "Lalonde-VO"

The "Lalonde-VO" method (Clement et al., 2017b) is a modified version of the "Lalonde" method where we have replaced the original zenith-only prior term with a novel prior term that incorporates the expected sun direction based on the current VO estimate. The motivation for incorporating this information is twofold. First, in cases where the sky cue fails, the shadow cue's bimodal probability distribution forces the algorithm to choose one of two equally likely solutions at random, leading to a high proportion of erroneous measurements (Figure 3.2a). By incorporating a weak prior based on the estimated camera pose, we can resolve the ambiguity in the two solutions (Figure 3.2b). Second, ambiguous shadow cues often result in an incorrect pair of maximum likelihood sun azimuths, yet there is typically a secondary pair of local maxima with lower probability that are in fact correct. The sky cue alone is not generally strong enough to bias the result towards the correct direction in these cases, but our new VO-informed prior term allows the algorithm to ignore incorrect shadow orientations and incorporate information from the weaker pair of maxima.

We define our VO-informed prior term as a Gaussian distribution over azimuth and zenith angles whose mean is the expected sun direction, and choose the covariance of this distribution such that the 3σ bounds on the azimuth prior span 360°, while the 3σ bounds on the zenith prior span 90°. In this way, we account for uncertainty in the camera poses and avoid excessively biasing the sun detection; we need only bias the result towards the correct 'half' of the sky.

3.3.3 Indirect Sun Detection Using a Bayesian Convolutional Neural Network

While the term $\mathbf{R}_{\mathbf{s}_k}$ in Equation (3.6) would generally be treated as an empirically determined static covariance, we would ideally like to use a per-observation covariance to weight each observation individually according to a measure of its intrinsic quality. In this work, we use a Bayesian Convolutional Neural Network (BCNN) to infer both the direction of the sun and an associated covariance matrix, and refer to our model as Sun-BCNN. We motivate the choice of a deep model through the empirical findings of Clement et al. (2017b) and Ma et al. (2017), who demonstrated that a CNN-based sun detector can substantially outperform hand-crafted models such as that of Lalonde et al. (2012) both in terms of measurement accuracy and in its application to a VO task.

We choose a deep neural network structure based on GoogLeNet (Szegedy et al., 2015) due to its use in past work that adapted it for orientation regression (Kendall and Cipolla, 2016; Kendall et al., 2015). Our model takes as input an image $\mathbf{x}_k$ and outputs a unit-norm vector $\hat{\mathbf{s}}_k$ corresponding to the estimated direction of the sun expressed in the reference frame of the camera:

$$\hat{\mathbf{s}}_k = \mathcal{S}(\mathbf{x}_k; \boldsymbol{\theta}), \tag{3.8}$$

where $\mathcal{S}$ refers to the Sun-BCNN model and $\boldsymbol{\theta}$ to the network parameters. Like Ma et al. (2017), we pre-train our model using a proxy image classification task, however we choose to transfer weights trained on the MIT Places dataset (Zhou et al., 2014) rather than ImageNet (Deng et al., 2009). We believe the MIT Places dataset is a more appropriate starting point for localization tasks than ImageNet since it includes outdoor scenes and is concerned with classifying physical locations rather than objects.

3.3.3.1 Cost Function

We train Sun-BCNN by minimizing the cosine distance between the unit-norm target sun direction vector $\mathbf{s}_k$ and the predicted unit-norm sun direction vector $\hat{\mathbf{s}}_k$, where k indexes the images in the training set:

$$\mathcal{L}(\boldsymbol{\theta}) = 1 - (\hat{\mathbf{s}}_k \cdot \mathbf{s}_k), \tag{3.9}$$

with $\hat{\mathbf{s}}_k = \mathcal{S}(\mathbf{x}_k; \boldsymbol{\theta})$. Note that in our implementation, we do not formulate the cosine distance loss explicitly, but instead minimize half the square of the tip-to-tip Euclidean distance between $\mathbf{s}_k$ and $\hat{\mathbf{s}}_k$, which is equivalent to Equation (3.9) since both vectors have unit length:

$$\begin{aligned}
\frac{1}{2}\left\|\hat{\mathbf{s}}_k - \mathbf{s}_k\right\|^2 &= \frac{1}{2}\left(\left\|\hat{\mathbf{s}}_k\right\|^2 + \left\|\mathbf{s}_k\right\|^2 - 2(\hat{\mathbf{s}}_k \cdot \mathbf{s}_k)\right) \\
&= 1 - (\hat{\mathbf{s}}_k \cdot \mathbf{s}_k) \\
&= \mathcal{L}(\boldsymbol{\theta}).
\end{aligned}$$

We ensure that our network output, $\hat{\mathbf{s}}_k$, has a unit norm by appending a normalization layer to the network.

3.3.3.2 Uncertainty Estimation

Following recent work on Bayesian Convolutional Neural Networks (BCNNs) (Gal and Ghahramani, 2016; Gal, 2016), we modify our model architecture to enable the computation of principled covariance estimates associated with each predicted sun direction. To achieve computationally tractable Bayesian inference within a CNN architecture, BCNNs exploit a connection between stochastic regularization (e.g., dropout, discussed in Section 2.3.1.4) and approximate variational inference of a Bayesian Neural Network. We outline the technique here briefly, and refer the reader to Gal and Ghahramani (2016) for more details.

The method begins with a prior $p(\boldsymbol{\theta})$ on the weights in a deep neural network and attempts to compute a posterior distribution $p(\boldsymbol{\theta}|\mathbf{X},\mathbf{S})$ given training inputs $\mathbf{X} = \{\mathbf{x}_k\}$ and targets $\mathbf{S} = \{\mathbf{s}_k\}$. This posterior can be used to compute a predictive distribution for test samples but is generally intractable. To overcome this, the BCNN approach notes that CNN training with stochastic regularization can be viewed as variational inference if we define a variational distribution $q(\boldsymbol{\theta})$ as:

$$q(\boldsymbol{\theta}_i) = \mathbf{M}_i \text{diag}\left\{\{b_j^i\}_{j=1}^{K_i}\right\}, \tag{3.10}$$

$$b_j^i \in \text{Bernoulli}(p_i). \tag{3.11}$$

Here, i indexes a particular layer in the neural network with K_i weights, $\mathbf{M}$ are the weights to be optimized, b_j^i are Bernoulli distributed binary variables, and p_i is the dropout probability for weights in layer i.

With this variational distribution $q(\boldsymbol{\theta})$, training a CNN with dropout is analogous to minimizing $\text{KL}(p(\boldsymbol{\theta}|\mathbf{X},\mathbf{S}) \;||\; q(\boldsymbol{\theta}))$, the Kullback-Leibler (KL) divergence between the variational distribution and the true posterior. At test time, the first two moments of the predictive distribution are approximated using Monte Carlo integration over the weights $\boldsymbol{\theta}$:

$$\mathbb{E}\left[\hat{\mathbf{s}}_k\right] = \bar{\hat{\mathbf{s}}}_k \approx \frac{1}{N}\sum_{n=1}^{N} \hat{\mathbf{s}}_k^n \tag{3.12}$$

$$\mathbb{E}\left[\hat{\mathbf{s}}_k \hat{\mathbf{s}}_k^T\right] \approx \tau^{-1}\mathbf{1} + \frac{1}{N}\sum_{n=1}^{N} \hat{\mathbf{s}}_k^n \, \hat{\mathbf{s}}_k^{n\,T} - \bar{\hat{\mathbf{s}}}_k \bar{\hat{\mathbf{s}}}_k^T, \tag{3.13}$$

where $\mathbf{1}$ is the identity matrix,

$$\hat{\mathbf{s}}_k^n = \mathcal{S}(\mathbf{x}_k; \boldsymbol{\theta}^n), \tag{3.14}$$

and $\boldsymbol{\theta}^n$ is a sample from $q(\boldsymbol{\theta})$, obtained by sampling the network with dropout. The model precision τ is computed as

$$\tau = \frac{pl^2}{2M\lambda}, \tag{3.15}$$

where p is the dropout probability, l is the characteristic length scale, M is the number of samples in the training data, and λ is the weight decay.

Following Gal and Ghahramani (2016), we build our BCNN by adding dropout layers after every

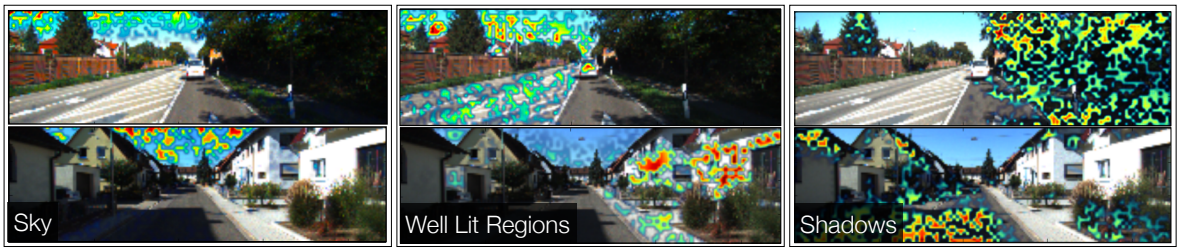


Figure 3.4: Three `conv1` layer activation maps superimposed on two images from the KITTI odometry benchmark (Geiger et al., 2012) `00` and `04` for three selected filters. Each filter picks out salient parts of the image that aid in sun direction inference.

convolutional and fully connected layer in the network. We retain these layers at test time to sample the network stochastically (a technique referred to as Monte Carlo dropout), and obtain the relevant statistical quantities using Equations (3.12) and (3.13).

3.3.3.3 Implementation and Training

We implemented our network in Caffe (Jia et al., 2014), using the `L2Norm` layer from the Caffe-SL fork[3] to enforce a unit-norm constraint on the final output. We trained the network using stochastic gradient descent, setting all dropout probabilities to 0.5, performing 30,000 iterations with a batch size of 64, and setting the initial learning rate to be between 10^{-3} and 10^{-4}. Training required approximately 2.5 hours on an NVIDIA Titan X GPU. Interestingly, Figure 3.4 shows that some convolutional filters learned by Sun-BCNN on the KITTI dataset appear to correspond to illumination variations reminiscent of the visual cues designed by Lalonde et al. (2012).

Data Preparation & Transfer Learning We resized images from their original size to $[224 \times 224]$ pixels to achieve the image size expected by GoogLeNet. We experimented with preserving the aspect ratio of the original image and padding zeros to the top and bottom of the resized image, but found that preserving the vertical resolution (as done by Ma et al. (2017)) results in better test-time accuracy. We did not crop or rotate the images, nor did we augment the dataset in any other way.

Model Precision We found an empirically optimal model precision τ (see Equation (3.15)) by optimizing the Average Normalized Estimation Error Squared (ANEES) across the entire test set for each dataset. The ANEES is computed as

$$\epsilon = \frac{1}{nK} \sum_{k=1}^{K} \left(\mathbf{s}_k - \bar{\hat{\mathbf{s}}}_k\right)^T \hat{\mathbf{R}}_{\mathbf{s}_k}^{-1} \left(\mathbf{s}_k - \bar{\hat{\mathbf{s}}}_k\right) \tag{3.16}$$

where $\hat{\mathbf{R}}_{\mathbf{s}_k}$ is the covariance matrix associated with $\bar{\hat{\mathbf{s}}}_k$ (Equation (3.13)), K is the number of images in the test set, and n is the dimension of $\mathbf{s}_k$. ANEES values close to one indicate that an estimator is *consistent*, that is, neither pessimistic nor optimistic about its confidence in the estimated quantity.

While the hyperparameter τ should in principle be tuned using a validation set, we omitted this step to keep our training procedure consistent with that of Ma et al. (2017). We note that the BCNN uncertainty estimates are affected by two significant factors: 1) variational inference is known to underestimate

[3] https://github.com/wanji/caffe-sl

predictive variance (Gal, 2016); and 2) we assume the observation noise is homoscedastic. As noted by Gal (2016), the BCNN can be made heteroscedastic by learning the model precision during training, but this extension is outside the scope of this work.

Data Partitioning We partitioned our data into training and testing sets using a leave-one-out approach based on temporally disjoint sequences of images. That is, given N sequences, the model tested on sequence i is trained with sequences $\{1, 2, ..., N\} \setminus i$. This process varied based on the dataset, and we discuss the specifics in the experimental discussion corresponding to each. In contrast to randomly holding out a subset of the data, this method minimizes the similarity of training and testing data for temporally correlated image streams.

3.4 Experiments

We conducted experiments both in simulation and on real-world data from urban and planetary analogue environments in order to assess the effectiveness of our method for correcting orientation drift using probabilistic measurements of the sun, and to compare the accuracy and effectiveness of Sun-BCNN to that of competing models for single-image sun detection (Lalonde et al., 2012; Clement et al., 2017b; Ma et al., 2017). Section 3.4.1 summarizes our experiments with an idealized sensor in a simulated environment using simulated sun measurements with varying degrees of artificial noise, while Sections 3.4.2 and 3.4.3 discuss our results for the KITTI odometry benchmark (Geiger et al., 2012; Geiger et al., 2013) and the Devon Island Rover Navigation Dataset (Furgale et al., 2012), respectively. Section 3.4.4 analyzes the sensitivity of Sun-BCNN models to cloud cover using the Oxford RobotCar dataset (Maddern et al., 2017), investigates the possibility of transfer learning between different urban environments, and between urban and planetary analogue environments, and discusses the impact of design choices with respect to mean and covariance computation for unit-norm 3D vectors. Table 3.1 summarizes the experiments presented in this chapter.

3.4.1 Simulation Experiments

We assessed the benefit of incorporating sun observations of varying quality by conducting a series of simulation experiments consisting of a stereo camera moving along loopy trajectories of varying shapes through a simulated field of point landmarks, with a single static directional landmark representing the sun. Figure 3.5 shows several such loopy trajectories.

We simulated the sun at 45° of zenith and an arbitrary azimuth angle, and corrupted observations of the ground truth sun vector with artificial noise such that the mean angular distance (a non-negative quantity computed from the dot product) between the ground truth and noisy sun vectors is 0°, 10°, 20°, or 30°. We label these conditions *GT-Sun-0*, *GT-Sun-10*, *GT-Sun-20*, and *GT-Sun-30*, respectively. We generated these noisy measurements by first sampling 3D vectors from an isotropic zero-mean multivariate Gaussian distribution, then adding these vectors to the ground truth sun vector, and finally normalizing the result to unit length.[4] We chose the covariance of this distribution to yield the desired average angular distance in each case. Note that although the distribution from which we sample noise

[4]Since the noise is isotropic, the projection onto the unit sphere makes this procedure equivalent to sampling 2D vectors in the local tangent space of the measurement vector (as perturbations along the radial direction are cancelled out), but is easier to implement.

Table 3.1: Summary of experiments in Chapter 3.

Section	Description	Dataset(s)
Section 3.4.1	Evaluation of VO accuracy using an idealized sensor and simulated sun measurements with varying levels of artificial noise.	Simulation
Section 3.4.2.1	Characterization of Sun-BCNN estimation errors and consistency in an urban environment.	KITTI
Section 3.4.2.2	Comparison of VO accuracy using simulated sun measurements, hand-engineered single-image sun detection, and learned single-image sun detection in an urban environment.	KITTI
Section 3.4.3.1	Characterization of Sun-BCNN estimation errors and consistency in a planetary analogue environment.	Devon Island
Section 3.4.3.2	Comparison of VO accuracy using a hardware sun sensor, hand-engineered single-image sun detection, and learned single-image sun detection in a planetary analogue environment.	Devon Island
Section 3.4.4.1	Evaluation of Sun-BCNN sensitivity to cloud cover.	Oxford
Section 3.4.4.2	Evaluation of Sun-BCNN model generalization across datasets.	KITTI, Devon Island, Oxford
Section 3.4.4.3	Comparison of methods for computing sample means and covariances of unit-norm sun direction vectors.	KITTI, Devon Island, Oxford

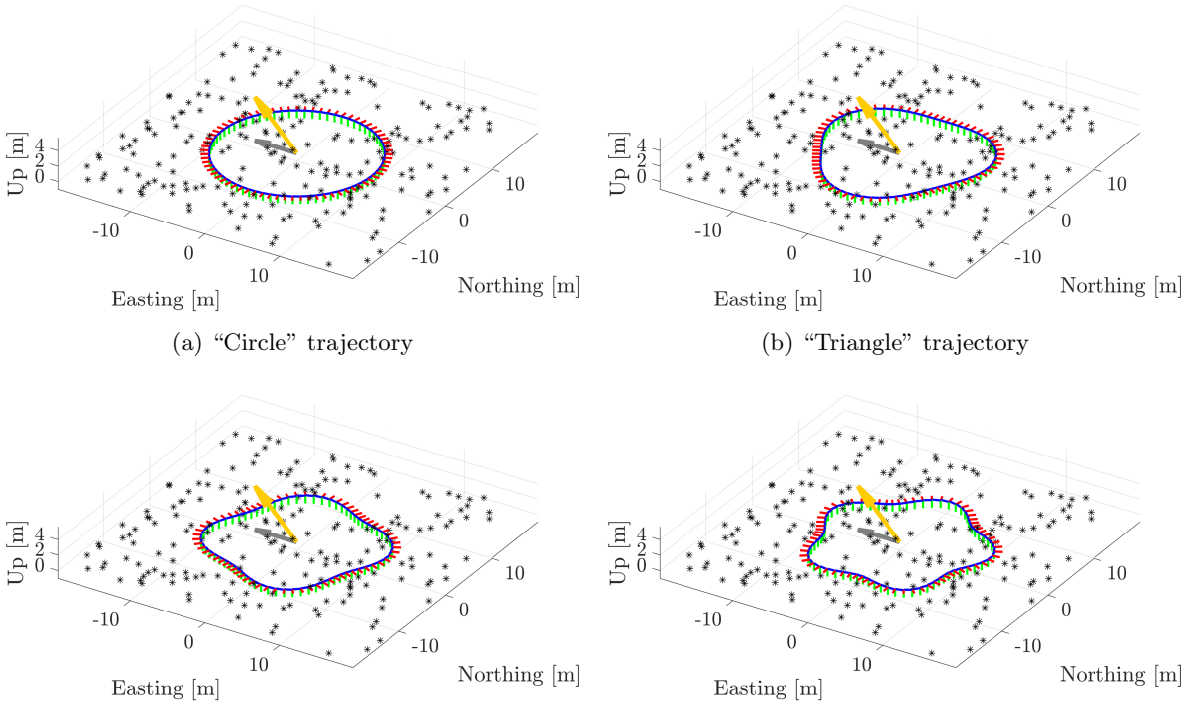
vectors is zero-mean, the average angular distances will not be zero-mean because angular distance is non-negative.

Our choice to add noise in $\mathbb{R}^3$ and re-normalize was motivated by the fact that this process yields approximately Gaussian error distributions over the azimuth and zenith error angles, which is an important property assumed by our VO pipeline to produce maximum likelihood motion estimates based on the fusion of multiple data sources. We note that these distributions are less Gaussian-like for larger covariances (due to the geometry of the unit 2-sphere) and for ground truth vectors near singularities (e.g., zero zenith).

We also experimented with sampling from a Von Mises-Fisher distribution (Fisher, 1953), which is approximately analogous to an isotropic Gaussian distribution that respects the geodesics on the unit 2-sphere. However, we observed that the resulting distributions on azimuth and zenith error were severely non-Gaussian, which violated the assumption of zero-mean Gaussian noise in our VO pipeline and interfered with our VO experiments.

Since our VO pipeline does not incorporate loop closures, the effects of drift in the VO solution can be clearly seen by examining individual loops in the camera trajectory. Figure 3.6 shows three loops from the "Circle" trajectory, demonstrating that the VO solution drifts significantly from the true trajectory by the 100th loop. Figure 3.7 plots the translational and rotational cumulative root mean squared error (CRMSE)[5] for this trajectory, which measures the growth in total estimation error over time. Figure 3.7c in particular highlights the significant effect of sun sensing on rotational error, where we see a clear progression in estimation error as the sun direction observations become more noisy. While the progression of translational error with respect to increasing observation noise is less clear (cf.

[5]CRMSE is computed as a running total of the RMSE over the entire trajectory.



(a) "Circle" trajectory

(b) "Triangle" trajectory

(c) "Square" trajectory

(d) "Star" trajectory

Figure 3.5: One loop of the "Circle", "Triangle", "Square", and "Star" trajectories, consisting primarily of translation and yaw rotation. Landmarks are shown as black asterisks, and the simulated sun direction is indicated with a yellow arrow along with its projection, in grey, on the EN-plane.

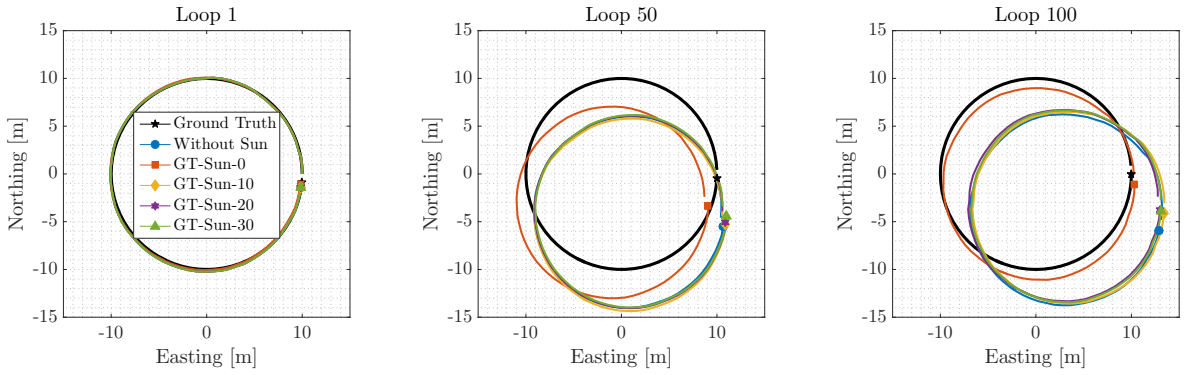
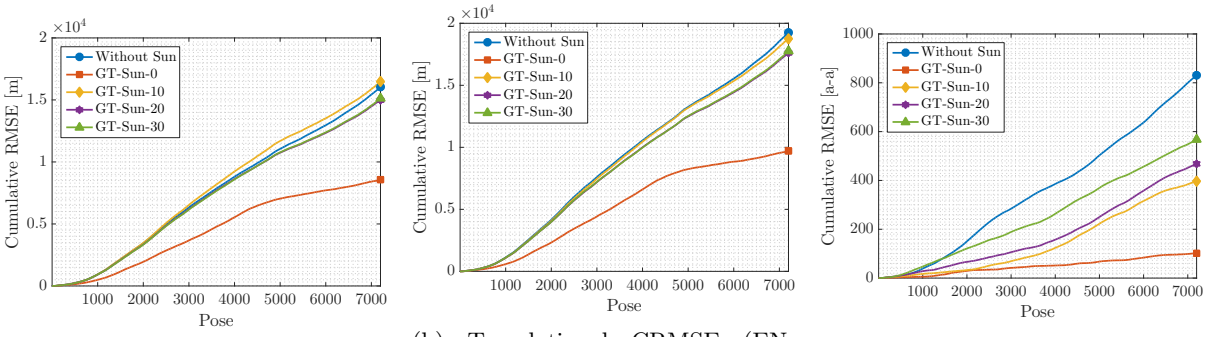


Figure 3.6: Motion estimates from selected segments of a 100-loop "Circle" trajectory with and without sun measurements corrupted by varying levels of artificial Gaussian noise. The effect of VO drift can be clearly seen, as well as the benefit of incorporating observations of a directional landmark such as the sun.



(a) Translational CRMSE (ENU)

(b) Translational CRMSE (EN-plane)

(c) Rotational CRMSE

Figure 3.7: Cumulative root mean squared error (CRMSE) of a simulated 100-loop circular trajectory both without sun corrections and with sun corrections corrupted by varying levels of artificial Gaussian noise. The accumulated estimation error is greatly reduced by incorporating observations of the sun, and the benefit decreases as these observations become noisier.

Figures 3.7a and 3.7b), we see that low-noise sun measurements nevertheless have a significant impact on translational error.

Table 3.2 shows that while all four simulation trajectories displayed consistent and predictable reductions in rotational average root mean squared error (ARMSE)[6], this was not always the case for translational ARMSE. This is because translational errors are only partially induced by rotational errors, with the remainder made up of 'sliding' motions orthogonal to the direction of travel. These non-rotational errors are highly dependent on the specific trajectory, where more or less of the observed feature tracks can be explained by a sliding motion instead of a rotation. Due to the coupling of translational and rotational errors, correcting for rotational error in such cases may actually make the translational error worse (e.g., on the "Triangle" sequence). While we did not implement this in our work, we speculate that incorporating an appropriate motion model into our VO formulation would significantly mitigate the impact of these errors by, for example, imposing a nonholonomic constraint on a ground vehicle or accounting for the dynamics of a quadcopter.

3.4.2 Urban Driving Experiments: The KITTI Odometry Benchmark

We investigated the performance of Sun-BCNN on the KITTI odometry benchmark training set (Geiger et al., 2012; Geiger et al., 2013), which consists of 21.6 km of urban driving data[7]. Importantly, the dataset includes 6-dof ground truth poses obtained from an accurate GNSS/INS tracking system, as well as calibrated transformations between this sensor and the colour stereo camera we use for sun estimation and VO in our experiments. This allowed us to create a training set of ground truth sun vectors for each image by querying a solar ephemeris model at each ground truth pose and rotating the resulting vector from the GNSS/INS frame $\underrightarrow{\mathcal{F}}_0$ (which is an ENU coordinate system) into the camera coordinate frame $\underrightarrow{\mathcal{F}}_k$. For each of our experiments, we trained Sun-BCNN on nine benchmark sequences and tested on the remaining one. This procedure is consistent with that of Ma et al. (2017), against whose Sun-CNN we directly compare, and allows us to evaluate each sequence using the maximum amount of training

[6] ARMSE is computed as the sum of the RMSE over the entire trajectory, divided by the number of poses.

[7] Because we rely on the first pose reported by the GNSS/INS system, we used the raw (rectified and synchronized) data corresponding to each odometry sequence. However, the raw sequence `2011_09_26_drive_0067` corresponding to odometry sequence `03` was not available on the KITTI website at the time of writing, so we omit sequence `03`.

Table 3.2: Comparison of translational and rotational average root mean squared errors (ARMSE) on simulated sequences, illustrating the impact of simulated sun measurements corrupted by varying levels of artificial Gaussian noise. The best result is highlighted in bold.

	Circle	**Triangle**	**Square**	**Star**
# Loops	100	100	100	100
Trans. ARMSE [m]				
Without Sun	2.22	2.00	2.33	1.41
GT-Sun-0	**1.19**	**1.62**	2.13	**0.75**
GT-Sun-10	2.29	2.07	**2.05**	1.32
GT-Sun-20	2.08	2.12	2.31	1.33
GT-Sun-30	2.10	1.95	2.16	1.38
Trans. ARMSE (EN-plane) [m]				
Without Sun	2.67	1.88	2.57	1.10
GT-Sun-0	**1.34**	**1.89**	2.56	**0.83**
GT-Sun-10	2.61	2.04	**2.26**	0.99
GT-Sun-20	2.44	2.03	2.57	0.88
GT-Sun-30	2.46	2.00	2.35	1.25
Rot. ARMSE ($\times\mathbf{10^{-3}}$) **[axis-angle]**				
Without Sun	115.32	144.56	107.27	111.19
GT-Sun-0	**14.10**	**113.58**	**59.21**	**30.69**
GT-Sun-10	55.22	115.03	75.62	39.17
GT-Sun-20	65.02	121.11	80.41	49.75
GT-Sun-30	78.73	145.22	100.91	72.39

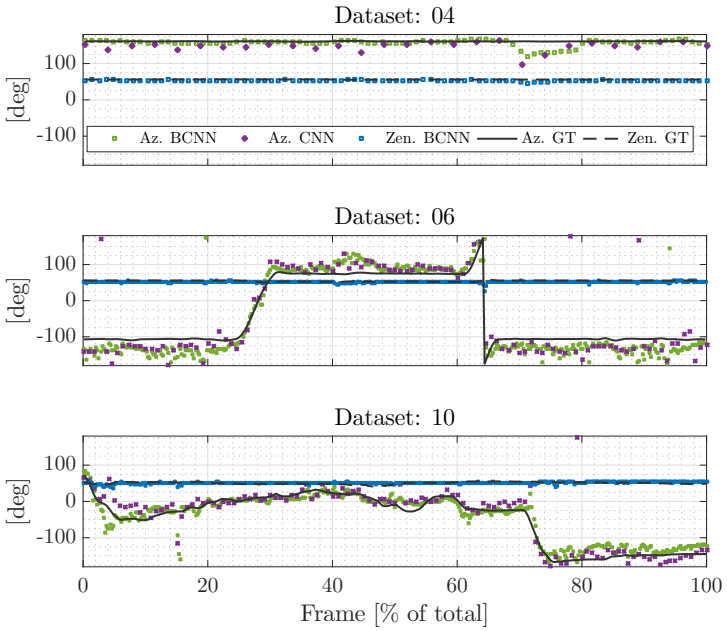


Figure 3.8: Azimuth (Sun-CNN and Sun-BCNN) and zenith (Sun-BCNN only) predictions over time for KITTI test sequences `04`, `06` and `10`. Sun-CNN is trained and tested on every tenth image, whereas Sun-BCNN is trained and tested on every image. In our VO experiments, we use the Sun-BCNN predictions of every tenth image to make a fair comparison.

data.

3.4.2.1 Sun-BCNN Test Results

Once trained, we analyzed the accuracy and consistency of the Sun-BCNN mean and covariance estimates. We obtained the mean estimated sun vector by sampling the network using Monte Carlo dropout (Equation (3.14)) with $N = 25$, using Equation (3.12) to compute the mean, and finally re-normalizing the resulting vector to preserve unit length. To obtain the required covariance on azimuth and zenith angles, we converted the sampled vector outputs to azimuth and zenith angles using Equation (3.7), and then computed the associated covariance matrix using Equation (3.13). We investigate the impact of this parameterization (as opposed to working in azimuth and zenith coordinates directly) in Section 3.4.4.3. As shown in Table 3.3, we chose a value for the model precision τ such that the Average Normalized Estimation Error Squared (ANEES) of each test sequence is close to one (i.e., the estimator is consistent).

Figure 3.8 shows three characteristic plots of the azimuth and zenith predictions over time, while Figures 3.9 and 3.10 plot the error distributions for zenith, azimuth, and angular distance for all ten KITTI odometry sequences. We see that the errors in azimuth and zenith are strongly peaked around zero and are reasonably well described by a Gaussian distribution, which are important properties assumed by our VO pipeline to produce maximum likelihood motion estimates based on the fusion of multiple data sources. Note that the error distribution in zenith is slightly biased towards negative values due to the presence of a long tail on the negative side of the mean. This is an artifact of the zenith-azimuth parameterization when the sun zenith is small (i.e., when the sun is high in the sky), since zenith angles are defined on $[0, \pi]$. In practice, we attempt to reduce the influence of the long negative tail by imposing

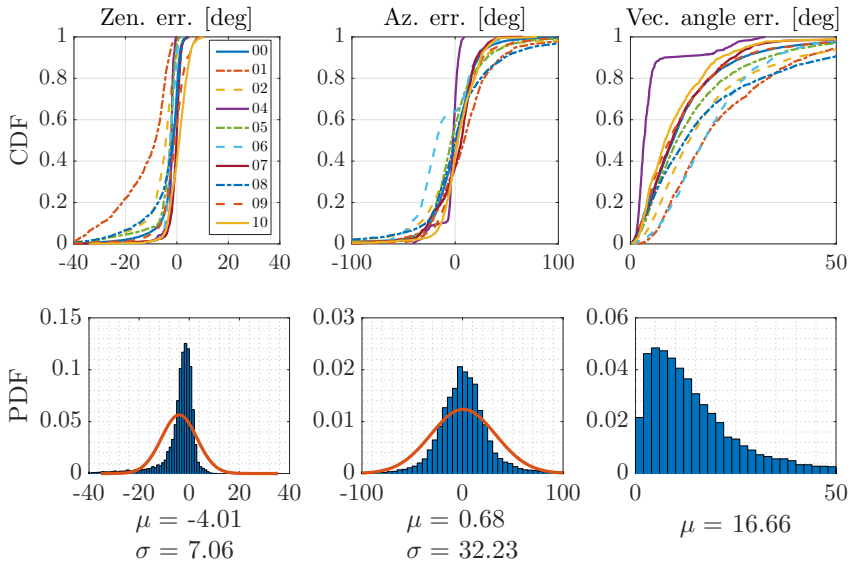


Figure 3.9: Distributions of zenith error, azimuth error, and angular distance for Sun-BCNN compared to ground truth for each test sequence in the KITTI dataset. *Top row*: Cumulative distributions of errors for each sequence individually. *Bottom row:* Histograms and Gaussian fits of aggregated errors.

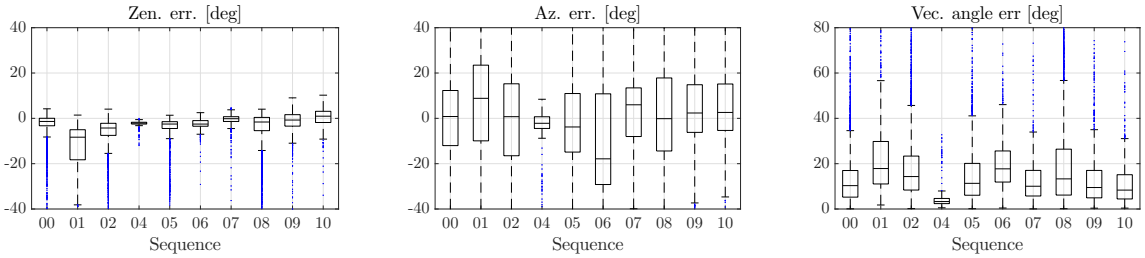


Figure 3.10: Box-and-whiskers plot of Sun-BCNN test errors on all ten KITTI odometry sequences (cf. Table 3.3).

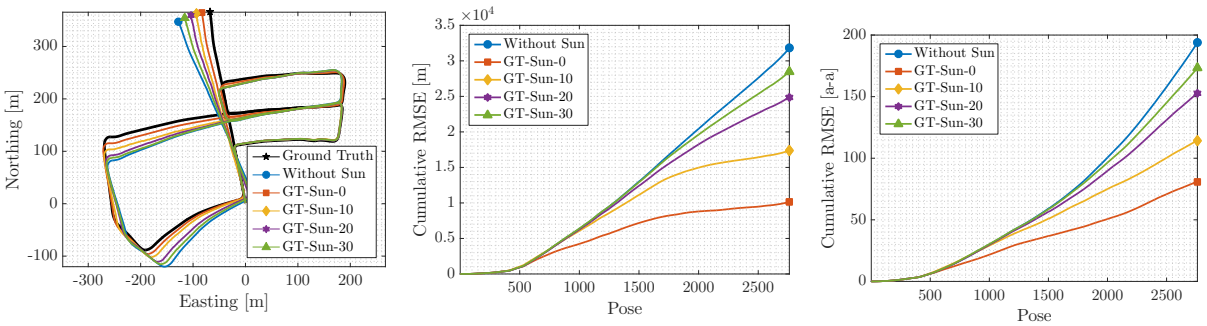
a robust Huber loss on the sun measurement errors in our optimization problem.

Table 3.3 summarizes the Sun-BCNN test errors numerically. Sun-BCNN achieved median vector angle errors of less than 15 degrees on every sequence except sequence `01` and `06`, which were particularly difficult in places due to challenging lighting conditions. It is interesting to note that sequences `00` and `06` also have higher than average ANEES values, which indicates that the estimator is overconfident in its estimates despite their low quality. We suspect this behaviour stemmed from the assumption of homoscedastic noise in the BCNN, which treats all input images as being equally amenable to sun estimation across the entire sequence.

Table 3.3: Test errors for Sun-BCNN on KITTI odometry sequences with estimates computed at every image.

	Zenith Error [deg]			**Azimuth Error [deg]**			**Vector Angle Error [deg]**			
Sequence	Mean	Median	Stdev	Mean	Median	Stdev	Mean	Median	Stdev	**ANEES**[1]
00	-2.59	-1.37	5.15	-0.33	0.81	25.61	13.56	10.31	13.14	1.00
01	-12.53	-8.31	10.33	8.95	8.83	33.67	22.16	17.85	15.00	1.38
02	-6.13	-4.26	7.38	-1.03	0.74	37.61	19.69	14.32	18.25	1.40
04	-2.42	-2.11	1.64	-3.89	-2.18	9.14	5.33	3.29	6.44	0.30
05	-4.31	-2.51	6.18	-0.74	-3.80	29.81	15.66	11.33	14.80	1.05
06	-2.48	-2.52	2.27	-12.22	-17.86	25.78	19.78	17.72	11.35	1.93
07	-0.69	-0.16	3.26	1.25	5.98	20.27	12.44	10.05	9.97	0.97
08	-4.46	-1.61	8.14	3.66	-0.14	41.73	19.90	13.30	19.59	1.04
09	-1.35	-0.75	5.60	4.78	2.36	23.84	13.09	9.48	12.66	0.73
10	0.59	0.95	3.90	3.64	2.61	19.15	11.23	8.34	9.83	1.08
All	-4.01	-2.26	7.06	0.68	0.53	32.23	16.66	12.08	15.91	-

[1] We compute Average Normalized Estimation Error Squared (ANEES) values with all sun directions that fall below a cosine distance threshold of 0.3 (relative to ground truth) and set $\tau^{-1} = 0.015$.



(a) VO and ground truth trajectories

(b) Translational CRMSE (EN-plane)

(c) Rotational CRMSE

Figure 3.11: VO results for KITTI odometry sequence 05 using simulated sun measurements at every tenth pose. We observe a clear progression in cumulative root mean squared error (CRMSE) in translation and rotation as noise in the simulated sun measurements increases.

3.4.2.2 Visual Odometry Experiments

We evaluated the influence of the estimated sun directions and covariances obtained from Sun-BCNN on the KITTI odometry benchmark using the sun-aided VO pipeline previously described. To place these results in context, we compared them against the results obtained using simulated sun measurements with varying levels of noise, the method of Lalonde et al. (2012) and its VO-informed variant (Clement et al., 2017b), and the Sun-CNN of Ma et al. (2017).

Simulated Sun Sensing In order to gauge the effectiveness of incorporating sun information in each sequence, and to determine the impact of measurement error, we constructed several sets of simulated sun measurements by computing ground truth sun vectors and artificially corrupting them with varying levels of zero-mean Gaussian noise. We obtained these ground truth sun vectors by transforming the ephemeris vector into each camera frame using ground truth vehicle poses. Using the same convention as our experiments with simulated trajectories, we created four such measurement sets with 0°, 10°, 20°, and 30° mean angular distance from ground truth.

Figure 3.11 shows the results we obtained using simulated sun measurements on sequence 05, in which the basic VO suffers from substantial orientation drift.[8] Incorporating absolute orientation information from the simulated sun sensor allowed the VO to correct these errors, but the magnitude of the correction decreased as sensor noise increases, consistent with the results of our simulation experiments. As shown in Table 3.4, which summarizes our VO results for all ten sequences, this was typical of sequences where orientation drift is the dominant source of error.

While the VO solutions for sequences such as 00 did not improve in terms of translational ARMSE, Table 3.4 shows that rotational ARMSE nevertheless improved on all ten sequences when low-noise simulated sun measurements are included. This implies that the estimation errors of the basic VO solutions for certain sequences were dominated by non-rotational effects, and that the apparent benefit of the Lalonde method on translational ARMSE in sequence 00 is likely coincidental.

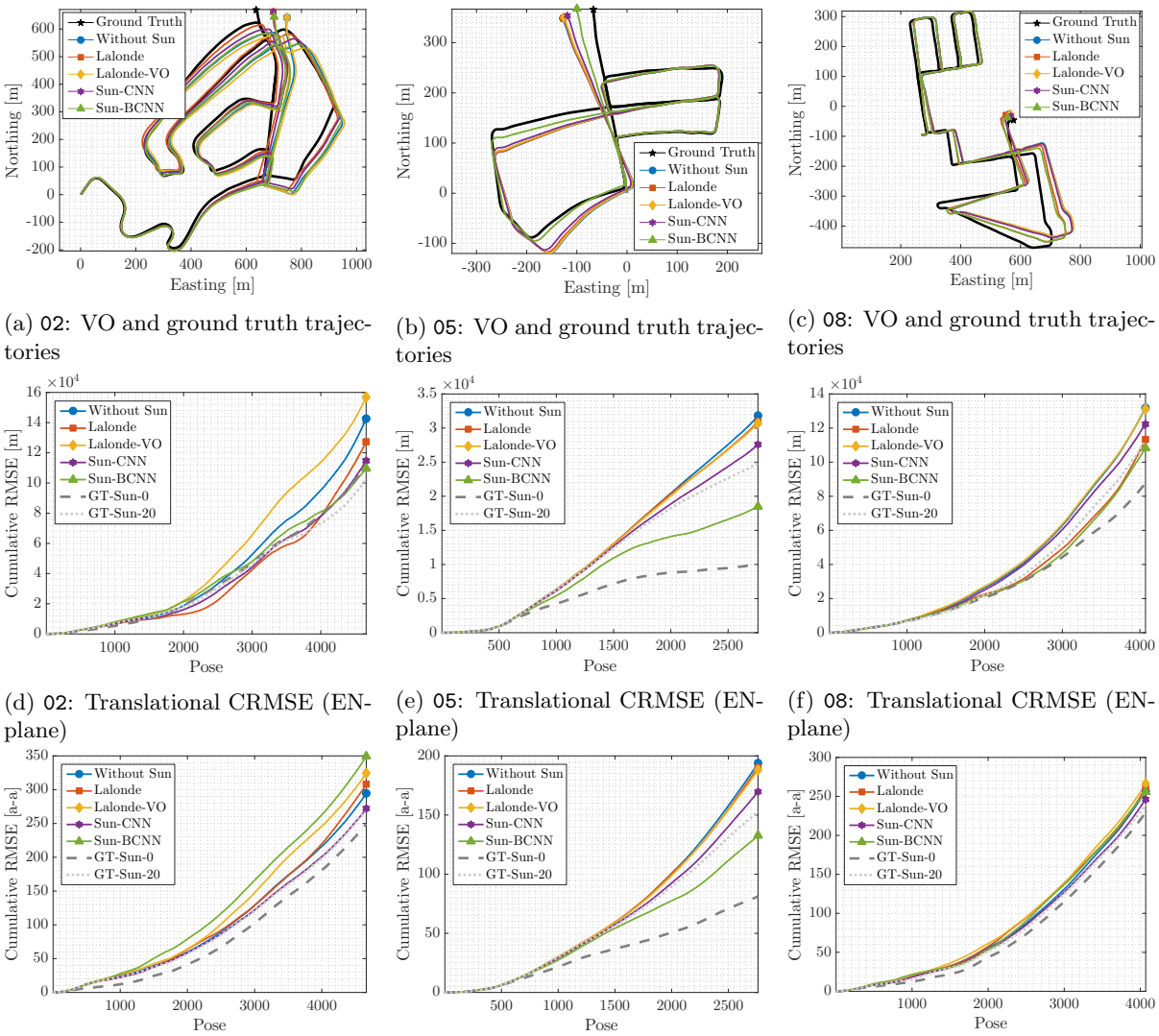
[8] In order to make a fair comparison to the Sun-CNN of Ma et al. (2017), who compute sun directions for every tenth image of the KITTI odometry benchmark, we subsample the sun directions obtained through each other method to match.

Figure 3.12: Visualization of Sun-BCNN predictions and associated ground truth sun directions on KITTI sequence 05. *Top two rows*: Sun-BCNN produces accurate predictions in a variety of azimuth values. *Bottom row*: Poor results occur rarely due to shadow ambiguities.

Vision-based Sun Sensing Figure 3.12 illustrates the behaviour of Sun-BCNN on six characteristic images from test sequence 05 by overlaying the Sun-BCNN predictions and associated ground truth sun directions for each image. The images in the top two rows both contain strong shadows which typically result in very accurate sun predictions. Conversely, the bottom row highlights two examples of rare situations where ambiguous shadows lead to very inaccurate predictions. As previously mentioned, we mitigated the influence of these outlier measurements by imposing a robust Huber loss on the sun measurement errors in our optimizer.

Figure 3.13 shows the results we obtained for sequences 02, 05, and 08 using the Sun-CNN of Ma et al. (2017), which estimates only the azimuth angle of the sun, our Bayesian Sun-BCNN which provides full 3D estimates of the sun direction as well as a measure of the uncertainty associated with each estimate, and the method of Lalonde et al. (2012) in its original and VO-informed (Clement et al., 2017b) forms, which provide 3D estimates of the sun direction without reasoning about uncertainty. A selection of results using simulated sun measurements are also displayed for reference. All four sun detection methods succeeded in reducing the growth of total estimation error on this sequence, with Sun-BCNN reducing both translational and rotational error growth significantly more than the other three methods. Both Sun-CNN and Sun-BCNN outperformed the two Lalonde variants, consistent with the results of Ma et al. (2017) and Clement et al. (2017b).

Table 3.4 shows results for all ten sequences using each method. With few exceptions, the VO results using Sun-BCNN achieved improvements in rotational and translational ARMSE comparable to those achieved using the simulated sun measurements with between 10 and 30 degrees average error. As previously noted, sequences such as 00 did not benefit significantly from sun sensing since rotational drift was not the dominant source of estimation error in these cases. Nevertheless, these results indicate that CNN-based sun sensing is a valuable tool for improving localization accuracy in VO – an improvement that comes without the need for additional sensors or a specially oriented camera to directly observe the sun.



(a) 02: VO and ground truth trajectories

(b) 05: VO and ground truth trajectories

(c) 08: VO and ground truth trajectories

(d) 02: Translational CRMSE (EN-plane)

(e) 05: Translational CRMSE (EN-plane)

(f) 08: Translational CRMSE (EN-plane)

(g) 02: Rotational CRMSE

(h) 05: Rotational CRMSE

(i) 08: Rotational CRMSE

Figure 3.13: VO results for KITTI odometry sequences 02, 05, and 08 using estimate sun directions at every tenth pose. *Top row*: Estimated and ground truth trajectories in the Easting-Northing (EN) plane. *Middle row*: Translational cumulative root mean squared error (CRMSE) in the EN-plane. *Bottom row*: Rotational CRMSE. Sun-BCNN significantly reduces the estimation error on sequence 05, while the Lalonde (Lalonde et al., 2012), Lalonde-VO (Clement et al., 2017b), and Sun-CNN (Ma et al., 2017) methods provide modest reductions in estimation error. The remaining sequences are less clear, but Sun-BCNN generally provides some benefit.

Table 3.4: Comparison of translational and rotational average root mean squared error (ARMSE) on KITTI odometry sequences with and without sun direction estimates at every tenth image. The best result (excluding simulated sun sensing) is highlighted in bold.

Sequence[1]	00	01[2]	02	04	05	06	07	08	09	10
Length [km]	3.7	2.5	5.1	0.4	2.2	1.2	0.7	3.2	1.7	0.9
Trans. ARMSE [m]										
Without Sun	4.33	198.52	28.59	2.48	9.90	3.35	4.55	28.05	10.44	5.54
GT-Sun-0	5.40	114.69	23.83	2.23	4.84	3.50	1.58	31.55	8.21	3.67
GT-Sun-10	4.85	123.84	25.34	2.45	5.84	2.80	2.94	28.47	8.65	4.81
GT-Sun-20	4.78	136.60	22.33	2.46	8.16	3.03	3.90	27.54	8.68	5.45
GT-Sun-30	4.83	157.14	27.30	2.48	8.93	3.44	4.62	26.73	10.10	5.28
Lalonde	**3.81**	200.34	28.13	**2.47**	9.88	3.36	4.61	29.70	10.49	**5.48**
Lalonde-VO	4.87	199.03	29.41	2.48	9.74	**3.30**	4.52	27.82	11.06	5.59
Sun-CNN	4.36	192.50	**26.58**	2.48	8.92	3.38	4.30	**26.99**	10.15	5.58
Sun-BCNN	4.44	**188.46**	26.89	2.48	**8.50**	4.10	**4.21**	27.71	**10.13**	5.61
Trans. ARMSE (EN-plane) [m]										
Without Sun	4.53	230.73	30.66	1.81	11.50	3.68	5.44	32.37	11.65	5.95
GT-Sun-0	3.41	136.76	24.12	1.46	3.67	3.96	1.80	21.51	7.77	3.71
GT-Sun-10	5.05	149.36	24.79	1.79	6.29	2.73	3.51	22.41	8.90	5.09
GT-Sun-20	5.14	164.37	22.04	1.80	9.01	3.13	4.66	27.58	8.86	5.81
GT-Sun-30	5.12	188.61	22.65	1.83	10.31	3.83	5.50	27.65	11.16	5.58
Lalonde	**3.95**	232.66	27.30	**1.81**	11.20	3.70	5.52	27.84	11.41	**5.87**
Lalonde-VO	5.38	231.33	33.68	1.82	11.13	**3.61**	5.42	32.24	12.41	6.00
Sun-CNN	4.56	224.91	24.65	1.82	9.99	3.74	5.16	30.09	11.21	5.99
Sun-BCNN	4.68	**220.54**	**23.58**	1.82	**6.70**	4.78	**5.05**	**26.59**	**10.97**	6.03
Rot. ARMSE ($\times 10^{-3}$) [axis-angle]										
Without Sun	23.88	185.30	63.18	12.97	70.18	23.24	49.96	63.13	26.77	21.54
GT-Sun-0	11.20	38.82	53.48	11.75	29.38	17.66	20.37	56.39	17.00	12.60
GT-Sun-10	17.05	64.51	58.78	12.86	41.47	18.90	34.05	54.89	19.71	14.26
GT-Sun-20	18.84	94.65	58.03	12.91	55.39	19.67	43.34	58.82	20.99	25.87
GT-Sun-30	23.40	121.21	57.79	13.01	62.73	23.96	49.92	56.74	25.63	20.15
Lalonde	**21.10**	188.06	66.02	**12.96**	69.00	23.27	50.49	64.22	26.27	**20.49**
Lalonde-VO	27.91	185.52	69.52	12.98	68.09	**22.79**	49.74	65.35	28.82	22.10
Sun-CNN	24.05	177.45	**58.32**	13.00	61.48	23.34	47.77	**60.55**	**26.19**	21.99
Sun-BCNN	26.96	**175.21**	75.02	13.00	**47.96**	23.80	**47.57**	62.85	26.29	20.85

[1] Because we rely on the timestamps and first pose reported by the GNSS/INS system, we use the raw (rectified and synchronized) sequences corresponding to each odometry sequence. However, the raw sequence `2011_09_26_drive_0067` corresponding to odometry sequence `03` was not available on the KITTI website at the time of writing, so we omit sequence `03` from our analysis.

[2] Sequence `01` consists largely of self-similar, corridor-like highway driving which causes difficulties when detecting and matching features using `libviso2`. The base VO result is of low quality, although we note that including global orientation from the sun nevertheless improves the VO result.

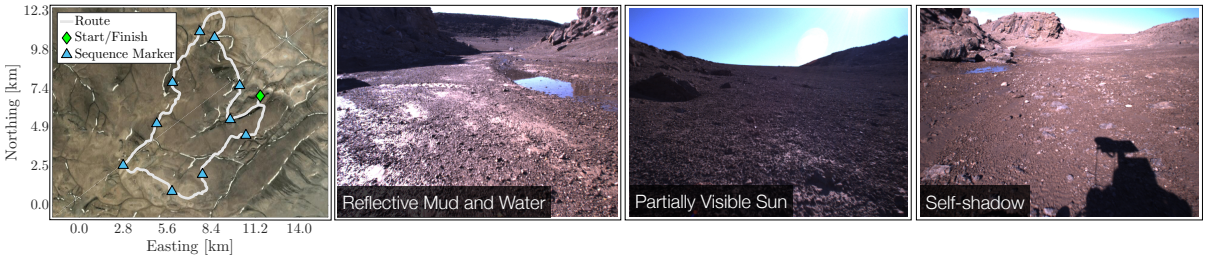


Figure 3.14: GNSS track and sample images from the Devon Island traverse, with the start of each sequence highlighted. The Devon Island dataset is conducive to visual sun sensing due to the presence of strong environmental shadows, reflective surfaces such as mud and water, occasionally visible sun, and self-shadowing by the sensor platform. (Map data: Google, DigitalGlobe)

3.4.3 Planetary Analogue Experiments: The Devon Island Rover Navigation Dataset

In addition to urban driving, we further investigated the usefulness of Sun-BCNN in the context of planetary exploration using the Devon Island Rover Navigation Dataset (Furgale et al., 2012), which consists of various sensor data collected using a mobile sensor platform traversing a 10 km loop on Devon Island in the Canadian High Arctic (Figure 3.14). The rugged landscape of Devon Island (Figure 3.14) is a significant departure from the structured urban environment of Karlsruhe. Unlike the KITTI odometry benchmark, the Devon Island dataset provides ground truth vehicle orientations for only a small number of images, which means that our previous method of generating ground truth sun vectors using ground truth poses is not applicable. However, the sensor platform used to collect the dataset was equipped with a hardware sun sensor and inclinometer, both of which were used by Lambert et al. (2012) to correct VO drift. Both the sun sensor and inclinometer provide information about the global vehicle orientation, however in our case we are interested in the minimal configuration of a single stereo camera with no auxiliary sensors, where global orientation information is extracted from the same image stream used to compute VO. Accordingly, we ignored the inclinometer in our experiments with the Devon Island dataset, and used the sun sensor measurements only to establish training targets for Sun-BCNN.

The Devon Island environment contains many features one might expect to be amenable to visual sun detection. As shown in Figure 3.14, the dataset contains strong environmental shadows, stretches of wet terrain featuring reflective mud and water, and some self-shadowing from the sensor platform itself. At times the sun is partially visible to the camera, although these images tend to be saturated and do not immediately allow for accurate localization of the sun in the image.

For the purposes of our experiments, we partitioned the dataset into 11 sequences of approximately 1 km each, chosen such that the full pose of the vehicle at the beginning of each sequence is available from the ground truth data (see Figure 3.14). In aggregate, the sequences contain approximately 13,000 poses with associated sun sensor measurements. We applied a similar training and testing procedure as for the KITTI dataset, with the exception that we withheld one sequence for validation and hyper-parameter tuning in addition to the sequence withheld for testing. This left nine sequences remaining to form the training sets for each test and validation pair.

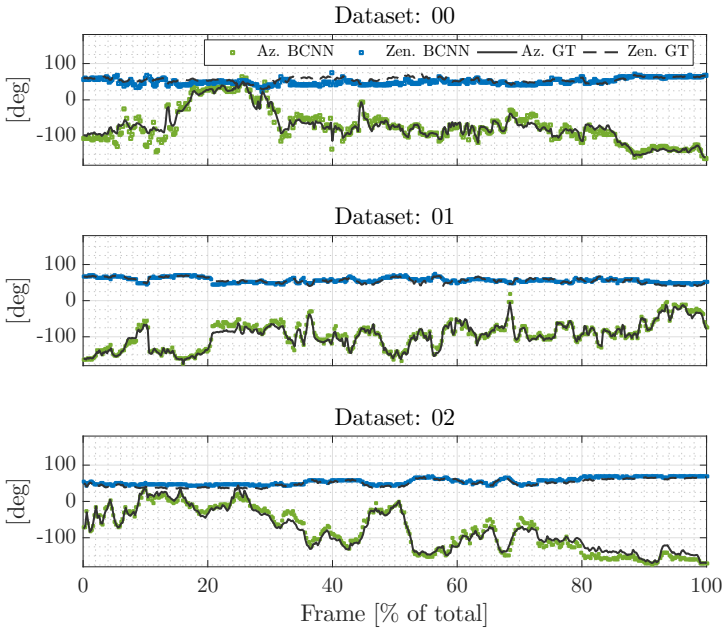


Figure 3.15: Sun-BCNN azimuth and zenith predictions over time for Devon Island test sequences `00`, `01` and `12`. Sun-BCNN was trained and tested on all frames (in our VO experiments, we used the Sun-BCNN predictions of every tenth image to make a fair comparison).

3.4.3.1 Sun-BCNN Test Results

As in our experiments with the KITTI odometry benchmark, we obtained the mean estimated sun vector by evaluating Equation (3.12) with $N = 25$ and re-normalizing the resulting vector to preserve unit length. To obtain the required covariance on azimuth and zenith angles, we again sampled the vector outputs, converted them to azimuth and zenith angles using Equation (3.7), and then applied Equation (3.13). As shown in Table 3.5, we chose a value for the model precision τ such that the Average Normalized Estimation Error Squared (ANEES) of each test sequence is close to one (i.e., the estimator is consistent).

Figure 3.15 shows three characteristic plots of the azimuth and zenith predictions over time, while Figures 3.16 and 3.17 plot the error distributions for zenith, azimuth, and angular distance for all 11 Devon Island odometry sequences. We see that the errors in azimuth and zenith are strongly peaked around zero and are better described by a Gaussian distribution than in the case of KITTI (cf. Figure 3.9), which as we previously mentioned are important properties assumed by our VO pipeline to appropriately fuse data. The distribution of zenith errors in the Devon Island dataset does not exhibit the same bias and long tail we observed in the KITTI dataset. This is likely because the sun is much lower in the sky (i.e., the zenith angle is further from zero) in the Devon Island dataset than in the KITTI dataset, so there is no clipping of the distribution near zero zenith.

Table 3.5 summarizes the test errors and ANEES of each sequence numerically, while Figures 3.16 and 3.17 plot the error distributions for zenith, azimuth, and angular distance for each sequence. Figure 3.15 shows three characteristic plots of the azimuth and zenith predictions over time. Sun-BCNN achieved median vector angle errors of less than 10 degrees on every sequence except sequence `08`. Consistent with the results we observed in the KITTI experiments, the sequences with the highest median vector angle error (sequences `02` and `08`) also have the highest ANEES values, again indicating that the

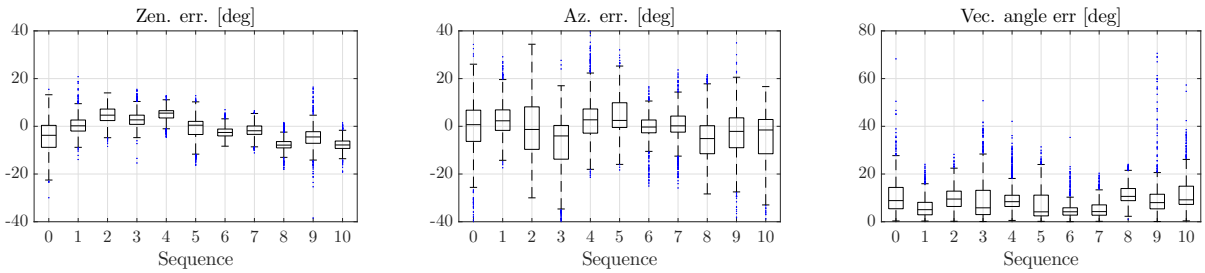


Figure 3.16: Box-and-whiskers plot of Sun-BCNN test errors on Devon Island sequences (cf. Table 3.5).

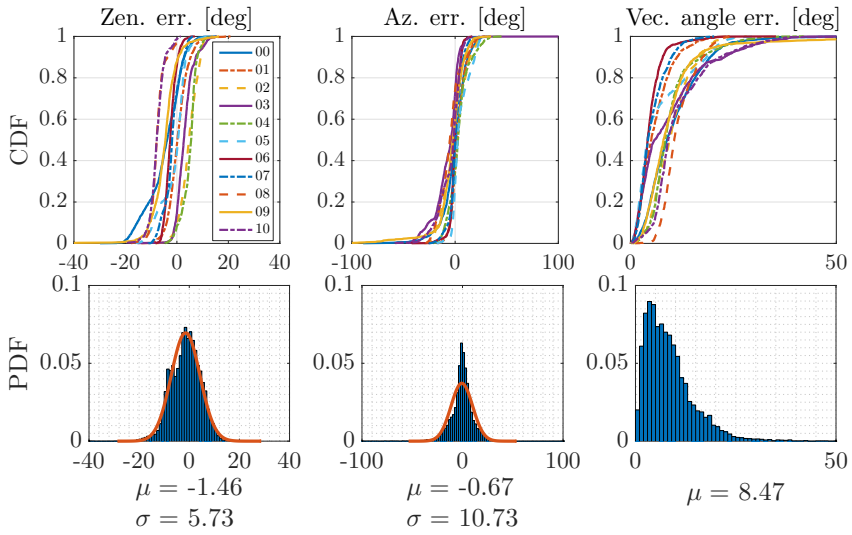


Figure 3.17: Distributions of zenith error, azimuth error, and angular distance for Sun-BCNN compared to ground truth over each Devon Island test sequence. *Top row*: Cumulative distributions of errors for each test sequence individually. *Bottom row:* Histograms and Gaussian fits of aggregated errors.

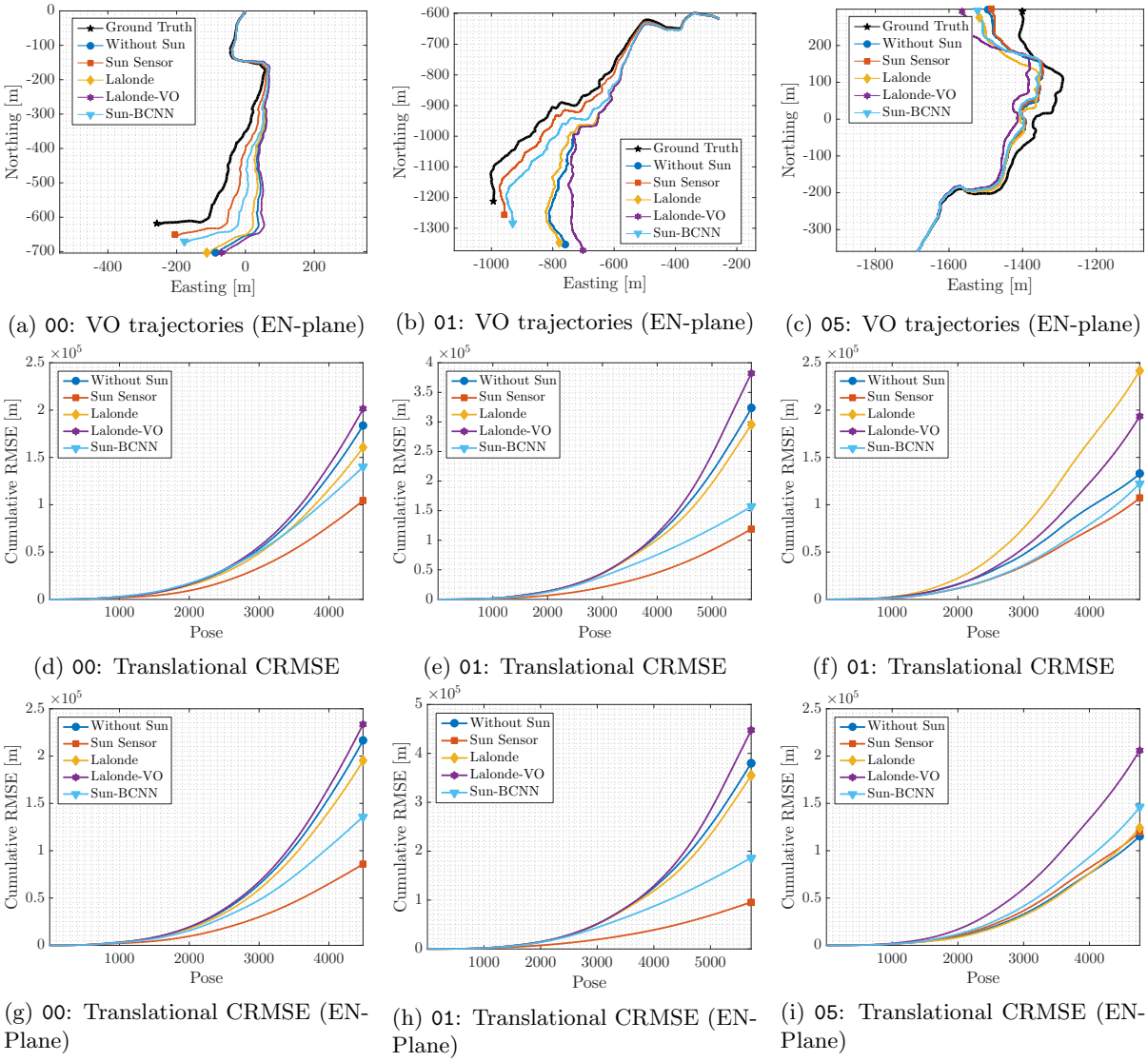
homoscedastic noise assumption is perhaps ill suited to this environment.

3.4.3.2 Visual Odometry Experiments

As in our KITTI benchmark experiments, we compare visual odometry results on each of our 11 test sequences both with sun-based orientation corrections and without. Notably, we do not report results using simulated sun measurements since we are unable to generate these measurements without ground truth vehicle poses for every image. We also do not report results using the Sun-CNN of Ma et al. (2017) since we do not have access to their model. However, we do compare the results obtained using Sun-BCNN to those obtained using the hardware sun sensor as well as the Lalonde (Lalonde et al., 2012) and Lalonde-VO (Clement et al., 2017b) methods.

Figure 3.18 shows sample VO results on three sequences from the Devon Island dataset using no sun measurements, the hardware sun sensor, Sun-BCNN, and the Lalonde variants. While the Lalonde methods struggle in this environment, Sun-BCNN yields significant improvements in VO accuracy, nearly on par with those obtained using the hardware sun sensor.

Table 3.6 summarizes these results numerically for all 11 sequences in the dataset. While the addition of sun sensing using either the hardware sensor or Sun-BCNN generally results in significant reductions in error, we note that in certain cases (e.g., sequence `05`), sun sensing has little or no impact on the VO result. We suspect that the translation errors in these cases are dominated by non-rotational effects, similarly to those observed in our experiments with the KITTI dataset, although it is difficult to be certain in the absence of rotational ground truth. As previously mentioned, the incorporation of a motion prior in the VO estimator would likely reduce the impact of these errors.



(a) 00: VO trajectories (EN-plane)

(b) 01: VO trajectories (EN-plane)

(c) 05: VO trajectories (EN-plane)

(d) 00: Translational CRMSE

(e) 01: Translational CRMSE

(f) 01: Translational CRMSE

(g) 00: Translational CRMSE (EN-Plane)

(h) 01: Translational CRMSE (EN-Plane)

(i) 05: Translational CRMSE (EN-Plane)

Figure 3.18: VO results for Devon Island sequences 00, 01, and 05 using estimated sun directions. *Top row*: Estimated and ground truth trajectories in the EN-plane. *Bottom rows*: Translational cumulative root mean squared error (CRMSE). Sun-BCNN significantly reduces the estimation error on sequences where the sun sensing has an impact (cf. Table 3.6).

Table 3.5: Test errors for Sun-BCNN on Devon Island sequences with estimates computed at every image.

	Zenith Error [deg]			**Azimuth Error [deg]**			**Vector Angle Error [deg]**			
Sequence	Mean	Median	Stdev	Mean	Median	Stdev	Mean	Median	Stdev	**ANEES**[2]
00	-4.77	-3.77	6.82	-0.65	0.69	12.41	10.48	8.86	6.96	1.27
01	0.47	0.21	3.91	2.96	2.31	7.01	5.97	5.06	4.01	0.59
02	4.66	4.68	3.52	-0.72	-1.32	11.78	10.02	9.51	4.76	1.37
03	3.09	2.70	3.41	-7.47	-4.03	12.88	9.39	5.83	8.75	1.11
04	4.93	5.53	2.90	3.27	2.72	10.09	9.78	8.41	5.60	0.89
05	-1.01	0.46	4.97	5.26	2.46	8.23	7.19	4.15	6.60	0.92
06	-2.45	-2.58	2.23	-0.23	-0.30	5.07	4.72	4.17	3.16	0.31
07	-1.80	-1.87	3.28	0.47	0.20	6.45	5.23	4.25	3.38	0.41
08	-7.46	-7.88	2.85	-4.93	-5.14	10.30	11.61	10.63	3.96	1.33
09	-4.72	-4.46	5.27	-3.91	-2.13	14.61	9.90	8.02	8.56	0.86
10	-7.69	-7.82	2.92	-4.81	-1.54	10.80	11.79	9.19	7.52	0.91
All	-1.46	-1.23	5.73	-0.67	-0.14	10.73	8.47	7.15	6.31	-

[1] We compute Average Normalized Estimation Error Squared (ANEES) values with all sun directions that fall below a cosine distance threshold of 0.3 (relative to ground truth) and set $\tau^{-1} = 0.01$.

Table 3.6: Comparison of average root mean squared error (ARMSE) on Devon Island sequences with and without sun direction estimates using both a hardware sun sensor and vision-based methods. The best result using a vision-based method is bolded.

Sequence	00	01	02	03	04	05	06	07	08	09	10
Length [km]	0.9	1.1	1.0	1.0	0.9	1.0	1.1	1.0	0.9	0.7	0.6
Trans. ARMSE [m]											
Without Sun	40.93	56.51	41.58	42.04	30.52	27.82	58.91	40.04	47.22	11.39	12.94
Hardware Sun Sensor	23.26	20.79	9.79	22.03	30.79	22.47	24.14	29.59	47.97	6.26	8.50
Lalonde	35.77	51.74	53.32	47.00	39.55	50.70	94.77	59.37	**45.78**	10.03	16.23
Lalonde-VO	44.83	66.91	44.17	59.84	42.87	40.62	52.16	36.04	50.52	11.34	16.74
Sun-BCNN	**31.17**	**27.45**	**16.00**	**26.02**	**29.34**	**25.70**	**33.43**	**32.25**	50.80	**4.27**	**14.92**
Trans. ARMSE (EN-plane) [m]											
Without Sun	48.20	66.49	43.58	45.92	31.08	24.23	43.01	22.33	40.85	9.30	15.59
Hardware Sun Sensor	19.13	16.74	8.99	21.18	28.27	25.08	29.27	21.76	28.89	5.14	9.70
Lalonde	43.45	62.03	36.21	49.44	**20.13**	**26.13**	53.22	**18.10**	35.62	6.01	18.45
Lalonde-VO	52.05	78.26	40.20	59.09	50.12	43.28	53.62	42.71	49.99	11.74	20.17
Sun-BCNN	**30.28**	**32.65**	**9.62**	**14.32**	33.26	30.62	**36.44**	23.18	**13.53**	**4.45**	**14.75**

3.4.4 Sensitivity to Training Conditions and Measurement Parameterization

In this section we analyze the sensitivity of our model to cloud cover, investigate the possibility of model transfer between urban and planetary analogue environments, and examine the impact of different methods for computing the mean and covariance of a norm-constrained vector on the accuracy and consistency of the estimated sun directions.

3.4.4.1 Cloud Cover

Given that both the KITTI and Devon Island datasets were collected in sunny conditions, it is natural to wonder whether and to what extent Sun-BCNN is affected by cloud cover. As shown in Figure 3.4, Sun-BCNN relies in part on shadows and other local illumination variations to estimate the direction of the sun. Since the diffuse nature of daylight in cloudy conditions tends to soften shadows and other shading variations, one might expect Sun-BCNN to perform worse in cloudy conditions. Accordingly, we investigated the effect of cloud cover on Sun-BCNN using selected sequences from the Oxford RobotCar Dataset (Maddern et al., 2017), which consists of 1000 km of urban driving along similar routes but in varying weather conditions and at varying times over the course of a year.

Procedure We selected three sequences collected within a two hour period on the same day (namely `2014-07-14-14-49-50`, `2014-07-14-15-16-36`, and `2014-07-14-15-42-55`), which consist of the same route observed under different lighting conditions. Figure 3.19 presents sample images from each of these sequences, which we label *Overcast*, *Sun-Cloud A*, and *Sun-Cloud B*, respectively. To evaluate the performance of Sun-BCNN in each of these conditions, we partition each sequence into a randomly selected set of training (80%), validation (10%) and test (10%) images, and then train and test Sun-BCNN on each of the nine train-test permutations.

Results Figure 3.20 shows the results of these experiments with box and whisker plots for zenith, azimuth, and vector angle errors while Table 3.7 summarizes the results numerically. We obtained the most accurate test predictions using the model trained on *Sun-Cloud B*, the sequence with the least amount of cloud cover. Notably, this model produced vector angle errors on the *Overcast* test set that were lower than those trained with its own *Overcast* training set. Moreover, we note that the *Sun-Cloud A* model achieved similar test errors when applied to the *Sun-Cloud B* test set as when applied to the *Overcast* test set. Similarly, the *Sun-Cloud B* model achieved similar test errors when applied to the *Sun-Cloud A* test set as when applied to the *Overcast* test set. From this we can conclude the following: 1) that Sun-BCNN can still perform well in the presence of cloud cover; and 2) that training in environments illuminated by strong directional light (i.e., sunny conditions) can significantly improve sun estimation accuracy in a variety of test conditions, even in the absence of highly directional illumination.

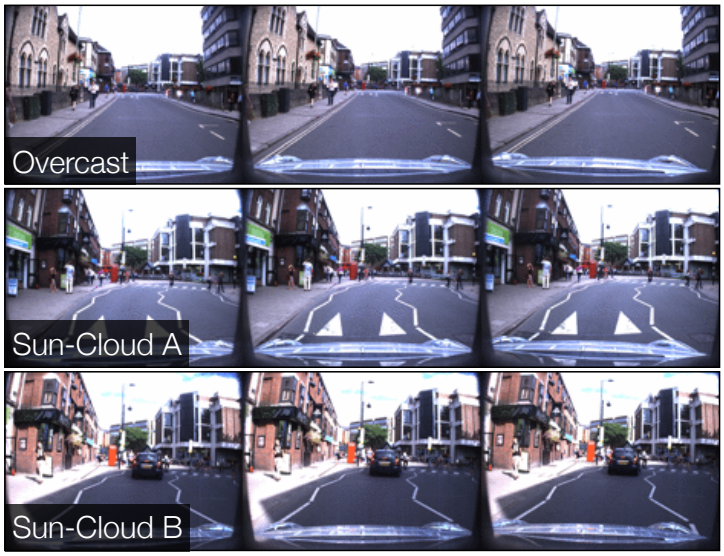


Figure 3.19: Sample images of approximately the same location taken from three different Oxford RobotCar sequences we used to investigate the effect of cloud cover on Sun-BCNN.

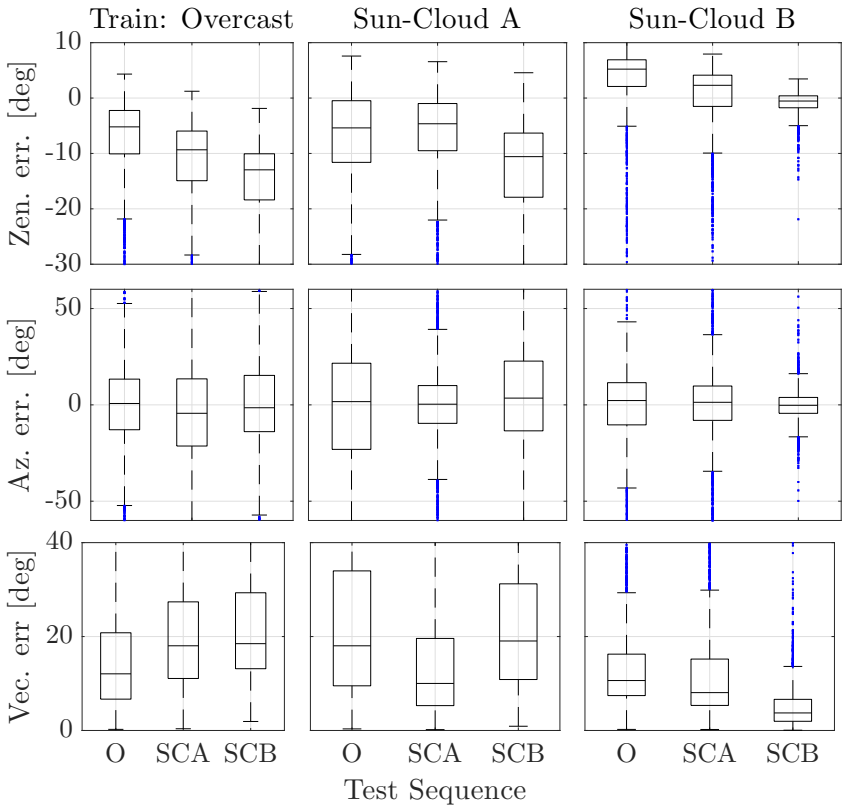


Figure 3.20: Box-and-whiskers plot for zenith, azimuth, and vector angle errors for nine different combinations of train-test sequences taken from the Oxford RobotCar dataset. Each column corresponds to a different training sequence, and each plot contains three different test sequences. In the bottom legend, we use the labels O: *Overcast*, SCA: *Sun-Cloud A*, SCB: *Sun-Cloud B*.

3.4.4.2 Model Generalization

It may also be natural to ask how well a Sun-BCNN model trained in an urban environment performs in a planetary analogue environment and vice versa. This would provide some indication of whether the model generalizes to new environments or if a philosophy of place-specific excellence (e.g., the place-specific visual features of McManus et al. (2014)) is more appropriate for the task of illumination estimation.

Procedure We attempted to answer this question by creating three larger datasets from combinations of the sequences used in our previous experiments:

1. KITTI odometry sequences `00 - 10`;
2. Devon Island sequences `00 - 10`; and
3. the previously discussed *Overcast*, *Sun-Cloud A*, and *Sun-Cloud B* sequences from the Oxford RobotCar dataset.

We randomly partitioned each dataset into training (90%) and test (10%) sets. We then trained three separate Sun-BCNN models on each training set, and evaluated each trained model on each of the three test sets.

Results Figure 3.21 shows the results of these experiments with box and whisker plots for zenith, azimuth, and vector angle errors while Table 3.8 summarizes the results numerically. We see that none of the three models generalize well to environments other than the one in which they were trained, yielding large and significantly biased test errors. We note, however, that the Oxford model was the least egregious offender, and speculate that this may be because the Oxford sequences contain significantly more training images than the other two datasets (approximately 3 times as many as the KITTI odometry benchmark and 5 times as many as the Devon Island dataset).

A possible explanation for the poor generalization of these models is the fact that each dataset was collected using different cameras with different optical properties and parameter settings. We believe these differences affect Sun-BCNN's ability to recover an accurate estimate of a three dimensional direction vector, since metrically important quantities such as the principal point and focal length of the sensor can vary significantly from camera to camera. Furthermore, differences in dynamic range may also significantly affect the ability of Sun-BCNN to treat shading variations consistently. An interesting avenue for future work may be to investigate ways of reformulating Sun-BCNN to be invariant to differences in camera intrinsics, and determine the impact of such a modification on model generalization between environments.

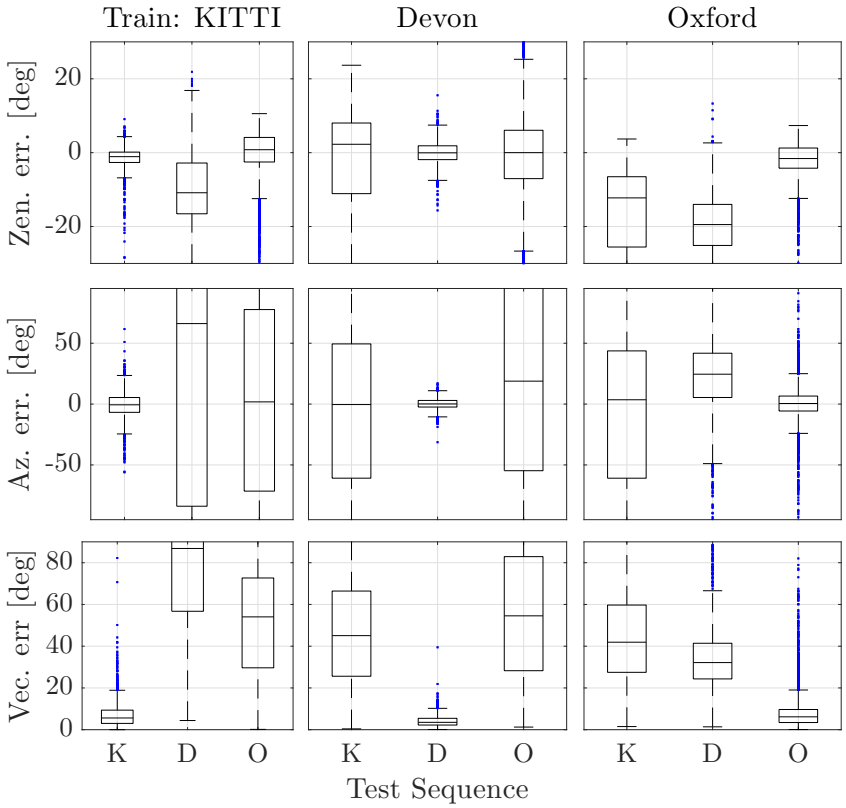


Figure 3.21: Box-and-whiskers plot for zenith, azimuth, and vector angle errors for nine different combinations of train-test datasets. Each column corresponds to a different training sequence, and each plot contains three different test sequences. In the bottom legend, we use the labels K: KITTI, D: Devon Island, O: Oxford. All three models produce large biased errors when applied to other datasets, likely due to variations in optical properties and parameter settings across cameras.

Table 3.7: Test errors for Sun-BCNN on three different Oxford RobotCar sequences collected on the same day with different lighting conditions.

		Zenith Error [deg]			**Azimuth Error [deg]**			**Vector Error [deg]**		
Train	**Test**	Mean	Median	Std.	Mean	Median	Std.	Mean	Median	Std.
Overcast[1]	Overcast	-7.12	-5.20	7.04	-0.66	0.72	29.36	15.22	12.06	11.73
	Sun-Cloud A	-11.58	-9.34	7.94	-5.71	-4.37	37.21	21.19	18.03	14.07
	Sun-Cloud B	-15.23	-12.96	8.00	0.05	-1.49	38.83	23.36	18.49	15.05
Sun-Cloud A[2]	Overcast	-7.17	-5.39	9.05	-0.67	1.68	51.27	23.66	18.03	18.11
	Sun-Cloud A	-6.49	-4.64	7.88	0.29	0.35	27.42	14.31	10.02	12.75
	Sun-Cloud B	-12.89	-10.58	8.94	1.87	3.51	40.41	23.45	19.06	16.75
Sun-Cloud B[3]	Overcast	3.34	5.22	6.46	-0.32	2.24	26.07	13.95	10.63	11.32
	Sun-Cloud A	-0.14	2.30	7.36	-1.08	1.34	28.54	13.76	8.06	14.60
	Sun-Cloud B	-0.84	-0.54	2.07	-0.36	-0.22	9.00	5.11	3.73	5.13

[1] `2014-07-14-14-49-50` [2] `2014-07-14-15-16-36` [3] `2014-07-14-15-42-55`

Table 3.8: Test errors for Sun-BCNN on different training and test datasets.

		Zenith Error [deg]			**Azimuth Error [deg]**			**Vector Error [deg]**		
Train	**Test**	Mean	Median	Std.	Mean	Median	Std.	Mean	Median	Std.
KITTI	KITTI	-1.49	-1.08	2.99	-0.64	-0.60	11.46	7.16	5.61	6.23
	Devon Island	-9.27	-10.86	9.97	26.78	66.15	113.23	81.32	86.82	33.48
	Oxford	-0.02	0.80	6.59	-0.44	1.81	91.30	52.39	54.05	29.46
Devon Island	KITTI	-2.37	2.27	14.30	-5.58	-0.38	78.01	48.16	45.06	27.85
	Devon Island	-0.08	-0.05	3.20	0.20	0.12	5.52	4.24	3.52	2.96
	Oxford	-1.35	0.00	11.57	17.12	18.85	96.86	55.52	54.55	29.88
Oxford	KITTI	-17.05	-12.25	13.19	-6.94	3.55	77.70	44.66	41.91	23.00
	Devon Island	-20.07	-19.47	9.81	20.92	24.56	45.52	35.16	32.15	16.07
	Oxford	-1.96	-1.59	4.60	0.19	0.48	15.08	8.08	6.16	7.68

Table 3.9: Comparison of Sun-BCNN prediction errors from different mean estimation methods.

		Zenith Error [deg]			**Azimuth Error [deg]**			**Vector Error [deg]**		
Sequence	**Mean**	Mean	Med.	Std.	Mean	Med.	Std.	Mean	Med.	Std.
KITTI	M-I	-1.50	-1.06	2.96	-0.56	-0.47	11.52	7.16	5.52	6.27
	M-II	-1.06	-0.76	2.44	-0.30	-0.37	30.18	11.49	5.95	18.60
Devon Island	M-I	-0.07	0.02	3.18	0.19	0.27	5.76	4.22	3.55	3.04
	M-II	0.04	0.09	3.17	1.11	0.26	24.62	9.19	4.05	20.22
Oxford	M-I	-1.97	-1.66	4.59	0.20	0.51	15.31	8.12	6.10	7.74
	M-II	-1.45	-1.27	3.95	-1.58	0.11	34.46	13.18	6.76	19.24

3.4.4.3 Mean and Covariance Computation

In our formulation, Sun-BCNN outputs a sample of unit-norm 3D vectors. Due to the unit-norm constraint, it is not immediately clear how to apply Equations (3.12) and (3.13) to calculate the sample mean and covariance. In this section we empirically evaluate two possible procedures for each computation using the previously discussed combined datasets for KITTI, Devon Island, and Oxford.

Mean We investigated two different methods for computing the mean of the sampled sun vectors, which we refer to as *Method I* (M-I) and *Method II* (M-II).

1. In *Method I* (used in this work), we first evaluate Equation (3.12) directly on the constrained unit vectors produced by N stochastic passes through the BCNN. We then re-normalize the resulting mean vector to enforce unit length, and convert it to azimuth and zenith angles using Equation (3.7).

2. In *Method II*, we first convert each of the N unit vectors to azimuth and zenith angles using Equation (3.7). We then evaluate Equation (3.12) on the angles themselves to obtain the mean in zenith-azimuth coordinates.

We evaluated both methods using the same combined datasets and partitioning scheme as in the transfer learning experiment previously presented. Table 3.9 presents the zenith, azimuth, and vector errors for the two mean computation methods. *Method I* produces lower vector errors and smaller standard deviations in azimuth and zenith on all three datasets.

Covariance We further investigated two different covariance computation methods, which we also refer to as *Method I* (C-I) and *Method II* (C-II).

1. In *Method I*, we first evaluate Equation (3.13) directly on the constrained unit vectors produced by N stochastic passes through the BCNN, yielding a 3×3 covariance. We then compute a 2×2 covariance on azimuth and zenith by propagating the 3×3 covariance through a linearized Equation (3.7).

2. In *Method II* (used in this work), we first convert each of the N unit vectors to azimuth and zenith angles, and then evaluate Equation (3.13) on the angles themselves.

Table 3.10: Comparison of ANEES values for different mean and covariance estimation methods.

Sequence	Covariance	Mean	ANEES
KITTI	C-I	M-I	0.95
		M-II	5.10
	C-II	M-I	1.40
		M-II	0.87
Devon Island	C-I	M-I	1.29
		M-II	10.05
	C-II	M-I	0.50
		M-II	0.85
Oxford	C-I	M-I	1.50
		M-II	2.14
	C-II	M-I	1.30
		M-II	0.89

Using the same datasets and partitioning scheme described in Section 3.4.4.2, we evaluated covariances on the test sets corresponding to each of the three models. To control for the effect of tuning the model precision τ, we replaced the diagonal elements of each covariance matrix with the diagonal elements of the empirical covariance corresponding to the entire test set (computed based ground truth azimuth and zenith errors). We then compared the consistency of the cross-correlations of each method (i.e., the off-diagonal components of the covariance matrix) by computing ANEES values over each model's corresponding test set using both mean computation methods. Table 3.10 lists the ANEES values produced by each method of covariance computation when paired with each mean computation method. *Method I* covariances produced better ANEES values when paired with *Method I* mean estimation, but *Method II* covariances paired well with either mean estimation scheme.

3.5 Conclusions and Future Work

This chapter presented Sun-BCNN, a Bayesian CNN trained to estimate the direction of the sun from a single RGB image in which the sun may not be visible. By leveraging the principled uncertainty estimates of the BCNN, we incorporated the model into a stereo VO pipeline and demonstrated significant reductions in error growth over 21.6 km of urban driving data from the KITTI odometry benchmark and a further 10 km traverse from the Devon Island Rover Navigation Dataset, achieving median test errors of approximately 12° and 7°, respectively. We further demonstrated Sun-BCNN's ability to deal with cloud cover on the Oxford RobotCar Dataset, analyzed the effect of model transfer between environments and cameras, and compared two methods of computing the mean and covariance of a norm-constrained vector. Although we applied Sun-BCNN to VO, we stress that it is equally suitable as a means of injecting global orientation information into other egomotion estimators (e.g., visual-inertial odometry).

Possible avenues for future work include incorporating an explicit motion model into the VO pipeline in order to better connect rotational and translational motion, amending the BCNN to produce temporally consistent estimates by leveraging the sequential nature of the test data (e.g., using a Recurrent Neural Network), and exploring ways in which the BCNN can account for different camera models to improve the generalizability of the trained models. Future research might also focus on regressing other

orientation cues such as the local gravity vector, or directly estimating the orientation of the camera.

3.6 Novel Contributions

Our main contributions can be summarized as follows:

1. We incorporated software-based visual sun detection as a 'pseudo-sensor' in a visual odometry pipeline, allowing for the extraction of global orientation information from the existing image stream and the correction of orientation drift in the motion estimate;

2. We demonstrated that Sun-BCNN, a Bayesian CNN with dropout layers after each convolutional and fully-connected layer, can achieve state-of-the-art accuracy on single-image sun detection at test time, and out-performs competing methods based on hand-engineered visual features;

3. We presented extensive experimental results on over 30 km of visual navigation data in urban and planetary analogue environments showing that indirect visual sun sensing significantly improves the accuracy of visual odometry over long trajectories;

4. We analyzed the sensitivity of Sun-BCNN to cloud cover, camera and environment changes, and measurement parameterization; and

5. We released an open-source implementation of Sun-BCNN, available at https://github.com/utiasSTARS/sun-bcnn.

3.7 Associated Publications

- Clement, L., Peretroukhin, V., and Kelly, J. (2017b). Improving the accuracy of stereo visual odometry using visual illumination estimation. In *Proceedings of the 2016 International Symposium on Experimental Robotics*, Springer Proceedings in Advanced Robotics, pages 409–419, Tokyo, Japan. Springer International Publishing.

- Peretroukhin, V., Clement, L., and Kelly, J. (2017). Reducing drift in visual odometry by inferring sun direction using a bayesian convolutional neural network. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2035–2042, Singapore.
  Note: Peretroukhin and Clement contributed equally to this work and jointly claim first authorship.

- Peretroukhin, V., Clement, L., and Kelly, J. (2018). Inferring sun direction to improve visual odometry: A deep learning approach. *International Journal of Robotics Research*, 37(9):996–1016.
  Note: Peretroukhin and Clement contributed equally to this work and jointly claim first authorship.

3.8 Associated Video

- Video summary of Sun-BCNN as presented at the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore:
  https://www.youtube.com/watch?v=c5XTrq3a2tE

Chapter 4

Canonical Appearance Transformations

Direct visual localization has enjoyed a resurgence in popularity with the increasing availability of cheap mobile computing power. The competitive accuracy and robustness of direct methods compared to state-of-the-art feature-based methods, as well as their natural ability to yield dense maps, makes them an appealing choice for a variety of mobile robotics applications. However, direct methods remain brittle in the face of appearance change due to their underlying assumption of photometric consistency, which is commonly violated in practice. In this chapter, we propose to mitigate this problem by training a deep convolutional encoder-decoder network to transform images of a scene such that they correspond to a previously-seen user-selected appearance condition, which we refer to as the *canonical* appearance. This approach is motivated by the observation that convolutional neural networks (CNNs) are well-suited to modelling environmental illumination, as illustrated in Chapter 3. We validate our method in multiple environments and illumination conditions using high-fidelity synthetic RGB-D datasets, and integrate the trained models into a direct visual localization pipeline, yielding improvements in visual odometry (VO) accuracy through time-varying illumination conditions, as well as improved metric localization performance under illumination change, where conventional methods fail. We further provide a preliminary investigation of transfer learning from synthetic to real environments in a localization context, and compare our approach against analytical image transformations commonly used to improve the robustness of direct methods to illumination change.

4.1 Motivation

Recently, direct visual localization algorithms such as those of Newcombe et al. (2011), Engel et al. (2015), Omari et al. (2015), and Whelan et al. (2016), which compare pixel intensities directly rather than tracking and matching abstracted features, have become popular due to their competitive accuracy compared to state-of-the-art indirect (feature-based) methods[2], their robustness to effects such as motion blur and camera defocus (Newcombe et al., 2011), and their natural ability to yield dense maps of an environment, which may be useful for higher-level tasks. With the increasing availability of cheap,

[2]See, e.g., the KITTI odometry leaderboard: http://www.cvlibs.net/datasets/kitti/eval_odometry.php.

compact computing power, direct visual localization algorithms are now capable of running online on a CPU (Forster et al., 2014; Engel et al., 2015; Omari et al., 2015).

Despite their successes, a significant disadvantage of direct methods is their underlying assumption of photometric consistency, that is, the assumption that the observed brightness or colour of objects in a scene remains constant through space and time. In practice, this assumption makes direct localization brittle in environments subject to time-varying illumination (e.g., changing shadows) or containing non-Lambertian materials (e.g., reflective surfaces), as well as in situations where camera parameters such as exposure and white balance may vary automatically in response to local scene conditions. This brittleness is especially problematic in long-term autonomy applications, where there is a need to localize reliably under significant appearance change.

While some have attempted to circumvent this difficulty by treating visual localization as an end-to-end learning problem (e.g., Kendall et al. (2015) and Costante et al. (2016)), such end-to-end methods have yet to prove as accurate or robust as state-of-the-art methods based on well established geometric and probabilistic modelling (Cadena et al., 2016). On the other hand, analytical models of appearance must often make approximations or assumptions that are frequently violated in practice (e.g., photometric consistency), or require detailed knowledge of the geometry, illumination, and material properties of the environment (Whelan et al., 2016; Kasper et al., 2016).

In this chapter we propose a hybrid solution for direct localization under varying illumination conditions that combines a conventional frame-to-keyframe localization pipeline with a learned image transformation that corrects for unmodelled effects such as illumination change. Rather than modelling illumination directly, we leverage existing sources of image data and recent work on image-to-image translation (Isola et al., 2017) to train a deep convolutional encoder-decoder network (Hinton and Salakhutdinov, 2006; Ronneberger et al., 2015) that learns to transform images of a scene that has undergone illumination change such that they correspond to a *canonical appearance* of the scene (i.e., a previously-seen reference condition). We refer to this learned transformation as a *canonical appearance transformation* (CAT). This approach is motivated by the observation that deep convolutional networks are effective tools for modelling environmental illumination, as demonstrated in Chapter 3.

Using high-fidelity synthetic RGB-D datasets, we demonstrate that our method yields significant improvements in visual odometry (VO) accuracy under time-varying illumination, as well as improved tracking performance in keyframe-based localization under conditions of severe illumination change, where conventional methods often fail. We compare our approach against analytical transformations commonly used to improve the robustness of direct methods to illumination change (Park et al., 2017), and further provide a preliminary investigation of transfer learning from synthetic to real environments in a localization context. An open-source implementation of our method using PyTorch (Paszke et al., 2017) is available at `https://github.com/utiasSTARS/cat-net`.

4.2 Related Work

Illumination robustness in visual localization has been previously studied from the perspective of illumination invariance, with methods such as those of McManus et al. (2014), Paton et al. (2017) and Clement et al. (2017a) making use of hand-crafted image transformations to improve feature matching over time. Similarly, affine models and other analytical transformations such as the image gradient and census transform have been used to improve the robustness of direct visual localization to illumination

change (Engel et al., 2015; Park et al., 2017). However, there has been little work on using machine learning techniques to generate such models from data.

The use of machine learning as a complement to model-based approaches has met with considerable success in the field of optimal control (e.g., Levine et al. (2016), Li et al. (2017)), and has recently gained a foothold in the domain of state estimation. Kendall et al. (2015) and Costante et al. (2016) learn end-to-end estimators based on deep neural networks, while others such as Handa et al. (2016), Haarnoja et al. (2016), and Peretroukhin and Kelly (2018) combine deep networks with traditional estimation machinery. Still other work has focused on training deep models to extract illumination information from images, which can be used to improve the performance of conventional localization pipelines (Peretroukhin et al., 2017; Ma et al., 2017).

Our work is related to the field of image-based rendering, which aims to synthesize new views of a scene by blending existing images (Shum and Kang, 2000; Fitzgibbon et al., 2005). In particular, our work bears resemblances to that of Flynn et al. (2016), who generate synthetic imagery using a convolutional neural network (CNN). In contrast, our goal is not to learn an image synthesis pipeline to be queried at arbitrary poses, but rather to learn a correction to the appearance of a single image taken from a fixed pose.

Our work is most similar to concurrent work by Gomez-Ojeda et al. (2018) and Porav et al. (2018). Gomez-Ojeda et al. (2018) train deep networks to enhance the temporal consistency and gradient information of image streams captured in environments with high dynamic range. In such environments, the main source of appearance change is the camera itself as it automatically modulates its imaging parameters in response to the local brightness of a static environment. In contrast, our method is concerned with improving localization under *environmental* illumination change, and is equally applicable to visual odometry (VO) and visual localization tasks. Porav et al. (2018) adopt a similar approach to ours for applying image-to-image translation to cross-appearance localization, incorporating adversarial learning (Goodfellow et al., 2014) and a cycle-consistency loss (Zhu et al., 2017) to partially train their models with unpaired image data. However, Porav et al. (2018) consider only the problem of translating between discrete appearance categories (e.g., day to night), whereas our approach is to learn a many-to-one mapping translating a continuum of appearance conditions to a single privileged condition. Furthermore, they do not investigate the usefulness of their approach for direct localization methods, focussing instead on sparse feature-based localization.

4.3 Methodology

Our goal in this work is to improve visual localization under varying appearance conditions by learning a transformation that maps images onto a privileged *canonical appearance* condition. We apply this transformation as an image pre-processing step in a direct visual localization pipeline based on concepts discussed in Chapter 2. Figure 4.1 pictorially summarizes our approach at a high level. Section 4.3.1 describes in detail the localization pipeline used in this work, while Section 4.3.2 discusses our approach to learning canonical appearance transformations using deep convolutional encoder-decoder networks. Section 4.4 describes our experimental setup and summarizes our results.

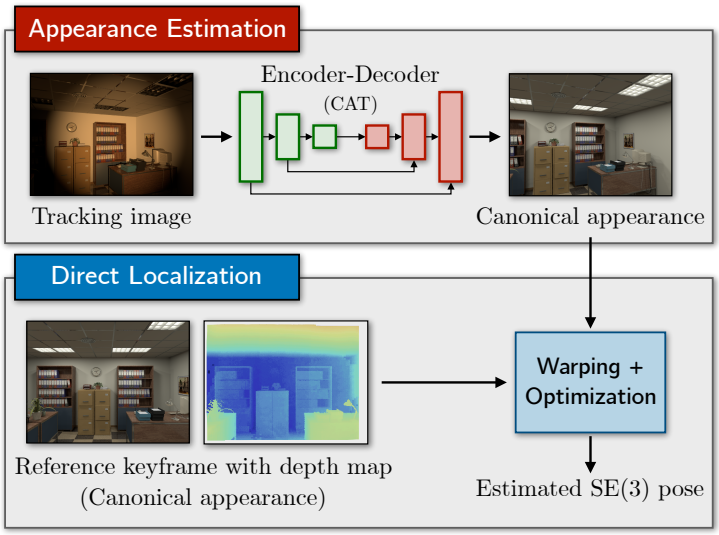


Figure 4.1: We train a deep convolutional encoder-decoder network to estimate the *canonical appearance* of a scene given an image captured under different illumination conditions, and use the transformed imagery in a direct visual localization pipeline to estimate the 6-dof pose of the camera under illumination change.

4.3.1 Direct Visual Localization

We adopt a keyframe-based direct visual localization pipeline similar to that proposed by Omari et al. (2015), which is suitable for use with stereo or RGB-D cameras. Our method uses an inverse compositional warping operation to implicitly establish correspondences between a reference image (keyframe) and a tracking (live) image, and minimizes the photometric error between corresponding pixels to estimate the relative pose of the camera. Keyframes are created whenever the translational or rotational distance between the tracking image and the active keyframe exceed a preset threshold. As a result of tracking live images against keyframes, as opposed to frame-to-frame tracking, our method is locally drift-free in the neighbourhood of the keyframe.

4.3.1.1 Observation Model

The observation model used in this work mirrors the model discussed in Section 2.2, and is repeated here for the reader's convenience. Assuming that our images have been undistorted and rectified through an appropriate calibration, we can approximate our camera by a pinhole model with focal lengths f_u, f_v and principal point (c_u, c_v). Thus, our (noiseless) observation model mapping 3D point $\mathbf{p} = [\,p_x \quad p_y \quad p_z\,]^T$ onto image coordinates $\mathbf{u} = [\,u \quad v\,]^T$ and depth map $\mathbf{D}$ is given by

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{D}(\mathbf{u}) \end{bmatrix} = \mathbf{g}(\mathbf{p}) = \begin{bmatrix} f_u p_x / p_z + c_u \\ f_v p_y / p_z + c_v \\ p_z \end{bmatrix}, \tag{4.1}$$

and the inverse mapping is given by

$$\mathbf{p} = \mathbf{g}^{-1}(\mathbf{u},\ \mathbf{D}(\mathbf{u})) = \mathbf{D}(\mathbf{u}) \begin{bmatrix} (u - c_u)/f_u \\ (v - c_v)/f_v \\ 1 \end{bmatrix}. \tag{4.2}$$

Note that $\mathbf{D}$ is easily replaced by a disparity map in the case of stereo vision, with only minor modifications to the model (cf. Equations (2.93) and (2.95)).

Using Equations (4.1) and (4.2), we can then map the image coordinates $\mathbf{u}_r$ of a reference image $\mathbf{I}_r$ onto the warped image coordinates $\mathbf{u}'_t$ of a tracking image $\mathbf{I}_t$ given the reference depth map $\mathbf{D}_r$:

$$\mathbf{u}'_t = \mathbf{g}\left(\mathbf{T}_{t,r}\ \mathbf{g}^{-1}\left(\mathbf{u}_r,\ \mathbf{D}_r(\mathbf{u}_r)\right)\right) \tag{4.3}$$

where $\mathbf{T}_{t,r} \in \mathrm{SE}(3)$ is the 6-dof pose of the tracking image relative to the reference image (i.e., the pose we want to estimate). Note that through a slight abuse of notation we treat the Cartesian and homogeneous coordinates of $\mathbf{p}$ interchangeably. Finally, we can compute a reconstruction $\mathbf{I}'_r$ of the reference image by sampling the tracking image:

$$\mathbf{I}'_r(\mathbf{u}_r) = \mathbf{I}_t(\mathbf{u}'_t). \tag{4.4}$$

Compared to a forward compositional warping function that reconstructs the tracking image based on the reference image, this inverse compositional approach (Baker and Matthews, 2004) has the advantage of resolving potential collisions caused by the many-to-one mapping of reference image coordinates onto tracking image coordinates (e.g., due to occlusions). It also has the advantage of requiring a depth or disparity map for the reference image only, which can reduce the computational burden of the algorithm.

4.3.1.2 Photometric (In)consistency

Direct visual localization typically relies on the assumption of *photometric consistency* to compute the error terms to be minimized. In other words, we assume that the observed brightness and colour of the scene stays constant across the reference and tracking images. In our inverse compositional formulation, this assumption can be expressed as

$$\mathbf{I}_r(\mathbf{u}) = \mathbf{I}'_r(\mathbf{u}) + \mathbf{n}_I, \tag{4.5}$$

where $\mathbf{I}'_r$ is the reconstruction of reference image $\mathbf{I}_r$ based on the current pose estimate $\mathbf{T}_{t,r}$, and $\mathbf{n}_I$ is zero-mean Gaussian noise with covariance $\mathbf{R}_I$.

In practice, the observed brightness of a scene may vary for a variety of reasons. For example, modern cameras will automatically adjust their gain and white balance parameters in response to the local brightness of the scene. While it is possible to modulate these parameters (Zhang et al., 2017) or estimate their effect using an affine model (Engel et al., 2015; Park et al., 2017) or calibrated camera response function (Engel et al., 2018), the response of the camera is not the only source of variation. Crucially, the illumination of an otherwise static scene may vary with time, or the materials in the scene may exhibit non-Lambertian reflectance. These effects are difficult to model analytically (although some attempts have been made in this context, e.g., by Whelan et al. (2016) and Kasper et al. (2016)) and

require detailed knowledge of the geometry, illumination, and material properties of the scene.

To account for variations in observed brightness, we can generalize Equation (4.5) so that it has the form

$$(\mathbf{f} \circ \mathbf{I}_r)(\mathbf{u}) = (\mathbf{g} \circ \mathbf{I}'_r)(\mathbf{u}) + \mathbf{n}_I, \tag{4.6}$$

where $\circ$ denotes function composition. In other words, we wish to find functions $\mathbf{f}(\cdot)$ and $\mathbf{g}(\cdot)$ that maximize the photometric consistency of the two images for a given $\mathbf{T}_{t,r}$. In this work, we choose $\mathbf{f}(\cdot) \equiv \mathbf{g}(\cdot)$, and learn an approximation of $\mathbf{f}(\cdot)$ from data. Specifically, we learn an $\mathbf{f}(\cdot)$ that transforms both images such that they correspond to a chosen *canonical appearance*, such as static diffuse illumination. This formulation captures two problematic cases for direct localization: first, it provides a means of enhancing the temporal consistency of the image stream, which can improve the accuracy and robustness of VO under time-varying illumination; and second, it allows us to create a map of an environment under nominal illumination conditions, then relocalize against it under different conditions.

4.3.1.3 Relative Motion Estimation

With $\mathbf{f}(\cdot) \equiv \mathbf{g}(\cdot)$, we can compute the per-pixel error terms from Equation (4.6) as

$$\mathbf{e}(\mathbf{u}) = (\mathbf{f} \circ \mathbf{I}_r)(\mathbf{u}) - (\mathbf{f} \circ \mathbf{I}'_r)(\mathbf{u}) \tag{4.7}$$

and the Jacobian of $\mathbf{e}$ with respect to a left perturbation $\boldsymbol{\epsilon}_{t,r}$ in $\mathbf{T}_{t,r}$ as

$$\frac{\partial \mathbf{e}(\mathbf{u})}{\partial \boldsymbol{\epsilon}_{t,r}} = -\frac{\partial (\mathbf{f} \circ \mathbf{I}'_r)(\mathbf{u})}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{p}} \frac{\partial \mathbf{p}}{\partial \boldsymbol{\epsilon}_{t,r}}. \tag{4.8}$$

Note that we do not need to differentiate $\mathbf{f}(\cdot)$ to obtain $\dfrac{\partial (\mathbf{f} \circ \mathbf{I}'_r)(\mathbf{u})}{\partial \mathbf{u}}$, as we can directly compute the image gradients of $\mathbf{f}(\mathbf{I}'_r)$ with respect to the image coordinates using the 3×3 Sobel-Feldman operators in Equation (2.112).

With these quantities in hand we can compute an estimate $\hat{\mathbf{T}}_{t,r}$ of the relative camera pose by using Gauss-Newton optimization (Sections 2.1.2.3 and 2.1.2.4) to solve the nonlinear least-squares problem,

$$\hat{\mathbf{T}}_{t,r} = \underset{\mathbf{T}_{t,r}}{\operatorname{argmin}} \sum_{\mathbf{u}} \mathbf{e}^T \mathbf{R}_e^{-1} \mathbf{e}, \tag{4.9}$$

where $\mathbf{R}_e$ is the covariance of $\mathbf{e}$ and we have omitted the dependence on $\mathbf{u}$ for notational convenience. As $\mathbf{R}_e$ encapsulates the noise properties of both the camera sensor and the depth sensor (or disparity map computation), it can vary from pixel to pixel. Accordingly, we estimate $\mathbf{R}_e$ as

$$\mathbf{R}_e = \mathbf{R}_I + \mathbf{G}_D \mathbf{R}_D \mathbf{G}_D^T, \tag{4.10}$$

where $\mathbf{R}_I$ is the covariance of $\mathbf{n}_I$ in Equation (4.6), $\mathbf{R}_D$ is the per-pixel covariance of the depth map,

$$\mathbf{G}_D = \frac{\partial \mathbf{e}}{\partial \mathbf{D}} = -\frac{\partial (\mathbf{f} \circ \mathbf{I}'_r)}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{p}} \frac{\partial \mathbf{p}}{\partial \mathbf{D}} \tag{4.11}$$

is the Jacobian of the per-pixel error with respect to the depth map, and we have again omitted the

dependence on $\mathbf{u}$.

In line with previous work (Engel et al., 2015), we apply a robust Huber loss function to Equation (4.9) to mitigate the effect of outliers and false correspondences, and solve the modified problem using Gauss-Newton optimization with the method of iteratively re-weighted least-squares (Section 2.1.2.5) to handle the robust loss function.

4.3.1.4 Keyframe Mapping and Localization

Our direct localization pipeline operates in both mapping (VO) and localization modes in a similar vein to topometric visual teach-and-repeat navigation (Paton et al., 2017; Clement et al., 2017a; Gridseth and Barfoot, 2019), where the camera follows a similar trajectory during both mapping and localization phases. As the camera explores the environment in mapping mode, we generate a graph of posed keyframes with corresponding image and depth data, creating new keyframes when the translational or rotational distance between the most recent keyframe pose and the current tracking pose exceeds a preset threshold. We do not incorporate any loop closures in the graph. In localization mode, the system is initialized with an active keyframe and an initial guess of the camera pose, and continuously identifies and localizes against the nearest keyframe (in the Euclidean sense).

4.3.1.5 Practical Considerations

The cost function described in Equation (4.9) is highly non-convex, and special care must be taken to avoid local minima. Like Newcombe et al. (2011), Engel et al. (2015), and Omari et al. (2015), we adopt a coarse-to-fine optimization scheme in which we iteratively re-solve Equation (4.9) at multiple scales in a Gaussian image pyramid to improve the efficiency and convergence radius of the problem. We traverse the pyramid from lowest to highest resolution, using the solution at each scale as the initial guess to the next. Following the advice of Newcombe et al. (2011), we optimize only for rotation at the lowest resolution. Similarly to Engel et al. (2015), we consider only pixels whose gradient magnitude exceeds a certain threshold, which allows us to smoothly trade off computation time against information usage (since high-gradient regions contain the most information). Finally, we make the approximation

$$\frac{\partial(\mathbf{f} \circ \mathbf{I}'_r)(\mathbf{u})}{\partial \mathbf{u}} \approx \frac{\partial(\mathbf{f} \circ \mathbf{I}_r)(\mathbf{u})}{\partial \mathbf{u}}, \tag{4.12}$$

under the assumption of small camera motion, which allows us to compute the image Jacobian for keyframes only (Omari et al., 2015).

Since we are assuming small frame-to-frame camera motion in our localization pipeline, we further make the approximation (cf. Equation (4.4))

$$(\mathbf{f} \circ \mathbf{I}'_r)(\mathbf{u}_r) \approx (\mathbf{f} \circ \mathbf{I}_t)(\mathbf{u}'_t). \tag{4.13}$$

This allows us to evaluate $\mathbf{f}(\cdot)$ only once, as a pre-processing step, on the reference and tracking images, prior to undertaking any warping operation. As a result, we can significantly improve the efficiency of the localization pipeline by avoiding the need to compute $\mathbf{f}(\mathbf{I}'_r)$ at every iteration of the pose optimization (which involves evaluating a large neural network) as well as the associated overhead costs of repeated data transfer between GPU and host machine.

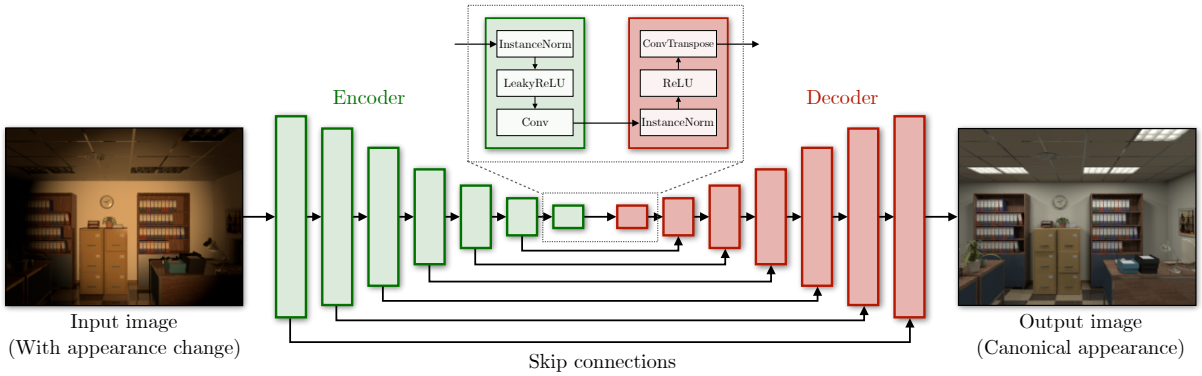


Figure 4.2: Our network architecture is a U-Net based on the model of Isola et al. (2017), consisting of seven encoder blocks (green) and seven decoder blocks (red), each sharing information with their counterpart blocks, which takes as input a $256 \times 192 \times 3$ image and outputs an image of the same dimensions. The input image is downsampled in each encoding block such that it is reduced to a $1 \times 1 \times C$ feature map at the bottleneck. Each encoder block consists of instance normalization, LeakyReLU activation, and stride-2 convolutions (downsampling), while each decoder block consists of instance normalization, ReLU activation, and stride-$1/2$ convolutions (upsampling).

4.3.2 Learning Canonical Scene Appearance

We now turn to the problem of finding an appropriate function $\mathbf{f}(\cdot)$ that minimizes Equation (4.7) for a fixed $\mathbf{T}_{t,r}$. While there are many approaches to finding $\mathbf{f}(\cdot)$, an appealing choice for handling the complexity of visual data is to train an encoder-decoder model (Hinton and Salakhutdinov, 2006) based on deep convolutional neural networks (CNNs). A typical architecture for such a model consists of a compression stage that convolves an input image with a battery of learned filters and downsamples it to a minimal representation, followed by a decompression stage that upsamples and manipulates this representation to construct a new image of the original size.

Following recent work on image-to-image translation (Isola et al., 2017), we adopt a variant of the encoder-decoder network called a *U-Net* (Ronneberger et al., 2015), which is augmented with skip connections between corresponding blocks in the compression and decompression stages. This allows the network to preserve information that may otherwise have been lost during the compression stage.

Figure 4.2 shows our model architecture, consisting of seven encoder blocks (green) and seven decoder blocks (red), each sharing information with their counterparts. Each encoder block consists of instance normalization, LeakyReLU activation, and stride-2 convolutions (downsampling), while each decoder block consists of instance normalization, ReLU activation, and stride-$1/2$ convolutions (upsampling). The outermost blocks are exceptions: the first encoder block consists of only strided convolutions, while the final decoder block constrains the range of the output image to $[0, 1]$ by applying a scaled and translated $\tanh(\cdot)$ function. We include dropout layers after the three innermost encoder and decoder blocks to reduce overfitting. For the purposes of defining our training loss function, we refer to the parameters of the U-Net as $\boldsymbol{\theta}$.

We train our model on pairs of corresponding images captured at identical poses under different illumination conditions, and apply this transformation to the incoming image stream directly. Practically, at present, this limits us to training on synthetic datasets since, to our knowledge, no real-world dataset exists that provides calibrated stereo or RGB-D images captured under variable illumination at identical poses over long trajectories. However, there is mounting evidence that models trained in simulation

Table 4.1: Summary of experiments in Chapter 4.

Section	Description	Dataset(s)
Section 4.4.2	Description of CAT training procedure.	ETHL, Virtual KITTI, Affine KITTI
Section 4.4.3	Description of CAT testing procedure.	ETHL, Virtual KITTI, Affine KITTI
Section 4.4.4	Comparison of direct VO performance using original images, analytical image transformations, and CAT.	ETHL, Virtual KITTI, Affine KITTI
Section 4.4.5	Comparison of direct visual localization performance against a keyframe map using original images, analytical image transformations, and CAT.	ETHL, Virtual KITTI, Affine KITTI

can nevertheless be useful for real-world vision tasks (Gaidon et al., 2016; Peris et al., 2012; Skinner et al., 2016; Tobin et al., 2017). Accordingly, we provide a preliminary investigation of synthetic-to-real transfer learning using data from real-world environments similar to the synthetic training environments.

4.4 Experiments

We conducted several visual localization experiments to validate the use of the canonical appearance transformation (CAT) models described in Section 4.3.2 as a component in the direct localization pipeline described in Section 4.3.1. Specifically, we examined the accuracy and success rates of both visual odometry under time-varying illumination and 6-dof metric localization against a keyframe map under subsequent illumination change. Table 4.1 summarizes the experiments presented in this chapter.

In each experiment we compared the performance of the localization pipeline using the original images, images transformed using illumination-resistant analytical mappings, and CAT-transformed images. We selected the gradient magnitude (GradM) and census transform (Census) as our two illumination-resistant analytical transformations, as they were reported by Park et al. (2017) to achieve the best performance on direct visual odometry under time-varying illumination.

We compute "GradM" images from grayscale images according to

$$(\mathbf{f} \circ \mathbf{I})(\mathbf{u}) = \|\nabla_{\mathbf{u}} \mathbf{I}(\mathbf{u})\|_2 \tag{4.14}$$

where the directional image derivatives are computed using the 3×3 Sobel-Feldman operators in Equation (2.112). Gradient magnitude images are invariant to local changes in intensity bias.

We compute "Census" images from grayscale images by applying the census transform (Zabih and Woodfill, 1994) in a 3×3 window about each pixel. For a given pixel $\mathbf{u}$, we evaluate the following function at each of its eight neighbouring pixels $\mathbf{u}'$:

$$\xi(\mathbf{I}(\mathbf{u}), \mathbf{I}(\mathbf{u}')) = \begin{cases} 0, & \mathbf{I}(\mathbf{u}) > \mathbf{I}(\mathbf{u}') \\ 1, & \mathbf{I}(\mathbf{u}) \leq \mathbf{I}(\mathbf{u}') \end{cases} \tag{4.15}$$

The results of these comparisons[1] are then concatenated to obtain an 8-bit value, which can be interpreted

[1]The specific choice of comparison operator is irrelevant as long as it is consistent.

Figure 4.3: *Top row:* Sample images from the `ETHL/syn1` sequences captured at the same pose under different illumination conditions. *Bottom row:* The same images with a learned transformation that attempts to re-illuminate each image under the "Static" condition.

as a decimal integer to obtain a new intensity value in $[0, 255]$ at $\mathbf{u}$. For example, the census transform of a 3×3 window where $\mathbf{I}(\mathbf{u}) = 64$ might be

$$\begin{bmatrix} 124 & 74 & 32 \\ 124 & 64 & 18 \\ 157 & 116 & 84 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 0 \\ 1 & & 0 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow 110\ 10\ 111 \rightarrow 215. \tag{4.16}$$

The census transform is invariant to changes that preserve intensity ordering, and is popular in the context of optical flow estimation and stereo matching (Hafner et al., 2013).

4.4.1 Datasets

We explored our method using two synthetic RGB-D datasets created using high-fidelity rendering techniques, as well as corresponding real-world data from similar environments.

ETHL Dataset The `ETHL` dataset (Park et al., 2017) consists of three sets of camera trajectories and corresponding RGB-D imagery (640×480 resolution) captured under various illumination conditions. Two of these sets, dubbed `ETHL/syn1` and `ETHL/syn2`, consist of images captured along two different trajectories under five different illumination conditions, all in a simulated environment based on the ICL-NUIM dataset (Handa et al., 2014). Figure 4.3 (top row) shows sample images from the `ETHL/syn1` sequences. The "Local", "Global", and "Local/Global" conditions consist of time-varying illumination, while the "Flashlight" condition consists of view-dependent illumination generated by a light source attached to the camera. The third set, dubbed `ETHL/real`, consists of three different camera trajectories with VICON ground truth captured in a cluttered desk scene under illumination conditions analogous to those found in the `ETHL/syn` sequences. Figure 4.4 shows sample images from the three `ETHL/real` sequences. We note that the appearance and illumination of the `ETHL/real` scene differ significantly from those found in the synthetic scenes.

Virtual KITTI Dataset The Virtual KITTI (`VKITTI`) dataset (Gaidon et al., 2016) is a partial virtual reconstruction of the KITTI vision benchmark (Geiger et al., 2013), consisting of five sets of camera trajectories with RGB-D imagery (1242×375 resolution) rendered under a variety of simulated

Figure 4.4: *Top row:* Sample images from the `ETHL/real` sequences captured at different poses under different illumination conditions. *Bottom row:* The same images with a learned transformation that attempts to re-illuminate each image under the "Static" condition of the `ETHL/syn` environment.

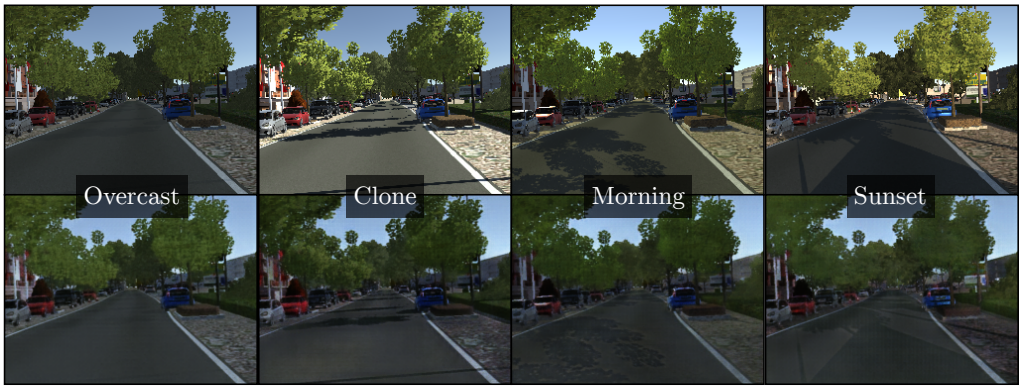


Figure 4.5: *Top row:* Sample images from the `VKITTI/0001` sequences captured at the same pose under different illumination conditions. *Bottom row:* The same images with a learned transformation that attempts to re-illuminate each image under the "Overcast" condition.

illumination conditions, including "Morning", "Sunset", "Overcast" and "Clone" (Figure 4.5). The latter is meant to replicate the conditions found in the original data. This dataset has previously been used to demonstrate transfer learning between real and synthetic environments for visual object tracking (Gaidon et al., 2016).

Affine-KITTI Dataset The KITTI vision benchmark itself does not contain trajectories with significant overlap, nor does it exhibit variation in environmental illumination since each sequence was recorded around the same time of day. We nonetheless investigated synthetic-to-real transfer learning by creating three copies of the 2.2 km KITTI odometry benchmark sequence `KITTI/05`, modified by a per-pixel affine transformation:

$$\mathbf{I}(\mathbf{u}) \leftarrow a\mathbf{I}(\mathbf{u}) + b. \tag{4.17}$$

While an affine transformation has no impact on directional effects such as shadows or reflectance, it is analogous to a global change in illumination intensity similar to the "Global" condition of the `ETHL` sequences. We refer to these conditions as "Clone" ($a = 1, b = 0$), "Light" ($a = 1.5, b = 0.1$) and "Dark"

Figure 4.6: *Top row:* Sample images from the `KITTI/05` sequence (Geiger et al., 2013) from the same pose under different affine brightness transformations. *Bottom row:* The same images with a learned transformation that attempts to re-illuminate each image under the `VKITTI` "Overcast" condition.

($a = 0.8, b = -0.2$), and to the dataset as a whole as Affine-KITTI (`AKITTI`).[2] Figure 4.6 illustrates the resulting images.

4.4.2 Training

We generated training image pairs consisting of a target image captured under a chosen canonical illumination condition and a corresponding input image captured under different conditions at the same pose. We chose the "Static" illumination condition as the canonical appearance of the `ETHL` sequences, and the "Overcast" condition as the canonical appearance of the `VKITTI` sequences, as they exhibit the most stable illumination and the fewest directional lighting effects (e.g., shadows). During training, we resized and center-cropped each image to 320×240 resolution, then applied a random crop to obtain image pairs of the desired 256×192 resolution. This random cropping step provides an easy way to augment the dataset and reduce overfitting by ensuring that different data are used in each training epoch.

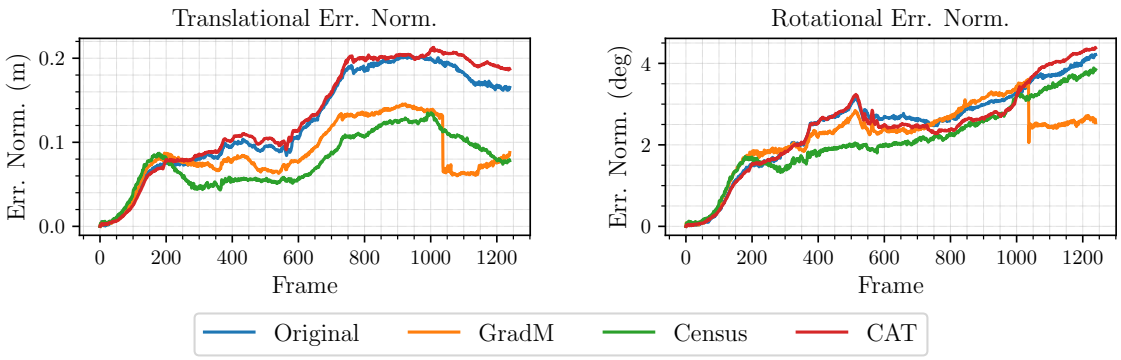
While Isola et al. (2017) combine an L_1 loss with an adversarial loss (Goodfellow et al., 2014) to learn a mapping between input and output images that maximizes a subjective measure of realism, in our case we are interested in learning a mapping that explicitly maximizes the photometric consistency of the output and target images in an L_2 sense (see Equation (4.9)). We therefore trained the U-Net directly using the squared L_2 loss:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{NWHC} \sum_{i=1}^{N} \sum_{\mathbf{u}} \left\| \mathbf{I}_r^i(\mathbf{u}) - (\mathbf{f} \circ \mathbf{I}_t^i)(\mathbf{u}) \right\|_2^2, \tag{4.18}$$

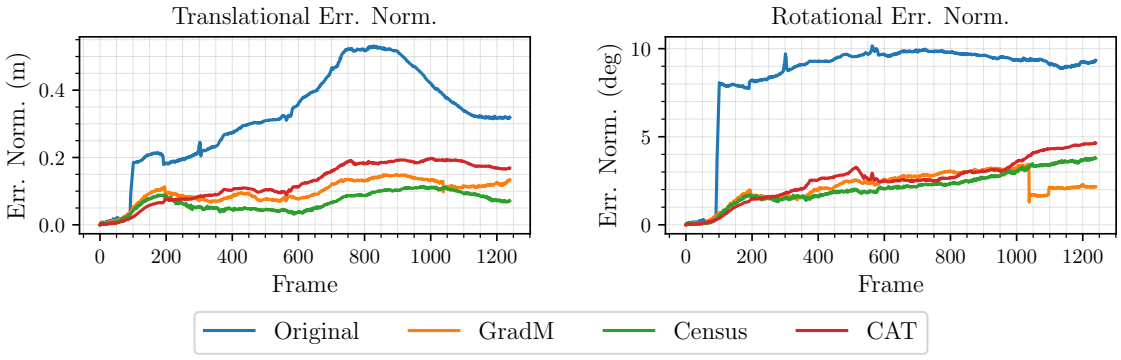
where $N = 64$ is our chosen batch size, W, H, and C are the width, height, and channels of $\mathbf{I}_r^i$, respectively ($256 \times 192 \times 3$), and $\boldsymbol{\theta}$ are the parameters of the CNN defining $\mathbf{f}(\cdot)$. We trained each of our models from scratch for 100 epochs, using the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-4} and other parameters identical to those used by Isola et al. (2017).

In each experiment, we trained a model using all available imagery from the other trajectories in the dataset, and tested it on the remaining trajectory (e.g., the model used for the four `VKITTI/0001`

[2] Note that the intensities have been rescaled to lie in the range [0,1] prior to applying these transformations. We clip out-of-range intensities to [0,1].



(a) "Static" (canonical)



(b) "Local"

Figure 4.7: Comparison of VO errors for the `ETHL/syn2` trajectory (continued on next page).

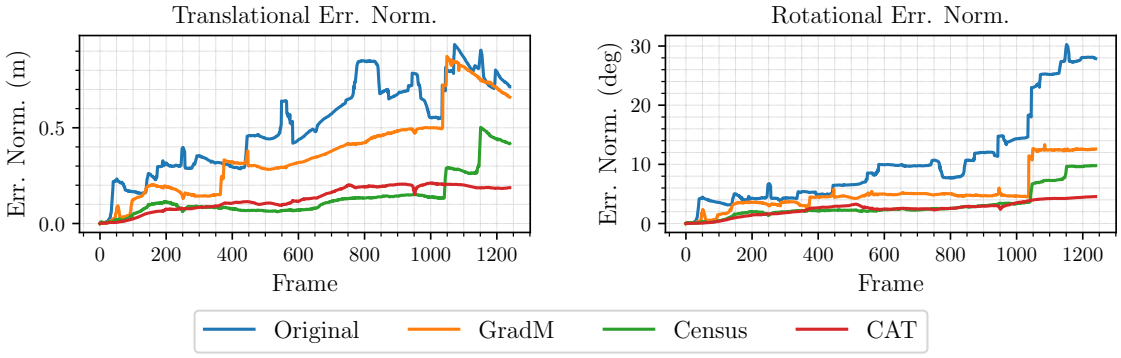
sequences was trained on the remaining sixteen `VKITTI` sequences listed in Table 4.5). Each training set consisted of approximately 4000–8000 image pairs.

4.4.3 Testing

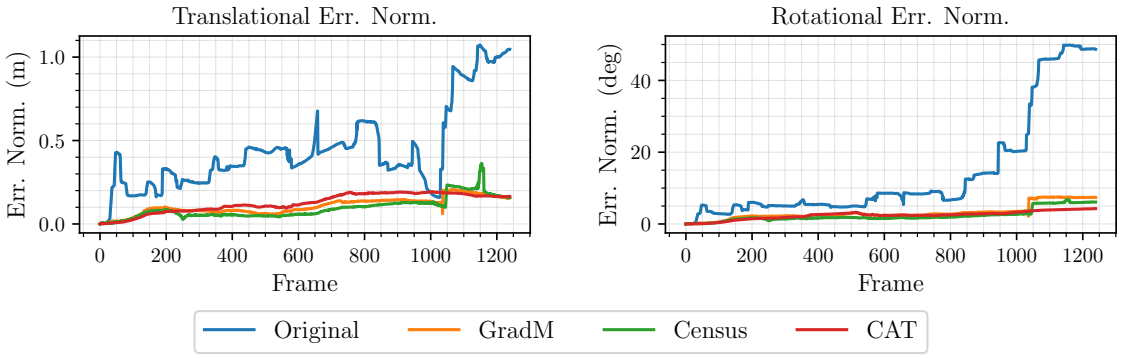
During testing, we did not use random cropping, but rather resized and center-cropped each input image directly to 256×192 resolution. Figures 4.3 and 4.5 show sample inputs (top rows) and outputs (bottom rows) of our trained models for each illumination condition of the `ETHL/syn1` and `VKITTI/0001` sequences. The models captured the low-frequency components of illumination change well, but struggled to render high-frequency texture or completely deal with other high-frequency effects such as the edges of strong shadows. As a result, the network outputs were often slightly blurred compared to the original images, and sometimes contained artifacts in regions of significant local appearance change. However, the impact of such artifacts was partially mitigated by our use of a robust loss function in our localization algorithm.

4.4.4 Direct Visual Odometry Under Illumination Change

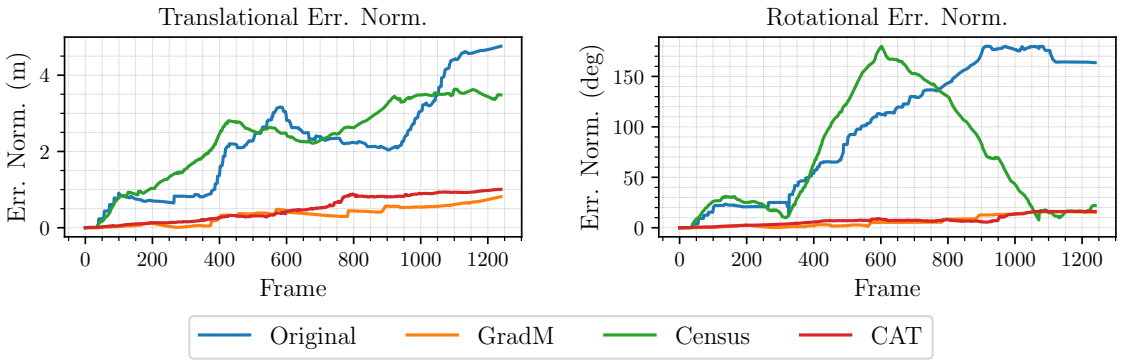
To assess the usefulness of our trained models for doing VO through time-varying illumination, we compared the performance of our direct VO pipeline on each `ETHL` sequence individually, using the original images (resized and cropped to 256×192 resolution), GradM and Census images, and CAT



(c) "Local/Global"



(d) "Global"



(e) "Flashlight"

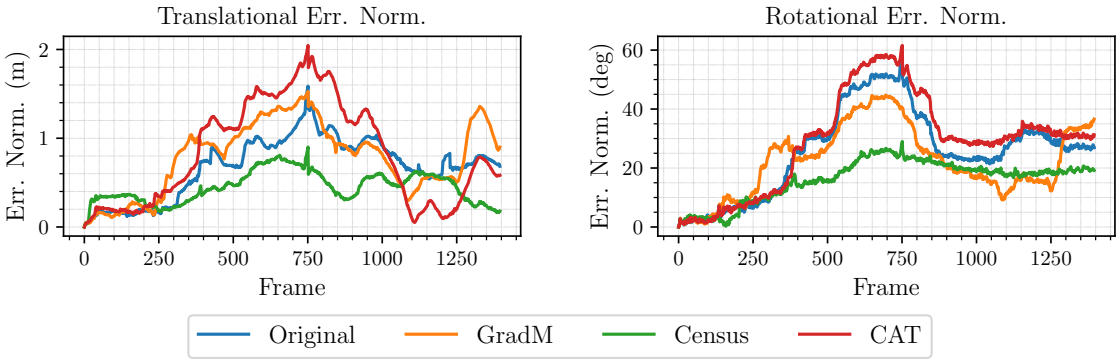
Figure 4.7: Comparison of VO errors for the `ETHL/syn2` trajectory (continued from previous page).

images. We omit the results of our VO experiments on the VKITTI dataset since, unlike the ETHL dataset, the illumination conditions remain static within each sequence and thus we observed little difference in VO performance.

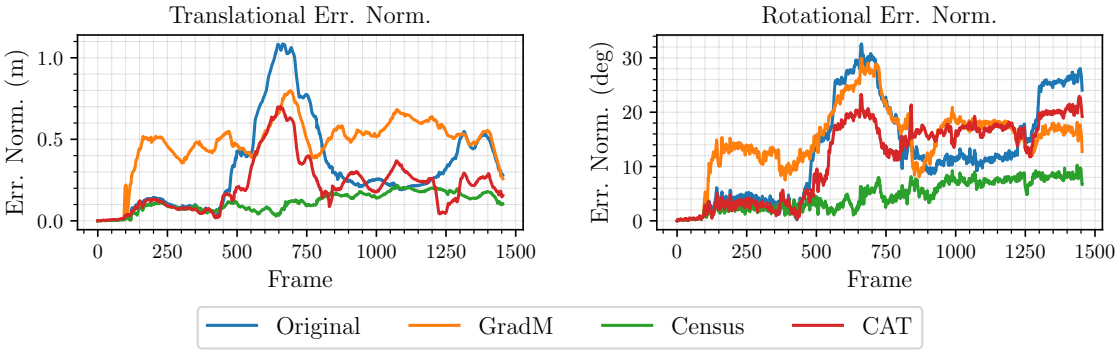
Figure 4.7 shows sample VO errors for each illumination condition of the ETHL/syn2 trajectory using both the original image streams and the outputs of each image transformation, while Table 4.2 summarizes our VO results for all ETHL sequences. Although our VO pipeline successfully tracked the entirety of the ETHL sequences despite rapidly time-varying illumination, the average translational and rotational errors were consistently lower using any of the transformed images, due to the improved temporal consistency of the transformed image stream. We note that for this VO task, the Census images frequently yielded the lowest estimation errors in both translation and rotation for every condition in the ETHL/syn environment except the "Flashlight" condition, where it produced very large estimation errors. In contrast, the GradM and CAT images perform much better on the "Flashlight" condition, with the GradM images yielding slightly lower estimation errors than the CAT images. However, we note that the CAT images generally yielded lower estimation errors than the GradM images in the other conditions, achieving accuracy on par with the canonical "Static" condition in most cases. This suggests that the trained CAT models produce consistent outputs and exhibit robustness to a broader range of illumination conditions than the GradM and Census transformations, but do not improve VO accuracy beyond the level that would achieved in the canonical condition itself.

We also investigated the possibility of transferring a CAT model trained in the synthetic ETHL/syn environment to the real-world ETHL/real environment. Figure 4.8 shows VO errors for the three ETHL/real sequences using the original images, GradM and Census images, and images from a CAT model trained only on ETHL/syn sequences. As shown in Table 4.2, the transferred CAT model neither significantly improved nor significantly worsened VO accuracy on these sequences. We believe this is because the ETHL/real sequences differ too much from the ETHL/syn sequences in terms of appearance and illumination for the learned model to be useful (see Figure 4.4). However, defaulting to a transformation near identity is a desirable property in such cases, as we should always be able to fall back on the original image stream. On the other hand, the census transform yielded significant improvements in VO accuracy on the ETHL/real sequences, consistent with the findings of Park et al. (2017).

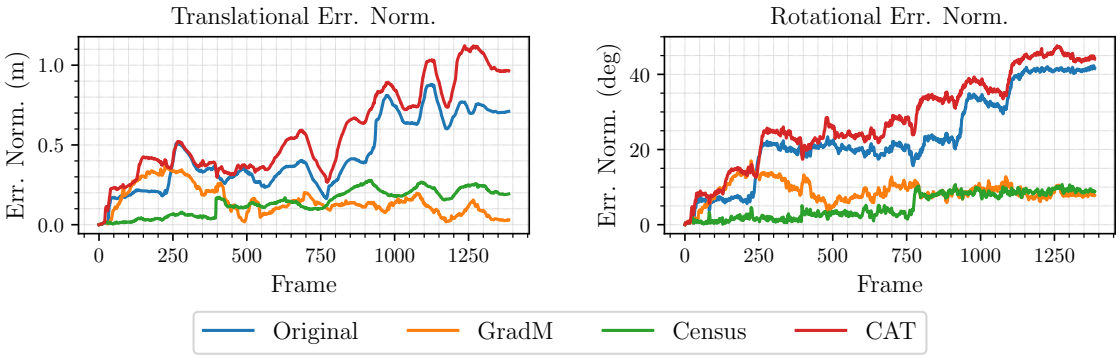
Together these results suggest that, while the census transform is robust to a relatively narrow range of time-varying illumination conditions, it is broadly applicable and can improve the accuracy of direct VO even in nominal conditions with static illumination. It may be the case that a marriage of the Census and CAT methods, where the model is trained to regress a canonical Census image rather than a canonical RGB image, would combine the strengths of both approaches, resulting in an image transformation that exhibits superior generalizability outside of the training environment while simultaneously widening the range of illumination conditions to which the model is robust. We leave this investigation to future work.



(a) "Global"



(b) "Local"



(c) "Flashlight"

Figure 4.8: Comparison of VO errors for the `ETHL/real` trajectories. The CAT model has been trained using only data from the `ETHL/syn` environment.

Table 4.2: Comparison of direct visual odometry (VO) results under rapidly time-varying illumination in the `ETHL` sequences. The best results are highlighted in bold.

	Frames Tracked (%)				**Avg. Trans. Err. (% Dist.)**				**Avg. Rot. Err. ($\times 10^{-2}$ deg/m)**			
Sequence (length)	Orig.	GradM	Census	CAT	Orig.	GradM	Census	CAT	Orig.	GradM	Census	CAT
`ETHL/syn1`												
(880 frames, 9.0 m)												
Static (canonical)	100.00	100.00	100.00	100.00	1.47	1.54	**0.96**	1.57	44.65	58.24	**25.39**	45.07
Local	100.00	100.00	100.00	100.00	3.16	1.95	**0.92**	1.72	114.99	77.81	**19.72**	45.08
Global	100.00	100.00	100.00	100.00	4.76	1.87	**1.18**	1.59	219.06	62.03	**37.56**	49.20
Local + Global	100.00	100.00	100.00	100.00	4.10	2.45	2.29	**1.78**	125.14	88.26	63.53	**61.74**
Flashlight	100.00	100.00	100.00	100.00	35.36	**6.59**	30.77	7.92	1095.28	287.02	935.49	**176.36**
`ETHL/syn2`												
(1240 frames, 7.8 m)												
Static (canonical)	100.00	100.00	100.00	100.00	1.61	1.10	**0.99**	1.70	31.66	28.56	**27.04**	31.07
Local	100.00	100.00	100.00	100.00	4.13	1.25	**0.87**	1.61	109.22	27.29	**27.06**	32.73
Global	100.00	100.00	100.00	100.00	5.64	1.38	**1.20**	1.60	181.08	40.69	**29.75**	31.28
Local + Global	100.00	100.00	100.00	100.00	6.53	4.76	1.73	**1.68**	134.47	68.14	39.96	**32.22**
Flashlight	100.00	100.00	100.00	100.00	27.96	**4.32**	30.00	6.37	1294.11	**82.75**	890.02	94.74
`ETHL/real`[1]												
Local	100.00	100.00	100.00	100.00	2.35	3.45	**0.83**	1.55	93.28	109.15	**33.68**	84.86
(1455 frames, 13.9 m)												
Global	100.00	100.00	100.00	100.00	3.09	3.56	**2.03**	3.73	117.72	101.31	**73.34**	131.79
(1396 frames, 22.1 m)												
Flashlight	100.00	100.00	100.00	100.00	3.59	1.22	**1.15**	4.76	195.28	72.94	**41.63**	234.79
(1387 frames, 12.1 m)												

[1] Model trained on all `ETHL/syn1` and `ETHL/syn2` sequences. Data from `ETHL/real` sequences were not used for training the models used on the `ETHL/syn` sequences.

4.4.5 Direct Visual Localization Across Illumination Conditions

In addition to VO, a common task for autonomous vehicles is 6-dof metric localization against an existing map, and an important competency for long-term autonomy is the ability to do so at different times of day when the illumination of the environment may have changed. To this end, we investigated the use of our trained CAT models for teach-and-repeat-style metric localization (Paton et al., 2017; Clement et al., 2017a; Gridseth and Barfoot, 2019) by first creating a map in the canonical condition, then localizing against it in different conditions using both the original and transformed images.

Figure 4.9 shows localization errors for the "Overcast", "Clone", "Morning", and "Sunset" conditions of the `VKITTI/0018` trajectory using the original images (resized and cropped to 256×192 resolution), GradM and Census images, and CAT images. Tables 4.3 and 4.5 summarize our localization results for all `ETHL/syn`, and `VKITTI` sequences, both in terms of localization success rates (percentage of the trajectory traversed before estimator divergence), and estimation error on the successfully traversed portions. We observed significant improvements in terms of both localization success and localization accuracy using the trained CAT models, with many otherwise untraversable sequences becoming fully or mostly traversable. In most cases, our method achieved localization accuracy on par with that achieved in the VO experiments, indicating that the outputs of our models are generally consistent for different inputs.

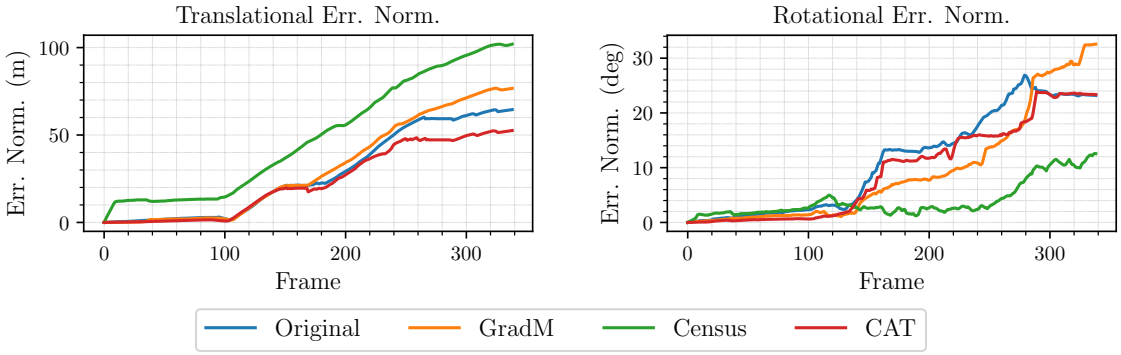
Compared to the CAT images, the GradM images yielded relatively poor localization performance in both the `ETHL` and `VKITTI` environments, suggesting that image gradients are not robust when the illumination conditions vary substantially. This is to be expected since important sources of gradient information, such as the edges of shadows, vary dramatically as a function of illumination. The Census images yield better localization performance than the GradM images, but again were not robust to the "Flashlight" condition of the `ETHL/syn` environment, and performed poorly on most `VKITTI` sequences, again likely due to changing shadows and lighting patterns affecting the ordering of pixel intensities. Consistent with the VO results reported in Table 4.2, the Census images generally resulted in superior localization accuracy on sequences where the estimator did not diverge. This further indicates that training CAT models on Census images rather than RGB images may be a fruitful direction for future research.

The outlier in these experiments was the `VKITTI/0020` sequence, on which no method yielded significant improvements in localization success rates. This may have been due to the fact that much of the camera's field of view in this sequence is occupied by moving vehicles, which makes localization challenging and introduces a significant amount of view-dependent high-frequency reflectance that was not well captured by our model.

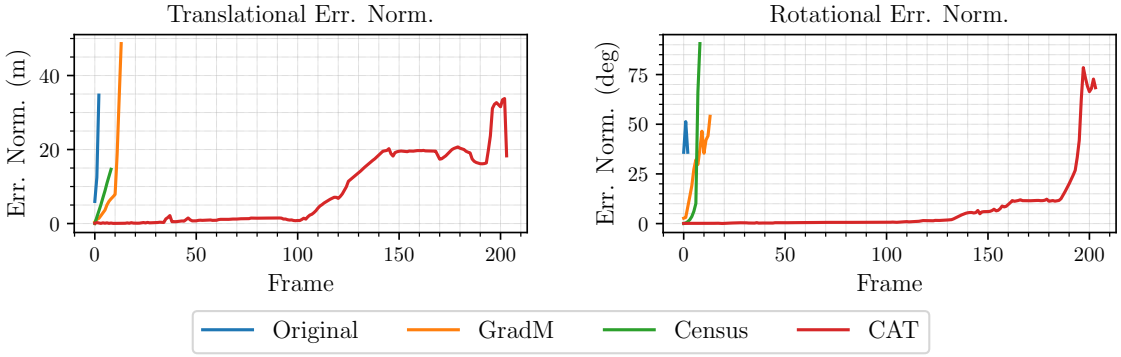
We further investigated the use of a CAT model trained on all `VKITTI` sequences for improving localization on the `AKITTI` sequences described in Section 4.4.1. To do this, we modified Equations (4.1) and (4.2) to use stereo disparity rather than depth, and used a standard block matching algorithm to compute the disparity map. We used the "Clone" condition of the `AKITTI/05` sequence to create the keyframe map for these experiments. As shown in Figure 4.10 and Table 4.5, the `VKITTI`-trained model yielded only modest gains in localization success and accuracy compared to the untransformed imagery on sequence `AKITTI/05`. Returning to Figure 4.6, we see that the CAT model outputs are not entirely consistent and contain visible artifacts, which is an indication that the `VKITTI` training data are not sufficiently representative of the real KITTI data to allow for sim-to-real transfer learning using this approach. Further investigation is needed to establish the limits of synthetic-to-real transfer learning in

this context, which we leave to future work.

The GradM images significantly improved localization performance for all `AKIITI` conditions, while the Census images did not. This difference is likely due to the fact that parts of the `AKITTI` "Light" and "Dark" images became fully saturated or fully desaturated (cf. Figure 4.6), which impacted the ordering of intensities more than the presence of strong gradients at edges and corners.

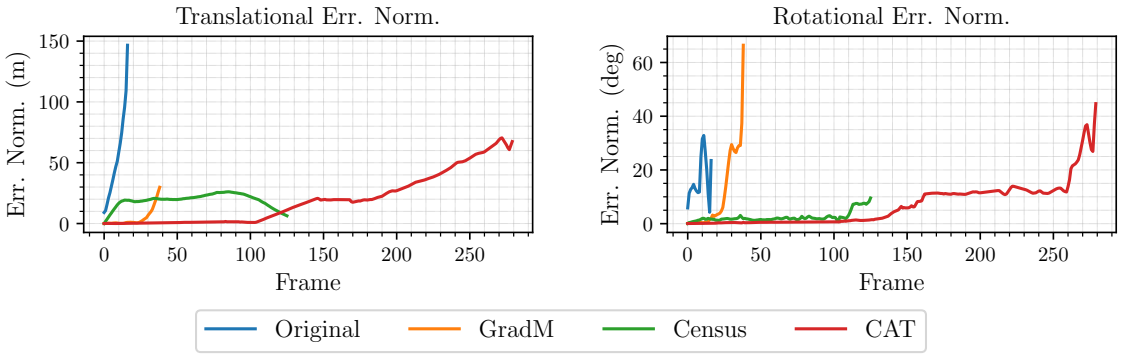


(a) "Overcast" (canonical)

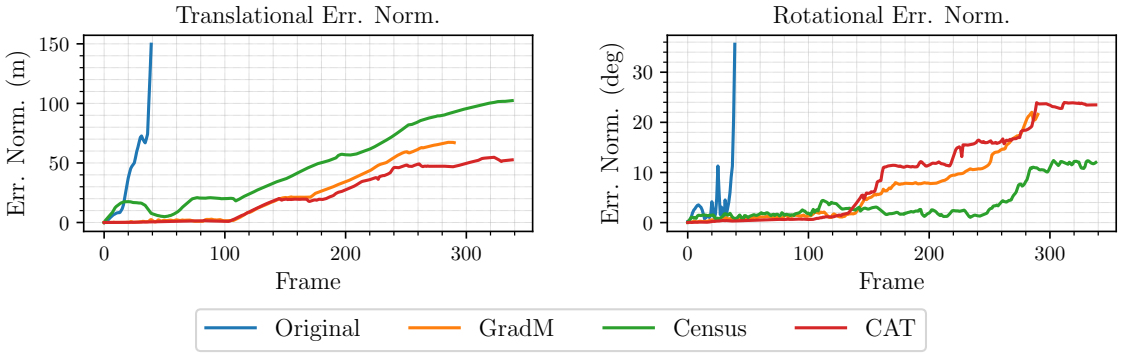


(b) "Clone"

Figure 4.9: Comparison of metric localization errors on the `VKITTI/0018` trajectory (continued on next page).

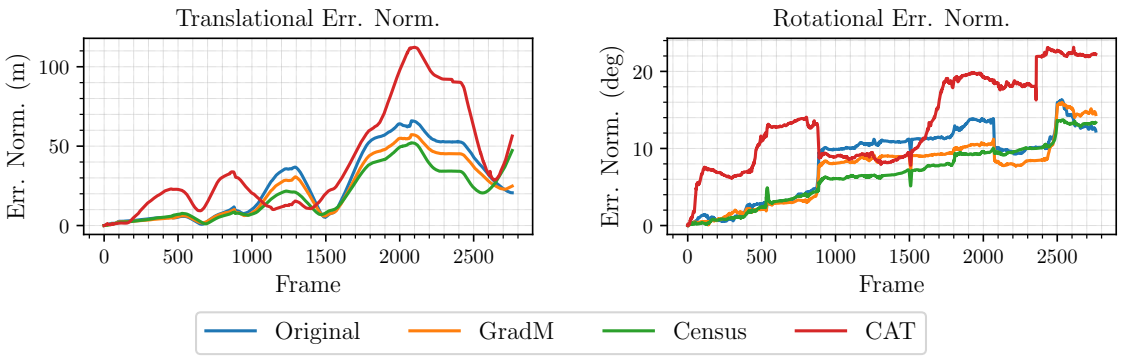


(c) "Morning"

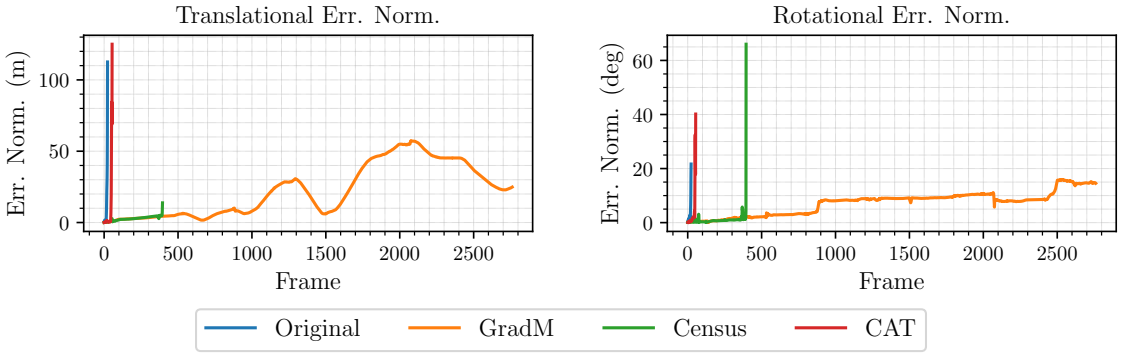


(d) "Sunset"

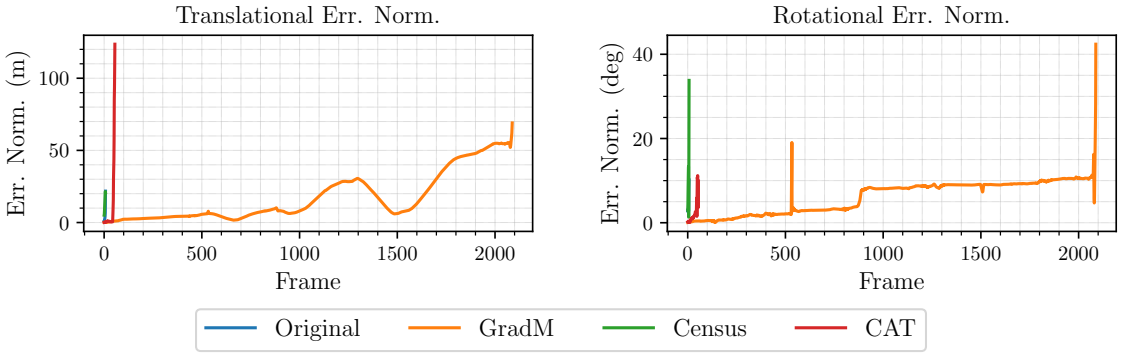
Figure 4.9: Comparison of metric localization errors on the `VKITTI/0018` trajectory (continued from previous page).



(a) "Clone" (map)



(b) "Light"



(c) "Dark"

Figure 4.10: Comparison of metric localization errors for the `AKITTI/05` sequences. The CAT model was trained using only data from the `VKITTI` environment.

Table 4.3: Comparison of direct visual localization performance on `ETHL` sequences against a keyframe map created in the canonical condition. The best results are highlighted in bold.

	Frames Tracked (%)				**Avg. Trans. Err. (% Dist.)**				**Avg. Rot. Err. ($\times 10^{-2}$ deg/m)**			
Sequence (length)	Orig.	GradM	Census	CAT	Orig.	GradM	Census	CAT	Orig.	GradM	Census	CAT
`ETHL/syn1` (880 frames, 9.0 m)												
Static (canonical)	**100.00**	**100.00**	**100.00**	**100.00**	1.47	1.54	**0.96**	1.57	44.65	58.24	**25.39**	45.07
Local	98.98	40.91	**100.00**	**100.00**	1.67	3.85	**0.98**	1.57	48.24	78.74	**26.08**	45.14
Global	5.57	**100.00**	40.11	**100.00**	25.05	**1.54**	2.20	1.56	474.29	58.22	**38.82**	45.06
Local + Global	7.39	19.77	40.34	**100.00**	56.80	10.90	2.33	**1.57**	640.85	180.71	**41.75**	45.37
Flashlight	1.59	21.82	1.36	**31.02**	>1000	12.89	>1000	**6.18**	>1000	330.17	>1000	**102.06**
`ETHL/syn2` (1240 frames, 7.8 m)												
Static (canonical)	**100.00**	**100.00**	**100.00**	**100.00**	1.61	1.10	**0.99**	1.70	31.66	28.56	**27.04**	31.07
Local	**100.00**	**100.00**	**100.00**	**100.00**	1.56	1.09	**1.00**	1.70	31.04	28.68	**27.42**	31.08
Global	4.76	**100.00**	**100.00**	**100.00**	123.82	1.10	**0.99**	1.70	>1000	28.70	**27.02**	31.10
Local + Global	5.08	54.27	**100.00**	**100.00**	125.41	1.83	**1.00**	1.70	>1000	52.30	**27.17**	31.05
Flashlight	1.85	16.77	1.37	**40.00**	682.36	10.18	>1000	**3.32**	>1000	162.50	>1000	**69.19**

Table 4.4: Comparison of direct visual localization performance on `AKITTI/05` sequences against a keyframe map created in the "Clone" condition. The best results are highlighted in bold.

	Frames Tracked (%)				**Avg. Trans. Err. (% Dist.)**				**Avg. Rot. Err. ($\times 10^{-2}$ deg/m)**			
Sequence (length)	Orig.	GradM	Census	CAT	Orig.	GradM	Census	CAT	Orig.	GradM	Census	CAT
`AKITTI/05`[1] (2762 frames, 2206 m)												
Clone (map)[2]	**100.00**	**100.00**	**100.00**	**100.00**	1.19	1.02	**0.89**	1.78	0.38	0.33	**0.30**	0.61
Light	0.87	**100.00**	14.34	1.99	96.98	1.02	**0.95**	21.18	24.46	0.33	**0.32**	8.68
Dark	0.29	**75.63**	0.29	2.03	130.83	**1.06**	335.87	24.43	114.15	**0.38**	278.19	4.63

[1] Model trained on all `VKITTI` sequences. Data from the `AKITTI/05` sequences were not used for training the models used on the `VKITTI` sequences
[2] The "Clone" condition was used to create the initial keyframe map for localization experiments with the `AKITTI/05` sequences.

Table 4.5: Comparison of direct visual localization performance on VKITTI sequences against a keyframe map created in the canonical condition. The best results are highlighted in bold.

	Frames Tracked (%)				**Avg. Trans. Err. (% Dist.)**				**Avg. Rot. Err. ($\times 10^{-2}$ deg/m)**			
Sequence (length)	Orig.	GradM	Census	CAT	Orig.	GradM	Census	CAT	Orig.	GradM	Census	CAT
VKITTI/0001 (447 frames, 332.5 m)												
Overcast (canonical)	**100.00**	**100.00**	**100.00**	**100.00**	0.98	1.07	**0.47**	0.96	0.72	0.74	0.51	**0.40**
Clone	39.15	64.21	**100.00**	**100.00**	0.78	1.76	**0.52**	0.96	0.90	1.61	0.53	**0.40**
Morning	40.49	51.90	**100.00**	56.15	0.72	1.33	**0.66**	1.88	0.86	1.21	**0.54**	3.03
Sunset	5.37	**51.01**	37.14	41.39	66.15	1.60	**0.35**	0.77	41.80	1.77	1.30	**0.74**
VKITTI/0002 (233 frames, 113.6 m)												
Overcast (canonical)	**100.00**	**100.00**	**100.00**	**100.00**	1.42	1.29	1.71	**0.54**	1.46	1.15	1.01	**0.81**
Clone	11.16	10.73	33.05	**100.00**	93.11	84.15	3.64	**0.54**	25.58	53.98	2.17	**0.77**
Morning	40.77	71.67	23.18	**100.00**	2.82	1.18	22.54	**0.53**	2.21	1.07	12.73	**0.89**
Sunset	63.52	**89.70**	56.65	86.70	1.31	1.28	1.99	**0.54**	1.29	1.11	0.96	**0.88**
VKITTI/0006 (270 frames, 51.9 m)												
Overcast (canonical)	**100.00**	**100.00**	**100.00**	**100.00**	0.41	0.27	**0.23**	0.33	0.27	0.25	**0.21**	0.27
Clone	1.11	34.44	49.26	**93.70**	>1000	>1000	>1000	**0.96**	>1000	>1000	>1000	**0.86**
Morning	84.07	74.81	48.15	**92.22**	**2.23**	27.47	>1000	10.21	6.22	167.00	>1000	**8.78**
Sunset	**88.89**	77.78	78.89	88.52	**1.49**	21.67	4.07	2.16	2.38	17.53	**2.32**	2.80
VKITTI/0018 (339 frames, 254.4 m)												
Overcast (canonical)	**100.00**	**100.00**	**100.00**	**100.00**	10.82	12.05	19.27	**9.16**	4.39	3.66	**1.57**	3.73
Clone	0.88	4.13	2.65	**60.18**	548.40	50.45	56.27	**4.24**	>1000	137.05	156.99	**3.24**
Morning	5.01	11.50	37.17	**82.60**	215.31	**7.69**	11.83	8.07	65.46	18.96	**1.53**	2.89
Sunset	11.80	85.84	**100.00**	**100.00**	65.16	9.57	19.31	**9.23**	7.08	2.40	**1.51**	3.73
VKITTI/0020 (837 frames, 711.2 m)												
Overcast (canonical)	**100.00**	**100.00**	**100.00**	**100.00**	8.05	**7.71**	12.99	9.00	1.46	1.00	1.44	**0.84**
Clone	5.38	5.50	8.36	**8.48**	49.85	137.00	41.72	**38.15**	39.55	106.68	34.16	**25.53**
Morning	**14.81**	7.65	8.12	**14.81**	39.67	**28.86**	28.91	50.47	**6.91**	11.84	16.67	7.45
Sunset	4.66	10.16	**15.17**	13.50	197.80	**29.43**	32.98	40.19	71.93	6.92	14.45	**6.79**

4.5 Conclusions and Future Work

This chapter presented a method for improving the robustness of direct visual localization to environmental illumination change by training a deep convolutional encoder-decoder network to re-illuminate images under a previously-seen canonical appearance condition. We validated the use of such canonical appearance transformations (CATs) for both visual odometry (VO) and topometric keyframe-based localization tasks using two high-fidelity synthetic RGB-D datasets, and demonstrated significant gains in terms of both localization accuracy and localization success rates under conditions of severe appearance change where conventional methods fail. We found that our method compares favourably to illumination-robust analytical image transformations, such as the gradient and census transforms, in terms of its ability to improve VO and localization performance over a wide variety of illumination conditions.

Avenues for future work include investigating how the requirement for identically posed training examples can be relaxed so that training can be easily accomplished outside of simulation, as well as further exploring the use of simulation-trained models in real-world environments, and investigating how networks such as ours can be integrated more deeply into visual localization systems. Evaluating alternative representations of the canonical appearance condition may also prove fruitful. In particular, training an image transformation to regress a canonical census-transformed image rather than a canonical RGB image may provide a means of combining the strengths of both approaches, yielding improved localization accuracy and generalization performance between simulated and real environments, while maintaining a flexible model capable of operating across a broad spectrum of appearance conditions.

In Chapter 5 we propose a technique for recasting the image-to-image translation problem to allow for self-supervised learning of a canonical appearance optimized for a particular environment, imaging sensor, and localization pipeline. While we employ a more restricted class of image transformations than in Chapter 4, this approach avoids the requirement for identically posed training examples, and does not require the user to manually select a canonical appearance condition, as the system automatically learns an appropriate representation geared towards optimizing localization performance.

4.6 Novel Contributions

Our main contributions can be summarized as follows:

1. We trained a convolutional encoder-decoder network to map images of a scene undergoing illumination change onto a previously-encountered canonical appearance, and incorporated the learned transformations as a pre-processing stage in a direct visual localization pipeline;

2. We demonstrated experimentally that our method yields significant improvements in visual odometry (VO) accuracy under time-varying illumination, as well as improved tracking performance in 6-dof metric localization under conditions of severe illumination change, where conventional methods fail;

3. We showed that our learning-based approach compares favourably against commonly used illumination-robust analytical image transformations;

4. We investigated the possibility of transfer learning from synthetic to real environments in a localization context; and

5. We released an open-source implementation of our method using PyTorch (Paszke et al., 2017), available at https://github.com/utiasSTARS/cat-net.

4.7 Associated Publications

- Clement, L. and Kelly, J. (2018). How to train a CAT: Learning canonical appearance transformations for direct visual localization under illumination change. *IEEE Robotics and Automation Letters*, 3(3):2447–2454.

4.8 Associated Video

- Video summary of CAT-Net as presented at the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia:
https://www.youtube.com/watch?v=ej6VNBq3dDE

Chapter 5

Matchable Image Transformations

Long-term self-localization remains challenging for vision-based systems due to appearance changes caused by lighting, weather, or seasonal variations. While experience-based mapping has proven to be an effective technique for bridging the 'appearance gap,' the number of experiences required for reliable localization over days or months can be very large, and methods for reducing the necessary number of experiences are needed for this approach to scale. Taking inspiration from physics-based models of colour constancy, this chapter proposes a method for learning a nonlinear RGB-to-grayscale mapping that maximizes the number of inlier feature matches for images captured under varying lighting and weather conditions, and which can be used as a pre-processing step in a conventional localization pipeline. Our key insight is that appearance-robust image transformations can be learned by training a deep neural network to predict a convenient measure of performance for a given pair of images and a target non-differentiable localization pipeline. We find that cross-appearance feature matching can be further improved by incorporating a learned low-dimensional pairwise context feature. Using synthetic and real-world datasets, we demonstrate substantially improved feature matching across day-night cycles, enabling continuous metric localization over a 30-hour period with a single mapping experience, and allowing experience-based localization to scale to long deployments with dramatically reduced data requirements. We further investigate the effectiveness of our approach for improving localization performance across seasonal appearance change and find that, while our method does improve localization success rates compared to baseline methods, experience-based mapping remains necessary for coping with physical changes to unstructured operating environments over the course of weeks or months.

5.1 Motivation

While the approach of learning Canonical Appearance Transformations in Chapter 4 was effective for improving the robustness of 6-dof metric visual localization under illumination change, it suffers from several limitations that undermine its practicality. First, the requirement for perfectly aligned image pairs effectively limits training to synthetic environments where capturing such data is feasible at the scale required to train deep models. Second, the choice of the canonical appearance condition is arbitrary and user-determined, and there is no guarantee that any given appearance condition is in fact the best choice for accurate, repeatable metric relocalization. Finally, the process of training the image transformation is only loosely connected to the ultimate result of improving localization success under appearance change.

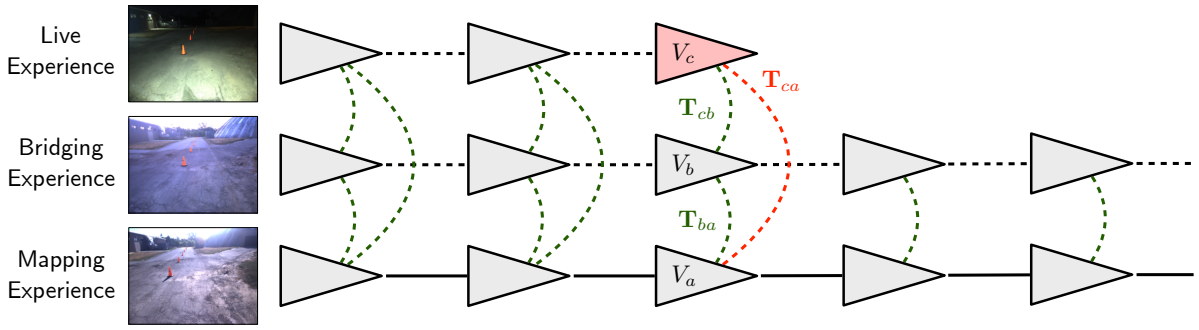


Figure 5.1: Multi-experience localization: A spatio-temporal pose graph records consecutive 'experiences' of an environment as vertices in a graph, where edges encode the 6-dof relative transformation between vertices corresponding to approximately the same physical location. We can localize a vertex V_c in the live (current) experience against a vertex V_a in a privileged mapping experience by first localizing against an intermediate 'bridging' experience, and then compounding the chain of relative transformations between the bridging vertex V_b and the corresponding map vertex: $\mathbf{T}_{ca} = \mathbf{T}_{cb}\mathbf{T}_{ba}$. (Figure adapted from Paton et al. (2016).)

This chapter seeks to address these limitations by reformulating the problem to connect the learned image transformation back to a target localization pipeline and thereby maximize some measure of localization quality. Rather than learning to translate between arbitrary appearance conditions, we propose to learn an image transformation that produces images that are *maximally matchable* for a given feature detector and matcher such as SURF (Bay et al., 2008), ORB (Rublee et al., 2011) or `libviso2` (Geiger et al., 2011). Specifically, we learn a drop-in replacement for the standard RGB-to-grayscale colourspace mapping used to pre-process RGB images for use with conventional feature detector/matcher algorithms, which typically operate on single-channel images. This formulation is inspired by prior work on colour constancy (Ratnasingam and Collins, 2010), which has been used to improve illumination-robustness in a number of mapping and localization pipelines (McManus et al., 2014; Paton et al., 2017; Clement et al., 2017a), and naturally admits a self-supervised training approach since training targets (e.g., quantity of inlier feature matches) can be generated on the fly by the localization pipeline. Importantly, our approach requires only image pairs with some degree of overlap, rather than requiring perfect alignment.

5.2 Related Work

Experience-based navigation (EBN) tackles the problem of navigating through an environment under different appearance conditions by generating a database of consecutive 'experiences' of the environment (Churchill and Newman, 2013; Linegar et al., 2015). By recalling the most relevant or most recent experience in the database, the system can localize itself relative to the chosen reference experience. Paton et al. (2016, 2018) extended this idea by linking experiences in a *spatio-temporal pose graph*, and demonstrated the effectiveness of their multi-experience localization (MEL) approach for long-term autonomous route following through daily and seasonal appearance change. As illustrated in Figure 5.1, MEL localizes against a privileged mapping experience by first localizing against a relevant prior experience and then compounding a chain of relative transformations in the graph. However, the number of intermediate 'bridging' experiences required for reliable long-term localization can be very large, particularly when the appearance of the environment changes over short timeframes (e.g., moving shadows).

As a result, methods for compressing experience graphs are necessary for this approach to scale.

Existing approaches to appearance robustness in metric visual localization have largely focused on illumination invariance. Corke et al. (2013), McManus et al. (2014), Paton et al. (2017), and Clement et al. (2017a) make use of physically motivated transformations as an image pre-processing step to improve feature matching across different illumination conditions for a given feature detector and descriptor. Analogously for direct visual localization, affine models and other simple analytical image transformations have been used to improve the robustness of photometric matching to illumination change (Engel et al., 2015; Park et al., 2017). Other approaches have focused on learning feature descriptors that are robust to certain types of appearance change in autonomous route following applications (McManus et al., 2015; Linegar et al., 2016; Krajnk et al., 2017; Zhang et al., 2018). However, McManus et al. (2015) and Linegar et al. (2016) produce correspondences that are only weakly localized, and Krajnk et al. (2017) and Zhang et al. (2018) require sets of true and false point correspondences to train feature descriptors, which are challenging to obtain at scale over long periods.

Deep image-to-image translation (Isola et al., 2017; Zhu et al., 2017) has recently been applied to the problem of metric localization across appearance change. Gomez-Ojeda et al. (2018) train a convolutional encoder-decoder network to enhance the temporal consistency of image streams captured in environments with high dynamic range. Here the main source of appearance change is the camera itself as it automatically modulates its imaging parameters in response to the local brightness of a static environment. Other work, including our own, has tackled the problem of localization across *environmental* appearance change, with Clement and Kelly (2018) learning a many-to-one mapping onto a privileged appearance condition (discussed in Chapter 4) and Porav et al. (2018) learning multiple pairwise mappings between appearance categories such as day and night. Image-to-image translation has also been applied the related task of appearance-invariant place recognition (Latif et al., 2018; Anoosheh et al., 2019), which typically relies on patch matching or whole-image statistics to identify images corresponding to nearby physical locations rather than estimating the 6-dof pose of the vehicle. While Porav et al. (2018) and Gomez-Ojeda et al. (2018) include additional loss terms to maximize gradient information, these heuristics are not directly tied to the performance of the localization pipeline. Moreover, Clement and Kelly (2018), Porav et al. (2018), and Gomez-Ojeda et al. (2018) require well-aligned pairs of training images exhibiting appearance variation, which are difficult to obtain at scale in the real world, and it is not clear how categorical appearance mappings such as those of Porav et al. (2018), Latif et al. (2018), and Anoosheh et al. (2019) should be applied to continuous appearance change in long-term deployments.

Surrogate-based methods for approximating computationally expensive or non-differentiable objective functions are common in the numerical optimization literature (Koziel et al., 2011). Neural network surrogates in particular have found applications in a variety of domains including computer graphics (Grzeszczuk et al., 1998) and computational oceanography (van der Merwe et al., 2007), where high-fidelity physics simulations are available but expensive to compute. Our method of learning a differentiable loss function is similar in spirit to Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) in that a complex discriminator/loss function is trained using a comparatively simple analytical loss function. It also bears resemblances to perceptual losses (Johnson et al., 2016), where the loss function is derived from the feature activations of a network trained on a proxy task such as image classification.

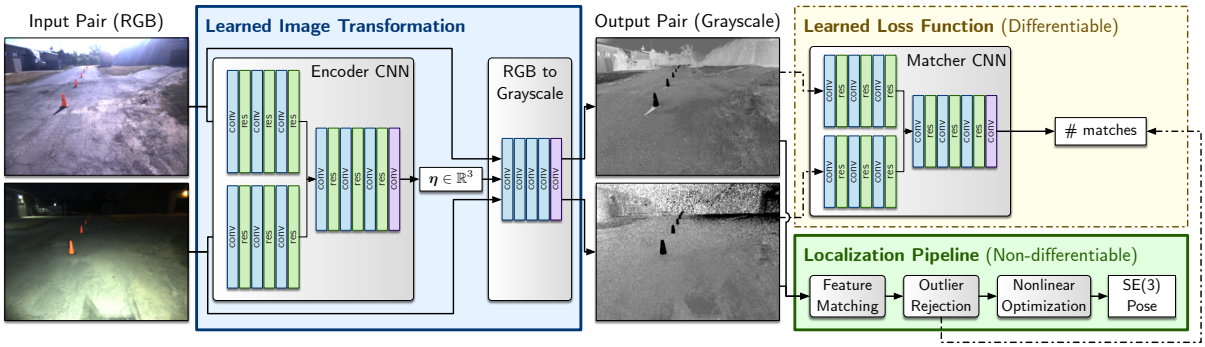


Figure 5.2: We learn an image transformation that improves visual feature matching performance over day-night cycles by maximizing the response of a differentiable proxy network trained to predict the number of inlier feature matches returned by a conventional non-differentiable feature detection/matching algorithm. The learned transformation can then be used as a pre-processing step in a visual localization pipeline to improve its robustness to appearance change.

5.3 Methodology

Our goal in this work is to learn a nonlinear transformation $f : \mathbb{R}^3 \to \mathbb{R}$ mapping the RGB colourspace onto a grayscale colourspace that explicitly maximizes a chosen performance metric of a vision-based localization pipeline. We investigate two approaches to formulating such a mapping: 1) a single function to be applied as a pre-processing step to all incoming images, similarly to Corke et al. (2013), McManus et al. (2014), Paton et al. (2017), and Clement et al. (2017a); and 2) a parameterized function tailored to the specific image pair to be used for localization, where the parameters of this function are derived from the images themselves. Additionally, the functional form of either mapping may be specified analytically (e.g., from physics) or learned from data using a function approximator such as a neural network.

In order to find an optimal colourspace transformation for a given application, we require an appropriate objective function to optimize, which should ideally be tied to the performance of the target localization pipeline. An intuitive choice of objective could be to directly minimize pose estimation error for the entire pipeline relative to ground truth if it is available. In the absence of accurate ground truth data, we might instead choose to maximize the number or quality of feature matches in the front-end of a feature-based localization pipeline. We adopt the latter approach in this work, since high-quality 6-dof ground truth is difficult to obtain reliably over long time scales.

Although it is straightforward to choose a target performance metric to optimize, the most commonly used localization front-ends in robotics rely on non-differentiable components such as stereo matching, nearest-neighbours search, and RANSAC (Fischler and Bolles, 1981), which are incompatible with the gradient-based optimization schemes commonly used in deep learning. In this work we propose to *learn* an objective function by training a deep convolutional neural network (CNN) to act as a *differentiable proxy* to the localization front-end. Specifically, we train a siamese CNN to predict the number of inlier feature matches for a given image pair, where the training targets are generated using a conventional non-differentiable feature detector/matcher algorithm such as `libviso2` (Geiger et al., 2011). This proxy network can then be used to define a fully differentiable objective function, allowing us to train a nonlinear colourspace mapping using gradient-based methods. Finally, the trained image transformation can be used as a pre-processing step in a conventional visual localization pipeline, enabling it to operate more reliably under appearance change. Figure 5.2 summarizes our full data pipeline pictorially.

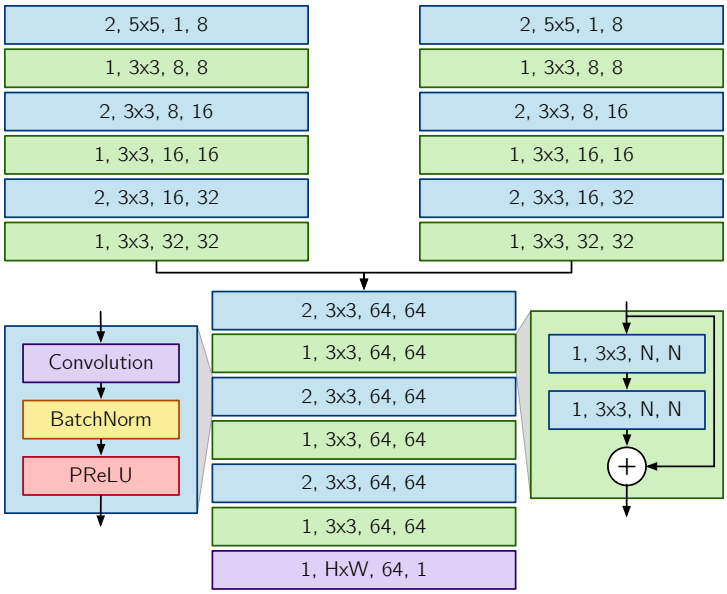


Figure 5.3: Architecture of the differentiable matcher proxy network $\mathcal{M}$. Each block is denoted by stride, convolutional kernel size, input channels, and output channels. The left and right branches share weights. The final output is produced by a fully-connected layer (implemented as a convolution operator), which projects the feature map to a scalar value. The network has approximately 365 thousand parameters.

5.3.1 Differentiable Feature Matcher Proxies

We consider the task of training a CNN $\mathcal{M}$, with parameters $\boldsymbol{\theta}$, to predict the number of inlier feature matches returned by a non-differentiable feature detector/matcher M for a given image pair $(\mathbf{I}_1, \mathbf{I}_2)$. This training objective is a convenient choice for our intended application as it is closely tied to the ability of our visual localization pipeline to operate across appearance change, however it is not by any means the only choice. For example, we could also train a CNN to predict a measure of localization accuracy such as geodesic distance from ground truth on the SE(3) manifold, similarly to the estimator correction framework proposed by Peretroukhin and Kelly (2018). Importantly, this formulation naturally admits a self-supervised training approach as training targets can be generated on the fly by M.

Figure 5.2 (right-hand side) summarizes the training setup for this task. A pair of single-channel images is fed into a conventional feature detection/matching algorithm (e.g., `libviso2`), and a summary statistic is computed such as the quantity of RANSAC-filtered inlier feature matches. This summary statistic forms the training target for a CNN whose task is to predict the statistic for the same image pair. Given enough training pairs, the network should learn a set of convolutional filters that correspond to the types of features and patterns that best predict the presence of inlier matches as reported by M in a given environment. Critically, the proxy network is fully differentiable and can provide a gradient signal to train a nonlinear image transformation in the following step.

Figure 5.3 summarizes the architecture of our matcher proxy network, which is a siamese network built from standard convolutional blocks using batch normalization (Ioffe and Szegedy, 2015) and PReLU nonlinearities (He et al., 2015), augmented with residual connections (He et al., 2016). Each image in the input pair is processed by one of two feature detection branches, which share weights to ensure that both images are mapped onto a common feature space. The outputs of the feature detection branches are concatenated along the channel dimension to be further processed by the remainder of the

network. Each non-residual convolution block downsamples the feature map by a factor of two, allowing for salient features to be learned at multiple scales. The final output is produced by a fully-connected layer (implemented as a convolution operator), which projects the feature map to a scalar value. The network has approximately 365 thousand parameters.

We train $\mathcal{M}$ to fit M in a least-squares sense by minimizing the mean squared error of the predicted match counts in a minibatch of image pairs:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(\mathcal{M}(\mathbf{I}_1^i, \mathbf{I}_2^i; \boldsymbol{\theta}) - M(\mathbf{I}_1^i, \mathbf{I}_2^i)\right)^2. \tag{5.1}$$

Once trained, we can fix $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ use the network $\mathcal{M}$ to define a differentiable loss function based on the predicted performance of M for a given pair of images. We discuss the formulation of this loss function and the image transformations we use it to train in Sections 5.3.2 and 5.3.3.

5.3.2 Physically Motivated Colourspace Transformations

Prior work by Ratnasingam and Collins (2010) has shown that under the assumptions of a single black-body illuminant and an infinitely narrow sensor response function, an appropriately weighted linear combination of the log-responses of a three-channel (e.g., RGB) camera maps to an invariant chromaticity space that is independent of both the intensity and colour temperature of the illuminant, and depends only on the imaging sensor and the materials in the scene:

$$\mathbf{F} = \log \mathbf{I}(\lambda_2) - \alpha \log \mathbf{I}(\lambda_1) - \beta \log \mathbf{I}(\lambda_3), \tag{5.2}$$

where $\mathbf{I}(\lambda_i)$ is the image of sensor responses at peak wavelength λ_i, the weights (α, β) are subject to the constraints

$$\frac{1}{\lambda_2} = \frac{\alpha}{\lambda_1} + \frac{\beta}{\lambda_3} \qquad \text{and} \qquad \beta = (1-\alpha), \tag{5.3}$$

and the indices $i = 1, 2, 3$ are chosen such that $\lambda_1 < \lambda_2 < \lambda_3$ (i.e., red, green, and blue channels).

The image formed from this pixel-wise linear combination of log-responses can then be rescaled to lie in the range $[0, 1]$ (or $[0, 255]$), thereby producing a valid grayscale image that can be further processed by a localization pipeline. This can be done by applying the transformation

$$\mathbf{F} \leftarrow \frac{\mathbf{F} - \min(\mathbf{F})}{\max(\mathbf{F}) - \min(\mathbf{F})}, \tag{5.4}$$

where $\min(\mathbf{F})$ and $\max(\mathbf{F})$ are computed over the entire processed image. Grayscale images generated using this procedure are somewhat resistant to variations in lighting and shadow, and have been shown to improve precision/recall performance on place recognition tasks (Corke et al., 2013), as well as stereo localization quality in the presence of shadows and changing daytime lighting conditions (McManus et al., 2014; Paton et al., 2017; Clement et al., 2017a).

Given the constraints defined by Equation (5.3), the weights (α, β) are completely specified as a function of the imaging sensor. However, in practice, these constraints are relaxed and the parameters (α, β) are tuned to a specific environment-sensor combination where the theoretical parameter values do not perform optimally for the target application. Indeed, Paton et al. (2017) and Clement et al. (2017a)

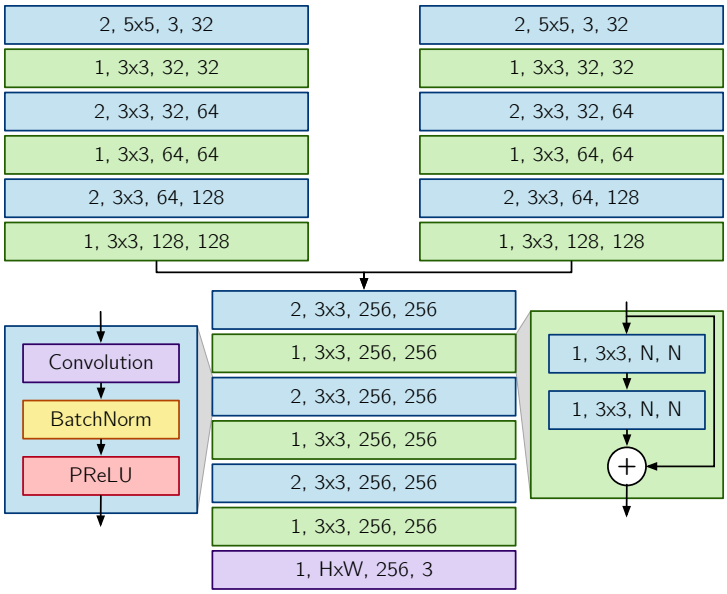


Figure 5.4: Architecture of the pairwise encoder network $\mathcal{E}$. Each block is denoted by stride, convolutional kernel size, input channels, and output channels. The left and right branches share weights. The final output is produced by a fully-connected layer (implemented as a convolution operator), which projects the feature map to a scalar value. The network has approximately 5.8 million parameters.

made use of two sets of parameter values tuned to maximize the stability of SURF features (Bay et al., 2008) in environments where grassy or sandy textures dominate.

We argue that environmental appearance is best thought of as continuous rather than categorical, and that a better approach to selecting the transformation parameters should take into account the content of the specific scene being imaged. Accordingly, we train a second encoder network $\mathcal{E}$ with parameters $\boldsymbol{\phi}$ to predict the optimal values of the transformation parameters (i.e., the parameter values that yield the most inlier feature matches) for a given image pair. Furthermore, we relax the constraints in Equation (5.3) and generalize Equation (5.2) to be of the form

$$\mathbf{F} = \alpha \log \mathbf{I}(\lambda_1) + \beta \log \mathbf{I}(\lambda_2) + \gamma \log \mathbf{I}(\lambda_3), \tag{5.5}$$

where the parameters are computed per image pair by the encoder network as

$$\boldsymbol{\eta} = \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \mathcal{E}(\mathbf{I}_1, \mathbf{I}_2; \boldsymbol{\phi}), \tag{5.6}$$

and Equation (5.5) is applied to both $\mathbf{I}_1$ and $\mathbf{I}_2$. Due to the need to rescale $\mathbf{F}$ to form a valid grayscale image, a degree of freedom in $\boldsymbol{\eta}$ is lost and (α, β, γ) represent the relative mixing proportions of the three colour channels. We enforce $\|\boldsymbol{\eta}\|_1 = 1$ to ensure that $\mathcal{E}$ produces a consistent range of outputs.

Figure 5.4 shows the architecture of $\mathcal{E}$, which is similar to that of $\mathcal{M}$ (Figure 5.3), but takes as input pairs of 3-channel images and outputs a 3-dimensional vector. The channel dimensions grow following the same doubling pattern, starting at 32 and increasing to 256, yielding a network with approximately 5.8 million total parameters.

We train $\mathcal{E}$ to maximize the mean number of inlier feature matches over a minibatch of image pairs, as predicted by $\mathcal{M}$. Equivalently, we can minimize its negation:

$$\mathcal{L}(\boldsymbol{\phi}) = -\frac{1}{N}\sum_{i=1}^{N}\mathcal{M}(\mathbf{F}_1^i, \mathbf{F}_2^i; \boldsymbol{\theta}^*), \tag{5.7}$$

where $\mathbf{F}_1^i, \mathbf{F}_2^i$ are computed from input RGB images $\mathbf{I}_1^i, \mathbf{I}_2^i$ using Equations (5.5) and (5.6).

Rather than rescaling using the minimum and maximum response of each output image as in Equation (5.4), we rescale by the joint mean $\mu_{\mathbf{F}}$ and standard deviation $\sigma_{\mathbf{F}}$ of the output pair and apply a clamping operation to map the output onto $[0, 1]$:

$$\mathbf{F} \leftarrow \frac{1}{2}\left[\frac{\mathbf{F} - \mu_{\mathbf{F}}}{3\sigma_{\mathbf{F}}}\right]_{-1,1} + \frac{1}{2}, \tag{5.8}$$

where we have used the notation $[\cdot]_{a,b} = \min(\max(\cdot, a), b)$. This rescaling scheme allows the model to saturate parts of the output images while still using the full range of valid pixel values. Moreover, it avoids introducing significant sparsity in the gradients through the $\min(\cdot)$ and $\max(\cdot)$ operators, which improves the flow of information when back-propagating error signals through our models.

5.3.3 Learned Nonlinear Colourspace Transformations

While the assumption of a single black-body illuminant is reasonable for daytime navigation where the dominant light source is the sun, it does not hold in many common scenarios such as nighttime driving. Moreover, the assumption of an infinitely narrow sensor response is unrealistic for real cameras. As an alternative to the physically motivated transformation outlined in Section 5.3.2, we investigate the possibility of learning a bespoke nonlinear mapping that maximizes matchability for a particular combination of imaging sensor, pipeline, and environment. We parameterize this mapping using a small neural network $\mathcal{T}$ with parameters $\boldsymbol{\psi}$, operating independently on each pixel of each input RGB image. We structure $\mathcal{T}$ as a multilayer perceptron (MLP) with five hidden layers of 16 nodes each, implemented using 1×1 convolutions and PReLU nonlinearities, totalling approximately 1,200 parameters.

We consider two versions of this MLP-based transformation, both with and without incorporating an additional pairwise context feature obtained from encoder network $\mathcal{E}$. In the case where $\mathcal{E}$ is used, the input to the network $\mathcal{T}$ becomes the concatenation of the input RGB image and the parameters $\boldsymbol{\eta}$ along the channel dimension, and the first convolutional layer of $\mathcal{T}$ is modified accordingly. We train $\mathcal{T}$ and $\mathcal{E}$ (if used) jointly by minimizing a similar loss function to Equation (5.7):

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\psi}) = -\frac{1}{N}\sum_{i=1}^{N}\mathcal{M}(\mathbf{F}_1^i, \mathbf{F}_2^i; \boldsymbol{\theta}^*) \tag{5.9}$$

where instead of computing $\mathbf{F}_1^i, \mathbf{F}_2^i$ using Equations (5.5) and (5.6), we have

$$\mathbf{F}_j^i = \begin{cases} \mathcal{T}(\mathbf{I}_j^i, \boldsymbol{\eta}^i; \boldsymbol{\psi}) & \text{if encoder network } \mathcal{E} \text{ is used} \\ \mathcal{T}(\mathbf{I}_j^i; \boldsymbol{\psi}) & \text{otherwise,} \end{cases} \tag{5.10}$$

and $\mathbf{F}_j^i$ is again rescaled using Equation (5.8) to fill the range of valid grayscale values.

Table 5.1: Summary of experiments in Chapter 5.

Section	Description	Dataset(s)
Section 5.4.2	Evaluation of matcher proxy network prediction accuracy and correlation with actual feature matcher performance.	Virtual KITTI, In The Dark, Multi-Season
Section 5.4.3.1	Comparison of feature matching and localization results using each RGB-to-grayscale mapping on synthetic daytime illumination change.	Virtual KITTI
Section 5.4.3.2	Comparison of feature matching and localization results using each RGB-to-grayscale mapping on real-world illumination change over a full day-night cycle.	In The Dark
Section 5.4.4	Comparison of feature matching and localization results using each RGB-to-grayscale mapping on real-world seasonal appearance change.	Multi-Season

5.4 Experiments

We conducted experiments on synthetic and real-world datasets to validate and compare each approach, which are summarized in Table 5.1. Specifically, we evaluated the ability of our matcher proxy network to capture the performance of `libviso2` feature matching across viewpoint and appearance changes, as well as the effect of each image transformation on overall feature matching and localization performance. When evaluating feature matching and localization, we assumed that we had a prior on the vehicle's topological location in the map, such that we could reliably identify the nearest vertex in the pose graph. This is typical for autonomous visual route-following systems, where the topological prior is derived by dead reckoning from a previous successful localization, or using place recognition or GNSS in the event that the system becomes lost (see, e.g., Paton et al. (2018)).

We refer to the generalized color-constancy model of Ratnasingam and Collins (2010) (Section 5.3.2) as "SumLog" and "SumLog-E" , where the latter uses Equation (5.6) to derive $\boldsymbol{\eta}$, and the former uses a constant value for $\boldsymbol{\eta}$ that maximizes inlier feature matches over the training set (similarly to the approach taken by Clement et al. (2017a) and Paton et al. (2017)). Analogously, we refer to the learned multilayer perceptron models (Section 5.3.3) as "MLP" and "MLP-E", where the latter incorporates $\mathcal{E}$ and the former does not. We refer to the standard RGB-to-grayscale transformation as "Gray".[1]

In each case, training proceeded in two stages. First, we pre-trained the matcher proxy network $\mathcal{M}$ using standard grayscale images. Training labels were generated using the open-source `libviso2` library (Geiger et al., 2011) to detect and match features, and the eight-point RANSAC algorithm to reject outlier matches. Second, we trained $\mathcal{E}$ and/or $\mathcal{T}$ using the matchability loss defined in Equation (5.7) or Equation (5.9). To ensure that $\mathcal{M}$ accurately predicted feature match counts for the output images, which differ significantly from standard grayscale images, we continued to update $\boldsymbol{\theta}^*$ in an alternating fashion by training $\mathcal{M}$ using the output images at each iteration. In this respect, our training procedure resembles the procedure used in adversarial learning, where the generator network is trained based on the output of the most recent discriminator/loss network and vice versa, until training converges to an equilibrium state (Goodfellow et al., 2014). All models were implemented in PyTorch (Paszke et al.,

[1]We refer specifically to the ITU-R 601-2 luma transform implemented by the `Pillow` library: $L = 0.299R + 0.587G + 0.114B$.

Figure 5.5: Sample frames from sequence `VKITTI/0001` under various illumination conditions (Gaidon et al., 2016).

Figure 5.6: Aerial view and corresponding sample frames from multiple traverses of the 250 m path from the UTIAS In The Dark dataset (Paton et al., 2016), which exhibits substantial variation in illumination conditions over a full day-night cycle.

2017) and trained for 10 epochs with a batch size of 8, using the Adam optimizer (Kingma and Ba, 2015) configured with default parameters and a learning rate of 10^{-4}.

5.4.1 Datasets

We evaluated our approach using both synthetic and real-world long-term vision datasets that exhibit severe illumination change as well as seasonal appearance variations. This section summarizes each dataset as well as our approach to creating appropriate training and testing sets.

Virtual KITTI Dataset The Virtual KITTI (`VKITTI`) dataset (Gaidon et al., 2016) is a synthetic reconstruction of a portion of the KITTI vision benchmark (Geiger et al., 2013), consisting of five sets of camera trajectories with RGB-D imagery (1242×375 resolution) rendered under a variety of simulated illumination conditions including "Morning", "Sunset", "Overcast" and "Clone". The "Clone" condition is intended to replicate the conditions found in the original data. This dataset is a convenient validation tool as it provides perfect data association and a range of daytime illumination conditions. Figure 5.5 shows corresponding sample images under each condition for sequence `VKITTI/0001`. For each trajectory, we trained models using images from the others, scaled to 636×192 resolution using bilinear interpolation. Since this dataset provides corresponding images from identical viewpoints, we augmented the training data to include viewpoint changes by associating each training image in one appearance condition with a randomly selected image from a window around the corresponding image in the other condition.

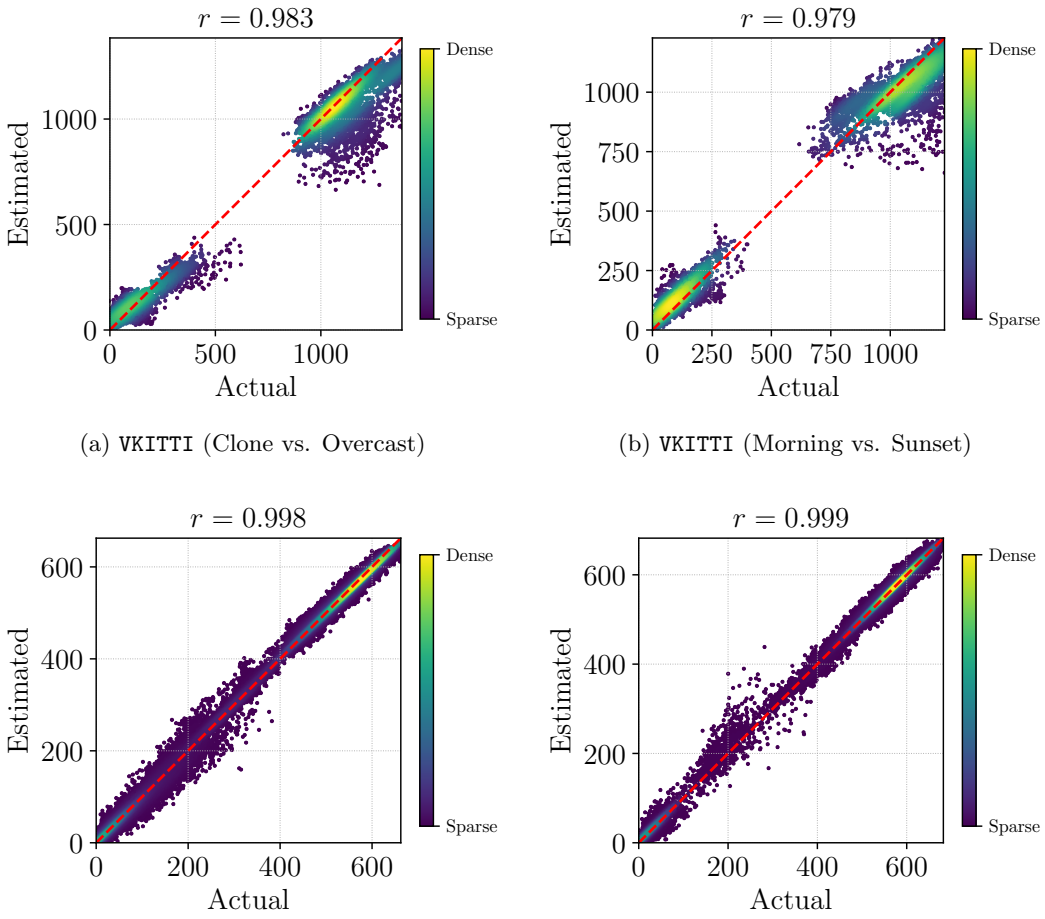
Figure 5.7: Corresponding sample frames from multiple traverses of the 160 m path from the UTIAS Multi-Season dataset (Paton, 2017), which exhibits both daily illumination change as well as seasonal changes to the appearance of the environment.

UTIAS In The Dark Dataset The UTIAS In The Dark (`InTheDark`) dataset (Paton et al., 2016) consists of stereo imagery (512×384 resolution) of a 250 m outdoor loop traversed autonomously over a continuous 30-hour period with the aid of on-board headlights to illuminate the scene at night. Figure 5.6 shows an aerial view of the path as well as sample images from the manual "Map" traverse and several autonomous traverses spanning the full day-night cycle. Repeat sequence images were matched against map images using the multi-experience localization system of Paton et al. (2016), which yields cross-appearance image correspondences with overlapping fields of view but non-identical poses. We trained our models using corresponding pairs of left-camera images from 66 autonomous repeat sequences and tested on an additional 7 sequences held out from the dataset (shown in Figure 5.6), yielding approximately 68,000 image pairs in the training set and 10,000 in the test set. All images were scaled to 256×192 resolution using bilinear interpolation.

UTIAS Multi-Season Dataset The UTIAS Multi-Season (`MultiSeason`) dataset (Paton, 2017) consists of stereo imagery (512×384 resolution) of a 160 m outdoor loop traversed autonomously over a continuous four month period, exhibiting significant changes in illumination, snow cover, and vegetation growth. Figure 5.7 shows corresponding sample images from the manual "Map" traverse and several autonomous traverse sequences. Similarly to the `InTheDark` dataset, repeat sequence images were matched against map images using the multi-experience localization system of Paton et al. (2016). We trained our models using left-camera imagery from 154 autonomous repeat sequences and tested on an additional 17 sequences held out from the dataset (shown in Figure 5.7), yielding approximately 97,000 training pairs and 12,000 test pairs. All images were scaled to 256×192 resolution using bilinear interpolation.

5.4.2 Feature Matcher Approximation

We trained the matcher proxy network $\mathcal{M}$ in a self-supervised manner to predict the number of feature matches for pairs of overlapping images captured under different illumination conditions and from potentially different poses. Training labels were generated on the fly for each image pair using the open-source



(a) `VKITTI` (Clone vs. Overcast)

(b) `VKITTI` (Morning vs. Sunset)

(c) `InTheDark`

(d) `MultiSeason`

Figure 5.8: Estimated vs. actual match counts for matcher proxy network $\mathcal{M}$ after ten epochs of training on each dataset, colour-coded by relative density. Match counts are aggregated over all test sequences and include self-matches. The dashed red line corresponds to perfect agreement of $\mathcal{M}$ with `libviso2`. In each case, the inlier match counts predicted by $\mathcal{M}$ are strongly correlated with the true values.

`libviso2` library (Geiger et al., 2011) in monocular flow matching mode with default parameters, using the eight-point RANSAC algorithm to reject outlier matches. In practice we trained $\mathcal{M}$ to minimize Equation (5.1) over all combinations of input images: $(\mathbf{I}_1, \mathbf{I}_1)$, $(\mathbf{I}_1, \mathbf{I}_2)$ and $(\mathbf{I}_2, \mathbf{I}_2)$.

Figure 5.8 plots actual and estimated feature match counts after ten epochs of pre-training, aggregated across test sets within each dataset. In each case, the test-time inlier match counts predicted by $\mathcal{M}$ were strongly correlated with the true performance of `libviso2`. These results indicate that our approach generalizes well and that $\mathcal{M}$ is capturing salient properties of feature matching rather than memorizing training examples. Accordingly, we can expect that the trained networks will provide a sufficiently high quality gradient signal to learn useful image transformations.

5.4.3 Localization Across Daily Illumination Change

This section discusses the impact of our method on localization across daily illumination change, using the Virtual KITTI and UTIAS In The Dark datasets. In both cases, we trained the encoder network

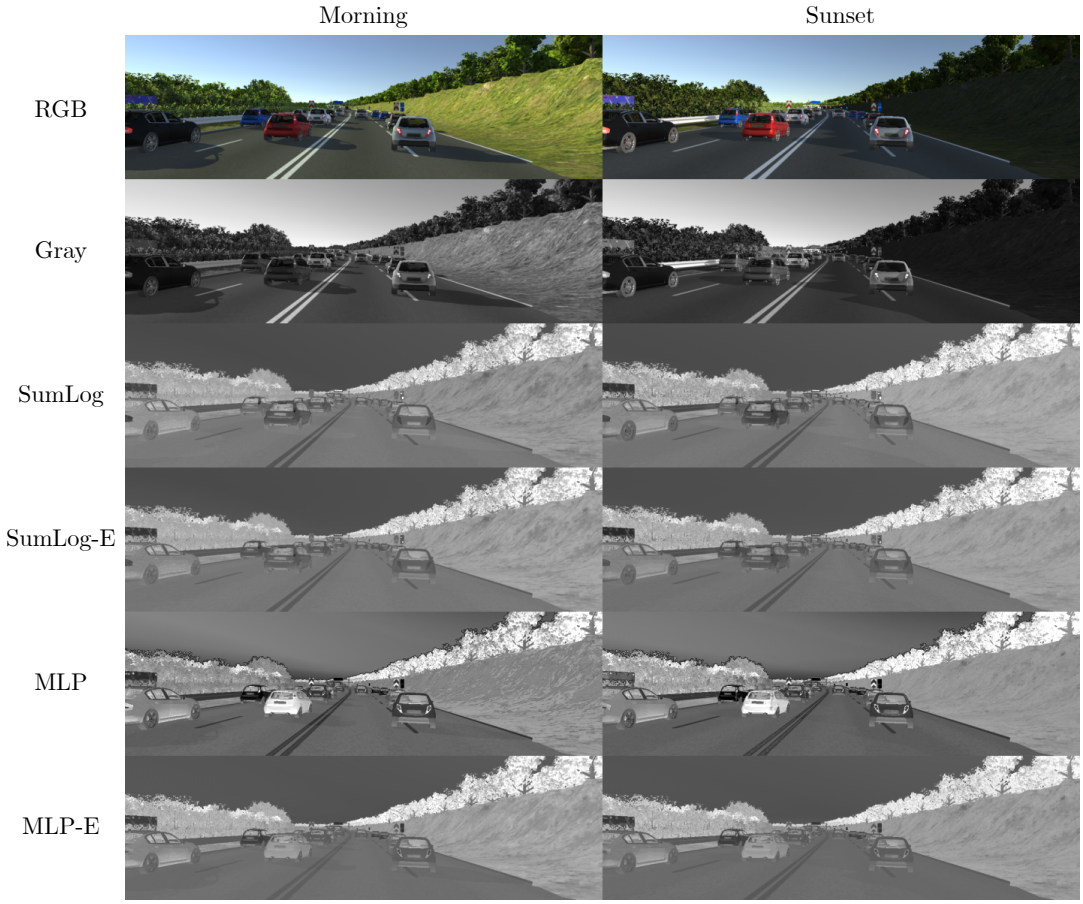
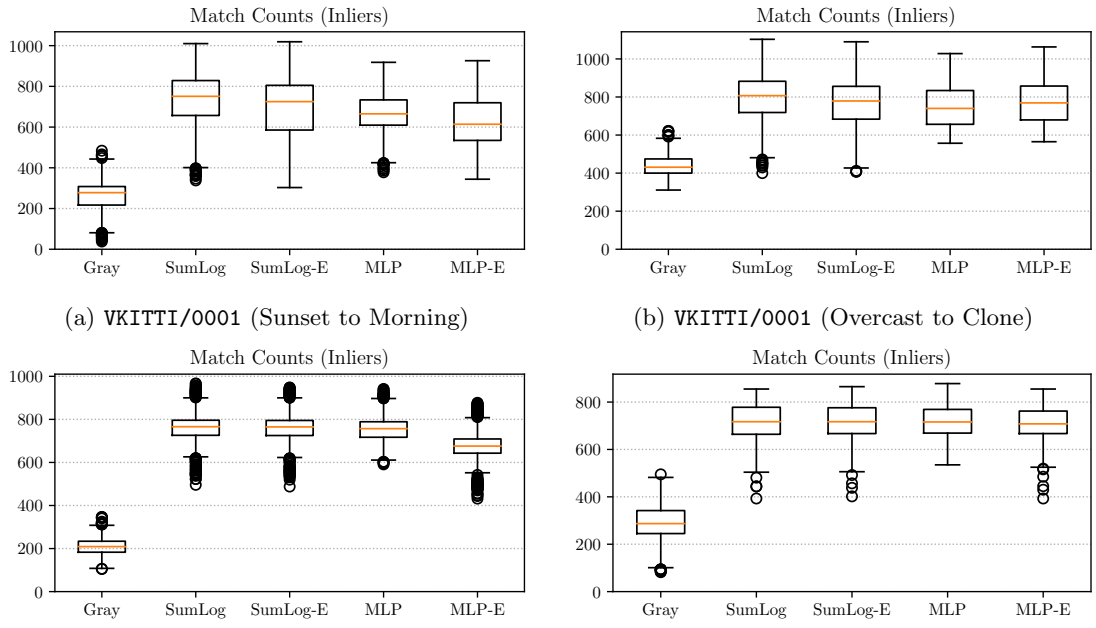


Figure 5.9: Sample RGB images and corresponding outputs of each RGB-to-grayscale transformation for sequence `VKITTI/0020` (Sunset to Morning).

$\mathcal{E}$ and/or the image transformer network $\mathcal{T}$ using the matchability loss defined in Equation (5.7) or Equation (5.9), fixing the parameters of $\mathcal{M}$ to their optimized values $\boldsymbol{\theta}^*$. To ensure that $\mathcal{M}$ accurately predicted feature match counts for the output images, which differ significantly from standard grayscale images, we continued to update $\boldsymbol{\theta}^*$ in an alternating fashion by training $\mathcal{M}$ using the output images at each iteration.

5.4.3.1 Virtual KITTI Dataset

Figure 5.9 shows sample RGB images from the `VKITTI/0020` Morning and Sunset sequences and their corresponding grayscale mappings using each RGB-to-grayscale transformation. The figure highlights the visual consistency of each output pair, particularly in terms the automatic removal of shadows and the enhancement of local contrast around edges, corners, and other structures pertinent to feature detection and matching. These properties emerged naturally as a result of minimizing the matchability losses defined by Equations (5.7) and (5.9) and were not explicit goals of the learning process. Moreover, the output images were specifically optimized for feature detection and matching using `libviso2` and RANSAC rather than being optimized for a heuristic such as total gradient information, which is only loosely tied to the performance of the localization pipeline.



(a) `VKITTI/0001` (Sunset to Morning)

(b) `VKITTI/0001` (Overcast to Clone)

(c) `VKITTI/0020` (Sunset to Morning)

(d) `VKITTI/0020` (Overcast to Clone)

Figure 5.10: Distribution of inlier `libviso2` matches for corresponding image pairs from the Virtual KITTI dataset, using each RGB-to-grayscale transformation. Orange lines indicate median values.

Feature Matching Performance Figure 5.10 compares the distributions of actual inlier `libviso2` feature matches using each RGB-to-grayscale transformation on the `VKITTI/0001` and `0020` trajectories, while Table 5.2 reports these results numerically for all five trajectories. We observe that all four transformations roughly tripled the mean number of inlier feature matches between corresponding images in the Sunset and Morning conditions compared to standard grayscale images. We achieved similar feature matching performance between the Overcast and Clone conditions, but note that the gains are less dramatic due to the absence of strong directional lighting in the Overcast condition, which simplified the matching task using standard Gray images.

We note that the greatest improvements were generally obtained from the SumLog and SumLog-E transformations, but that the pairwise encoder network did not confer a significant benefit on the `VKITTI` sequences and could even be detrimental at times. We attribute these results to three factors. First, the synthetic images provided in the Virtual KITTI dataset satisfy the assumptions of a single black-body illuminant (i.e., daytime illumination from the sun) and an infinitely narrow sensor response (i.e., a simulated camera) used to derive Equation (5.5), so we can expect the SumLog methods to perform optimally on this dataset. Second, the homogeneity of the dataset in terms of the material textures used to render each environment means that we can expect a single transformation to work well across all five trajectories. Indeed these results are consistent with the findings of Corke et al. (2013), McManus et al. (2014), Clement et al. (2017a), and Paton et al. (2017), where one or two sets of parameter values were sufficient to achieve good performance across varying daytime conditions. Finally, the relatively small size of the dataset means that our networks were more likely to overfit to the training data, which may partly explain why the high-capacity encoder network appears to have been detrimental in many cases, and why the MLP methods did not perform as well as their counterpart SumLog methods.

Table 5.2: Actual inlier feature matches using `libviso2` and each RGB-to-grayscale transformation on sequences from the Virtual KITTI dataset. The best results are highlighted in bold.

	Inlier Feature Matches $\mu(\sigma)$				
Test Sequence	Gray	SumLog	SumLog-E	MLP	MLP-E
`VKITTI/0001`					
Sunset to Morning	262 (82)	**726 (136)**	689 (157)	661 (108)	623 (116)
Overcast to Clone	444 (58)	**790 (129)**	767 (125)	747 (106)	770 (107)
`VKITTI/0002`					
Sunset to Morning	240 (22)	**812 (67)**	803 (70)	774 (65)	702 (62)
Overcast to Clone	290 (41)	739 (67)	757 (76)	755 (65)	**764 (84)**
`VKITTI/0006`					
Sunset to Morning	125 (33)	669 (44)	**735 (43)**	711 (37)	642 (41)
Overcast to Clone	142 (33)	647 (39)	**666 (43)**	566 (47)	546 (38)
`VKITTI/0018`					
Sunset to Morning	234 (53)	**817 (36)**	816 (37)	745 (37)	731 (40)
Overcast to Clone	311 (46)	548 (52)	**555 (50)**	450 (42)	486 (39)
`VKITTI/0020`					
Sunset to Morning	210 (39)	**762 (69)**	758 (71)	756 (62)	675 (69)
Overcast to Clone	287 (78)	716 (71)	716 (72)	**718 (62)**	708 (64)

5.4.3.2 UTIAS In The Dark Dataset

Figure 5.11 shows sample RGB images from the challenging sequence `InTheDark/0041` and their corresponding grayscale mappings using each RGB-to-grayscale transformation. In this case the matching task involved vastly different illumination conditions as the scene was illuminated at night using headlights with a limited range and a different spectrum to the sun. We see again that each model produced grayscale image pairs that were visually more consistent than the standard grayscale transformation, and that local illumination variations such as shadows, uneven lighting, and specular reflection were minimized as a result of optimizing Equation (5.7) or Equation (5.9). We note that the transformed nighttime images were significantly noisier than their daytime counterparts due to the much lower signal-to-noise ratio in underexposed parts of the image.

Feature Matching Performance Figure 5.12 compares the distributions of actual inlier `libviso2` feature matches using each RGB-to-grayscale transformation on representative test sequences, while Table 5.3 reports numerical results for all seven test sequences. All four methods significantly increased the mean number of inlier matches across each sequence, with the greatest improvements generally obtained from the SumLog-E transformation. Sequences `InTheDark/0006` and `0071` were exceptions in that the Gray transformation yielded the best feature matching performance, likely due to the fact that both sequences were recorded under very similar conditions to the "Map" sequence.

Unlike `VKITTI`, the pairwise encoder network provided a noticeable performance boost to both the SumLog and MLP transformations on the `InTheDark` sequences. We attribute this difference to more variation in illumination and terrain, as a single transformation is less likely to perform well under more varied conditions. We also note that the MLP-E transformation generally performed similarly to the SumLog-E transformation on this dataset. This suggests that, in spite of key assumptions being broken, a linear combination of log-responses may in fact be an optimal solution for this problem, and that a careful choice of weights is the key to robust cross-appearance feature matching across day-night cycles.

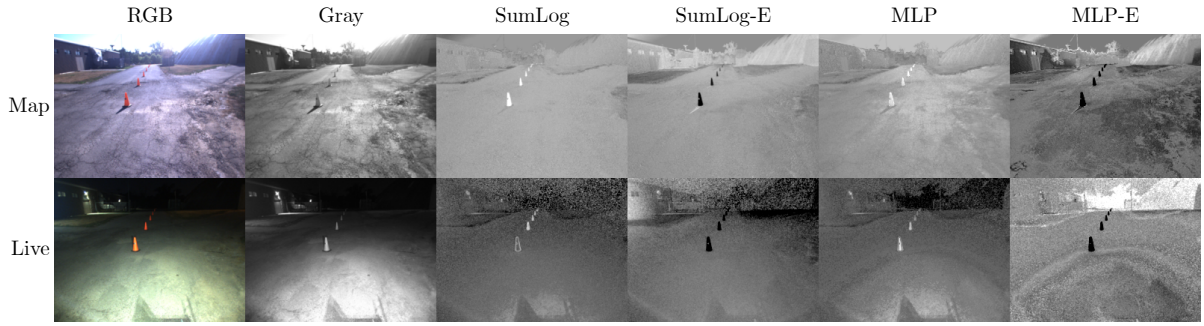
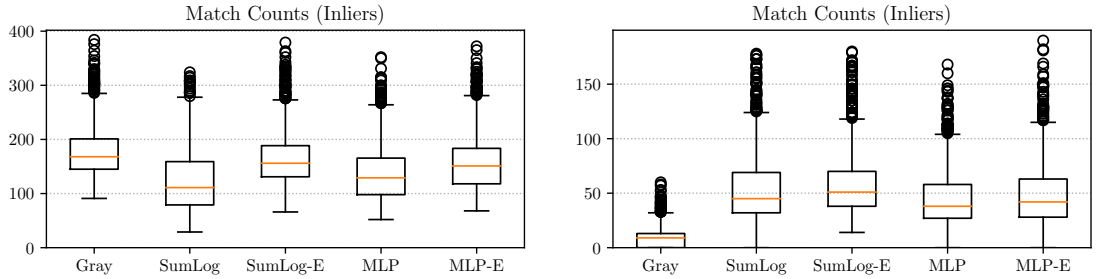
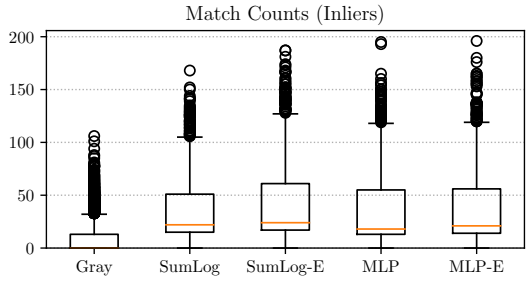


Figure 5.11: Sample RGB image pair and corresponding outputs of each RGB-to-grayscale transformation for sequence `InTheDark/0041` (Night to Morning).

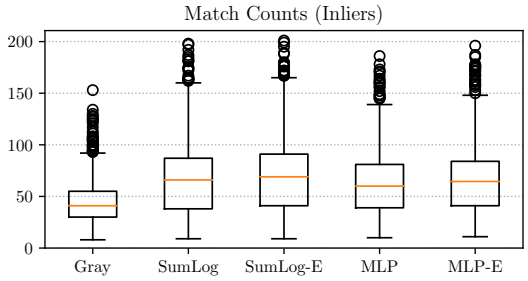


(a) `InTheDark/0006` (Morning to Morning)

(b) `InTheDark/0027` (Evening to Morning)



(c) `InTheDark/0041` (Night to Morning)



(d) `InTheDark/0089` (Afternoon to Morning)

Figure 5.12: Distribution of inlier `libviso2` matches for corresponding image pairs from the UTIAS In The Dark dataset, using each RGB-to-grayscale transformation. Orange lines indicate median values.

Table 5.3: Actual inlier feature matches using `libviso2` and each RGB-to-grayscale transformation on sequences from the UTIAS In The Dark dataset. The best results are highlighted in bold.

	Inlier Feature Matches $\mu(\sigma)$				
Test Sequence	Gray	SumLog	SumLog-E	MLP	MLP-E
`InTheDark`					
Map (08:54)	-	-	-	-	-
`0006` (09:46)	**178 (48)**	125 (56)	165 (51)	138 (52)	158 (53)
`0027` (18:36)	9 (9)	52 (29)	**57 (26)**	44 (25)	48 (28)
`0041` (21:48)	10 (16)	33 (25)	**39 (31)**	33 (29)	35 (29)
`0058` (05:48)	99 (34)	95 (47)	**114 (43)**	97 (44)	113 (40)
`0071` (09:18)	**181 (44)**	126 (52)	167 (47)	141 (48)	161 (48)
`0083` (14:01)	53 (21)	76 (39)	**83 (37)**	82 (37)	**83 (37)**
`0089` (16:43)	45 (22)	67 (35)	**70 (36)**	63 (31)	68 (33)

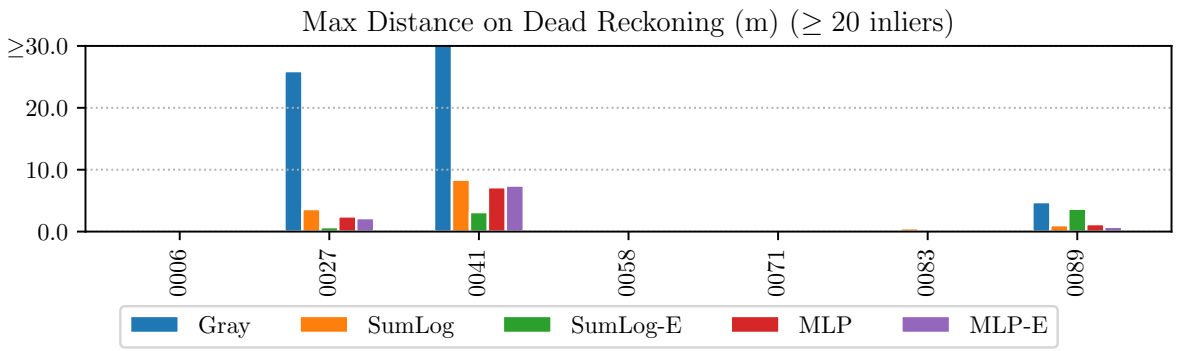


Figure 5.13: Maximum distances travelled on dead reckoning for each test sequence of the UTIAS In The Dark dataset, based on a threshold of 20 inlier feature matches against the "Map" sequence.

Localization Performance We evaluated localization performance in an autonomous route-following context by examining the distribution of distances in each sequence that would have been navigated using dead reckoning (e.g., visual odometry) as a result of failing to localize against the map. Intuitively, the longer the system must rely on dead reckoning, the more likely it is that the system will diverge from the desired route and require manual intervention to resume course. A typical criterion for requiring manual intervention is dead reckoning in excess of 10 meters, but other thresholds may be used depending on the application and the accuracy of the underlying dead reckoning method.

In order to make this assessment, we assumed that we would have a reasonable prior on the vehicle's topological location in the map (e.g., from dead reckoning), such that we could reliably identify the nearest vertex in the pose graph. Given this prior, we deemed a localization attempt to be successful if the number of inlier feature matches between the live image and the nearest map image exceeded a chosen threshold (e.g., 10, 20, or 30 inlier matches). A threshold of 10 inlier matches, for example, is a common criterion for localization success in many vision-based systems (e.g., the `libviso2` visual odometry pipeline) as it implies the existence of a self-consistent inlier set larger than the minimal set needed to estimate the vehicle pose.

Figure 5.13 compares the maximum distances travelled using dead reckoning for each `InTheDark` test sequence based on a relatively conservative threshold of 20 inlier feature matches against the "Map" sequence. For sequences `InTheDark/0006`, `0058`, `0071`, and `0083`, which were captured in the morning and early afternoon, the standard Gray images and each set of transformed images yielded sufficiently many inlier matches with sufficient consistency to achieve continuous localization (i.e., zero meters travelled on dead reckoning) throughout the route. However, sequences `InTheDark/0027` (evening) and `0041` (night) presented significant difficulty for localization, and could be each expected to require manual intervention even for a generous dead reckoning threshold of 20 meters. This difficulty was substantially alleviated using any of the four image transformations, each of which yielded maximum distances on dead reckoning of less than 10 meters on all seven test sequences. Notably, the SumLog-E transformation achieved maximum dead reckoning distances below four meters on all seven test sequences.

Table 5.4 summarizes these results numerically based on thresholds of 10, 20, and 30 inlier matches. As the threshold became more conservative, localization failures became more common, as expected. However, the proposed image transformations continued to provide substantially more robust localization performance compared to standard grayscale images. For example, we achieved a maximum distance on

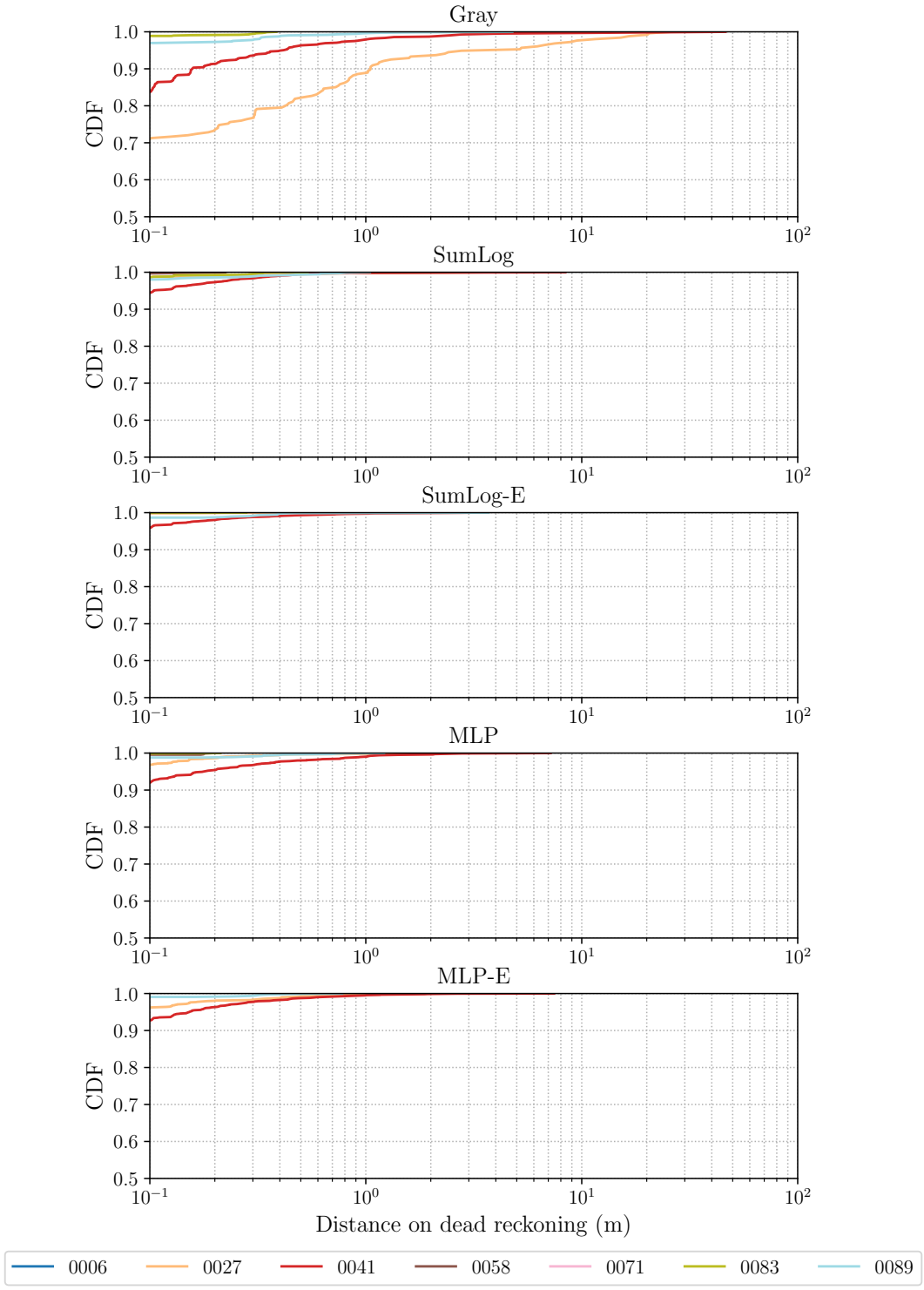


Figure 5.14: Empirical cumulative distribution functions of distances travelled on dead reckoning for each test sequence of the UTIAS In The Dark dataset, based on a threshold of 20 inlier feature matches against the "Map" sequence.

dead reckoning of 10.0 meters using the SumLog-E method on sequence `InTheDark/0041` (night) with a threshold of 30 inlier matches. In contrast, the system would be forced to rely on dead reckoning for approximately 40% of the route using the standard Gray method in this case.

We can also examine the empirical cumulative distribution functions (CDFs) of distances travelled on dead reckoning for each `InTheDark` test sequence, which are plotted in Figure 5.14 based on a threshold of 20 inlier matches against the map. These plots indicate the proportion of the route that would have been traversed without relying on dead reckoning for more than the distance indicated on the horizontal axis. From the figure, we see the benefit to localization performance achieved using any of the four image transformations, all of which yielded localization failures in excess of 10 cm on less than 10% of the route, and no failures in excess of 10 m, for all seven test cases. In contrast, the results using standard Gray images indicate localization failures in excess of 10 cm on 29% of the route for sequence `InTheDark/0027` and 16% of the route for sequence `InTheDark/0041`, as well as the existence of localization failures in excess of 10 m on both sequences, which would trigger a manual intervention.

These results imply that continuous 6-dof visual localization across day-night cycles is achievable using only a single mapping experience and a relatively simple image pre-processing step, even using fairly conservative localization criteria. This represents a dramatic reduction in data requirements to scale experience-based localization to very long deployments.

Table 5.4: Maximum distances travelled on dead reckoning for each test sequence of the UTIAS In The Dark dataset, based on various thresholds of inlier feature matches against the "Map" sequence. The best results are highlighted in bold.

	Maximum Distance on Dead Reckoning (m)														
	$\geq$ 10 Inliers					**$\geq$ 20 Inliers**					**$\geq$ 30 Inliers**				
	G[1]	SL[2]	SL-E[2]	MLP	MLP-E	G[1]	SL[2]	SL-E[2]	MLP	MLP-E	G[1]	SL[2]	SL-E[2]	MLP	MLP-E
`InTheDark`															
Map (08:54)	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
`0006` (09:46)	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.1	**0.0**	**0.0**	**0.0**
`0027` (18:36)	7.5	0.7	**0.0**	0.3	0.3	25.9	3.6	**0.7**	2.5	2.2	101.1	10.5	**0.7**	8.3	7.8
`0041` (21:48)	14.9	0.5	**0.2**	0.9	1.4	46.3	8.4	**3.2**	7.2	7.4	104.5	15.1	**10.0**	18.6	15.7
`0058` (05:48)	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.2	**0.0**	0.2	**0.0**	**0.0**	0.6	0.3	0.6	**0.0**
`0071` (09:18)	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
`0083` (14:01)	**0.0**	0.2	**0.0**	**0.0**	**0.0**	0.4	0.5	0.3	0.2	**0.0**	2.2	4.0	**0.3**	0.4	0.5
`0089` (16:43)	0.3	0.3	0.2	**0.0**	**0.0**	4.8	1.0	3.7	1.2	**0.8**	6.1	6.1	4.7	**2.9**	**2.9**

[1] G: Gray
[2] SL: SumLog

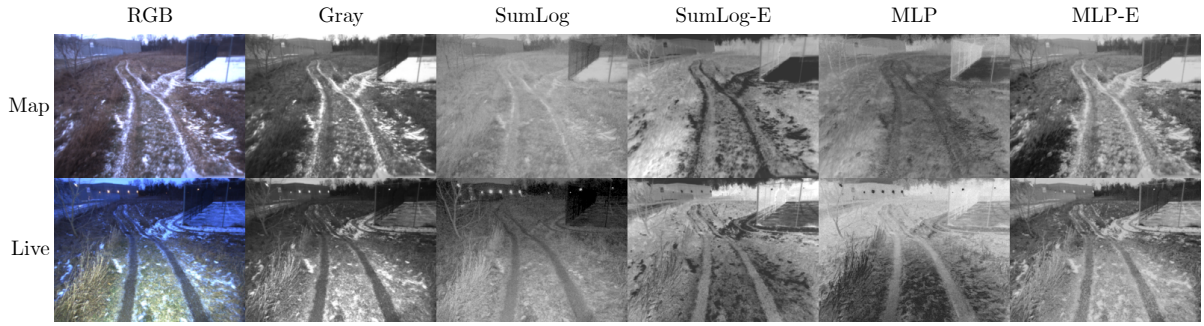
5.4.4 Localization Across Seasonal Appearance Change

Seasonal appearance change is extremely challenging for metric visual localization as the structure of the environment physically changes over time due to snow cover, vegetation growth cycles, and other effects (see Figure 5.7). These changes significantly reduce the availability of stable visual features that can be reliably matched over periods of weeks or months. However, the appearance of persistent structures such as buildings, fences, and signs varies largely as a function of illumination, and these variations can be captured using the image transformations discussed in this chapter. In this section, we repeat our experiments using the UTIAS Multi-Season dataset to assess the extent to which colourspace transformations can improve cross-seasonal localization. The training procedures used in these experiments are identical to those described in Sections 5.4.2 and 5.4.3. Figure 5.15 shows the outputs of each model on four representative `MultiSeason` test sequences exhibiting varying degrees of change in illumination, snow cover, and vegetation.

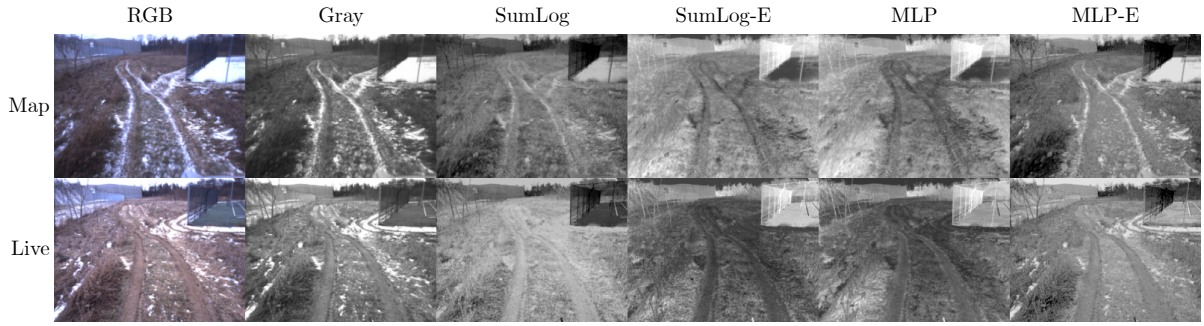
Feature Matching Performance Figure 5.16 shows the distributions of inlier feature matches for each of the four sequences depicted in Figure 5.15, which are summarized numerically for all 17 test sequences in Table 5.5. We see that for sequences `MultiSeason/0123` and `0152`, the SumLog-E and MLP-E transformations significantly increased the mean number of inlier matches compared to the standard Gray transformation, however they had virtually no effect for sequences `MultiSeason/0224` and `0638`. We attribute these differences to two factors. First, sequences `MultiSeason/0224` and `0638` represent substantially further departures from the original "Map" condition than do the other two sequences. Since the dataset was collected in a largely unstructured environment, persistent structures are scarce, and both the appearance and geometry of the ground undergo significant changes due to snow cover and vegetation growth. Second, approximately 40% of the Multi-Season dataset was collected during the first three weeks of testing, with the remainder spread out over the course of approximately four months. As a result, the dataset is biased overall towards winter imagery, which implies that our trained models can be expected to perform better in winter conditions than in other seasons. With a more balanced dataset, we might expect some improvement in performance on springtime sequences such as `MultiSeason/0638`, however this would not account for the more important structural changes in the environment.

Table 5.5 shows that, in terms of mean inlier match counts, the SumLog-E and MLP-E methods generally achieved the best results for this dataset. These results underscore the benefit of the pairwise encoder network, as a single transformation cannot easily account for the variability in terrain and illumination that this dataset exhibits. This was especially apparent for sequence `MultiSeason/0123`, where the one-model-fits-all SumLog and MLP methods failed to account for nighttime illumination with headlights, as the models were optimized based on a training set consisting predominantly of daytime imagery. In contrast, the encoder network provided the SumLog-E and MLP-E models with additional capacity to handle a wider range of appearance conditions, resulting in twice as many inlier matches compared to the standard Gray method, and eight times as many as the SumLog and MLP models.

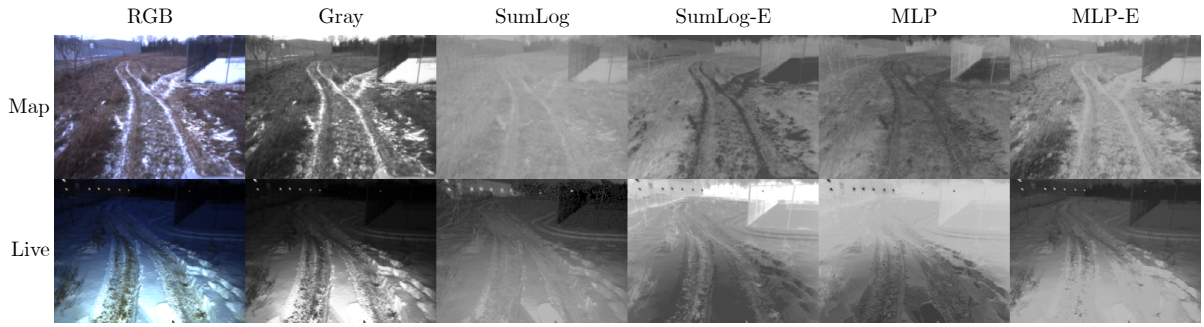
Further, the SumLog-E and MLP-E methods often achieved comparable increases in the mean number of inlier feature matches. Consistent with our earlier experiments with the Virtual KITTI and UTIAS In The Dark datasets, this suggests that a linear combination of log-responses represents the optimal functional form for producing maximally matchable grayscale images from RGB images, and that there is little to be gained by replacing this function with a multilayer perceptron model.



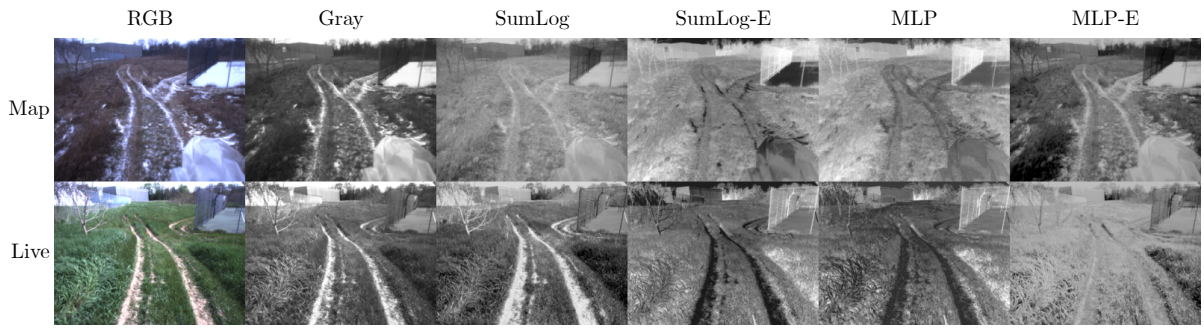
(a) `MultiSeason/0123` (Nighttime with headlights, similar snow cover)



(b) `MultiSeason/0152` (Daytime, reduced snow cover)

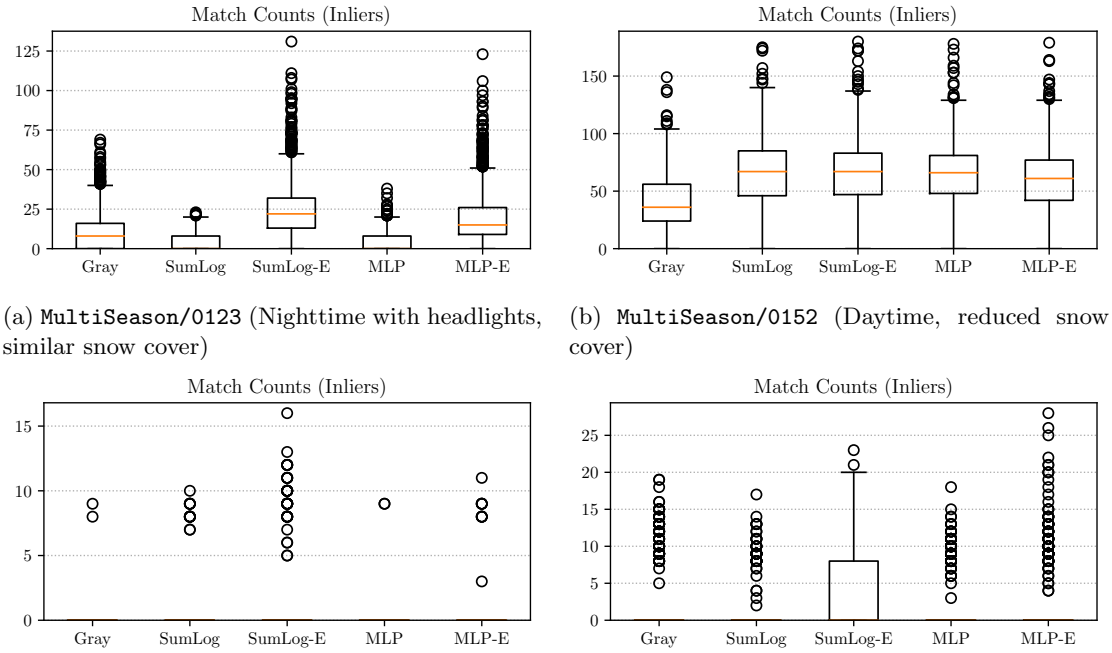


(c) `MultiSeason/0224` (Nighttime with headlights, heavy snow cover)



(d) `MultiSeason/0638` (Daytime, new vegetation, no snow cover)

Figure 5.15: Sample RGB image pairs and corresponding outputs of each RGB-to-grayscale transformation for representative sequences from the UTIAS Multi-Season dataset.



(a) `MultiSeason/0123` (Nighttime with headlights, similar snow cover)

(b) `MultiSeason/0152` (Daytime, reduced snow cover)

(c) `MultiSeason/0224` (Nighttime with headlights, heavy snow cover)

(d) `MultiSeason/0638` (Daytime, new vegetation, no snow cover)

Figure 5.16: Distribution of inlier `libviso2` matches for corresponding image pairs from the UTIAS Multi-Season dataset, using each RGB-to-grayscale transformation. Orange lines indicate median values.

Table 5.5: Actual inlier feature matches using `libviso2` and each RGB-to-grayscale transformation on sequences from the UTIAS Multi-Season dataset. The highest mean number of matches for each sequence is highlighted in bold.

	Inlier Feature Matches $\mu(\sigma)$				
Test Sequence	Gray	SumLog	SumLog-E	MLP	MLP-E
`MultiSeason`					
`0008`	191 (56)	178 (52)	**201 (58)**	189 (56)	200 (60)
`0053`	18 (14)	14 (12)	25 (17)	22 (14)	**36 (16)**
`0108`	26 (17)	**40 (20)**	**40 (20)**	36 (19)	34 (17)
`0123`	10 (12)	3 (5)	**25 (17)**	3 (6)	19 (17)
`0144`	23 (20)	25 (18)	31 (20)	30 (21)	**36 (22)**
`0152`	41 (26)	**68 (29)**	67 (29)	66 (28)	62 (28)
`0177`	6 (8)	21 (15)	23 (17)	**26 (17)**	25 (17)
`0212`	2 (4)	10 (10)	12 (13)	10 (11)	**13 (12)**
`0224`	0 (0)	0 (1)	**1 (3)**	0 (1)	0 (1)
`0277`	5 (6)	10 (8)	10 (8)	10 (7)	**11 (8)**
`0328`	4 (6)	6 (7)	**13 (8)**	8 (8)	12 (8)
`0377`	15 (12)	27 (13)	30 (14)	**34 (14)**	29 (14)
`0428`	3 (6)	6 (8)	**8 (10)**	7 (9)	**8 (10)**
`0475`	6 (8)	14 (9)	**20 (9)**	16 (10)	18 (10)
`0538`	4 (7)	3 (6)	**8 (8)**	6 (8)	**8 (8)**
`0581`	3 (5)	1 (3)	**5 (7)**	2 (4)	**5 (7)**
`0683`	2 (5)	1 (3)	**3 (5)**	2 (4)	**3 (5)**

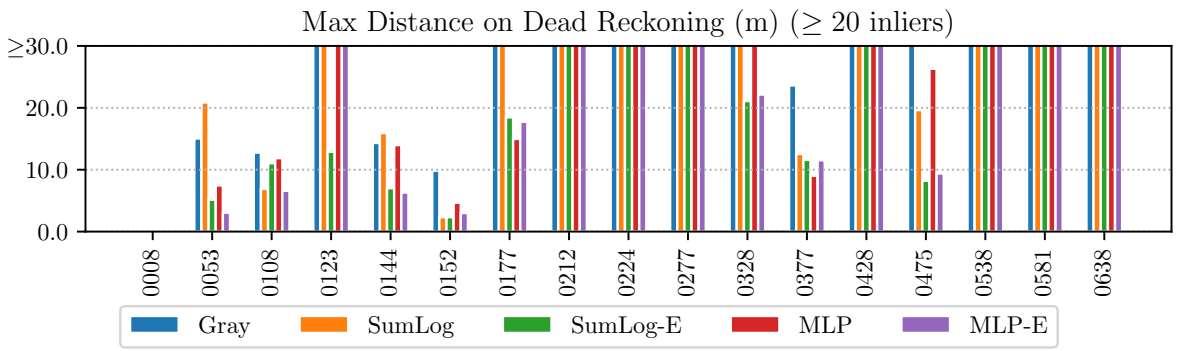


Figure 5.17: Maximum distances travelled on dead reckoning for each test sequence of the UTIAS Multi-Season dataset, based on a threshold of 20 inlier feature matches.

Localization Performance Figure 5.17 shows the maximum distances travelled using dead reckoning for each `MultiSeason` test sequence based on a threshold of 20 inlier feature matches against the "Map" sequence, while Table 5.6 summarizes these results numerically for thresholds of 10, 20, and 30 inlier matches. For a threshold of 20 inlier matches, Table 5.6 shows that the SumLog-E and MLP-E transformations significantly reduced the maximum distance on dead reckoning compared to the standard Gray transformation on all but two test sequences. However, manual interventions based on a threshold of 10 m on dead reckoning were avoidable on only 5 of 17 test sequences using the SumLog-E method and 6 of 17 test sequences using the MLP-E method. Consulting Table 5.6, we see that the situation is improved using a less conservative threshold of 10 inlier feature matches, where manual interventions were avoidable on 11 of 17 test sequences using the SumLog-E method and 12 of 17 test sequences using the MLP-E method.

As shown in Figure 5.17, performance dropped off sharply from sequence `MultiSeason/0177` onwards, with the exception of sequences `MultiSeason/0377` and `0475`. Returning to Figure 5.7, we see that the appearance of the environment begins to deviate substantially from that of the "Map" sequence on this portion of the dataset, but that sequences `MultiSeason/0377` and `0475` most closely resemble the initial appearance condition. Indeed, the standard Gray transformation also achieved better performance on these two sequences than on many others.

Figure 5.18 plots the empirical cumulative distribution functions (CDFs) of distances travelled on dead reckoning for each `MultiSeason` test sequence, based on a threshold of 20 inlier matches against the map. Compared to the results for daily/nightly appearance change shown in Figure 5.14, the results for seasonal appearance change are significantly murkier. However, we see that there is a clear benefit to the SumLog-E and MLP-E methods compared to the standard Gray transformation, as the CDF lines are more densely clustered in the top-left corner of their respective plots. In particular, we can see that localization failures in excess of 10 meters occur on less than 10% of the route for most sequences using the the SumLog-E and MLP-E methods, compared to up to 22% of the route using the Gray transformation.

Overall, these results demonstrate the significant impact of our technique on cross-seasonal localization performance using a single mapping experience, but also outline its limitations and highlight the continued need for long-term map management to achieve reliable long-term visual localization in unstructured environments over the course of months or years. While our method presents a viable

strategy for substantially reducing the data requirements of state-of-the-art experience-based localization pipelines, it is unlikely that any image pre-processing technique, however sophisticated, will be sufficient on its own to enable fully reliable 6-dof visual localization in the face of physical seasonal changes to unstructured operating environments. More complex image transformation models such as the encoder-decoder model discussed in Chapter 4 may generate output images that are visually similar, but are ultimately untrustworthy as they are liable to 'hallucinate' physical changes to the environment that may not result in improved localization. However, we conjecture that in more structured outdoor environments such as cities, the ubiquity of persistent structures whose appearance varies chiefly as a function of illumination suggests that image transformations such as those proposed in this chapter may be sufficient to enable reliable cross-seasonal localization using only vision and a single mapping experience.

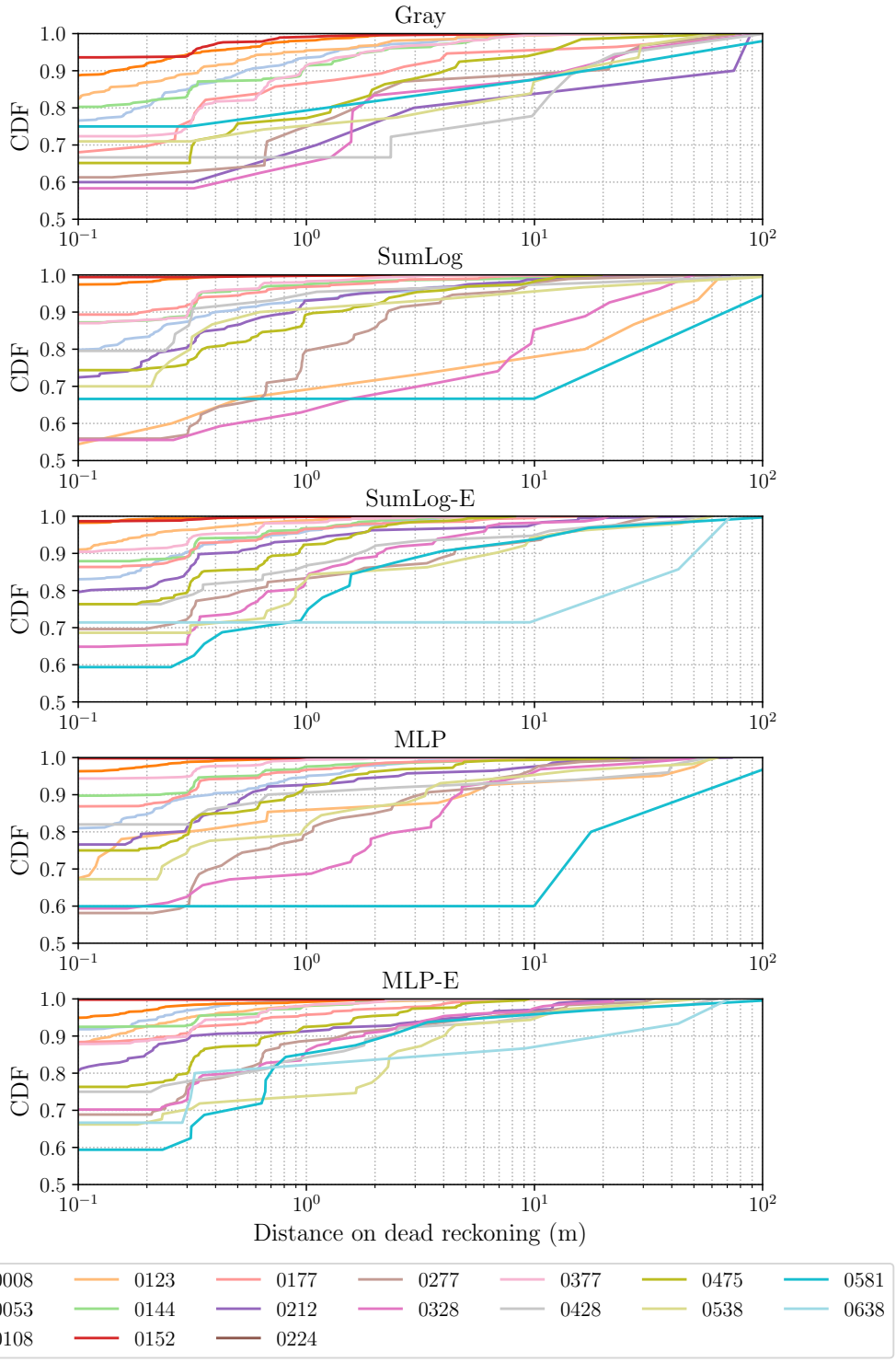


Figure 5.18: Empirical cumulative distribution functions of distances travelled on dead reckoning for each test sequence of the UTIAS Multi-Season dataset, based on a threshold of 20 inlier feature matches against the "Map" sequence.

Table 5.6: Maximum distances travelled on dead reckoning for each test sequence of the UTIAS Multi-Season dataset, based on various thresholds of inlier feature matches against the "Map" sequence. The best results are highlighted in bold.

	Maximum Distance on Dead Reckoning (m)														
	≥ 10 Inliers					**≥ 20 Inliers**					**≥ 30 Inliers**				
	G[1]	SL[2]	SL-E[2]	MLP	MLP-E	G[1]	SL[2]	SL-E[2]	MLP	MLP-E	G [1]	SL[2]	SL-E[2]	MLP	MLP-E
`MultiSeason`															
`0008`	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.5	**0.3**	**0.3**	**0.3**	0.4
`0053`	3.7	9.8	1.8	3.8	**0.2**	15.1	20.9	5.2	7.5	**3.1**	27.9	58.0	9.2	12.2	**6.2**
`0108`	2.3	0.9	0.7	**0.6**	1.1	12.8	6.9	11.1	11.9	**6.6**	27.9	**11.8**	13.4	12.2	15.6
`0123`	11.9	42.4	**4.9**	30.8	9.1	36.2	66.3	**12.9**	60.1	41.2	40.4	167.4	**26.3**	154.9	42.7
`0144`	5.9	3.0	**1.0**	5.7	1.3	14.3	15.9	7.0	14.0	**6.3**	38.5	39.8	**14.0**	23.6	**14.0**
`0152`	3.6	0.6	0.5	1.1	**0.2**	9.9	**2.3**	2.3	4.7	3.0	14.8	5.0	6.1	6.1	**5.5**
`0177`	21.0	7.3	12.1	6.1	**4.9**	76.2	44.4	18.5	**15.0**	17.7	80.9	69.6	**33.9**	40.8	46.0
`0212`	28.0	15.5	17.3	19.7	**10.0**	89.0	71.6	56.2	73.1	**30.3**	169.3	79.2	79.8	75.1	**73.9**
`0224`	169.5	141.2	**71.0**	169.5	96.3	**169.5**	**169.5**	**169.5**	**169.5**	**169.5**	**169.5**	**169.5**	**169.5**	**169.5**	**169.5**
`0277`	16.0	17.5	**6.6**	7.8	7.4	80.0	64.3	34.4	61.7	**33.2**	85.1	74.9	82.6	68.5	**63.1**
`0328`	45.2	44.0	**5.6**	43.3	9.1	99.0	49.8	**21.1**	55.0	22.1	125.8	124.0	93.9	59.5	**56.5**
`0377`	12.3	1.4	2.3	**1.2**	**1.2**	23.6	12.6	11.6	**9.0**	11.5	58.2	19.3	19.2	15.2	**14.2**
`0428`	56.7	14.7	**13.1**	25.1	13.7	102.3	140.1	**52.5**	57.7	63.0	141.2	140.4	126.7	**78.8**	79.1
`0475`	53.9	11.4	**1.6**	5.6	4.5	74.6	19.6	**8.2**	26.3	9.4	126.6	69.4	**20.3**	56.4	24.1
`0538`	42.3	49.2	**9.8**	41.4	16.4	61.4	140.0	60.0	63.7	**52.7**	140.2	140.0	132.7	**77.2**	131.3
`0581`	35.5	47.0	**19.3**	41.4	26.8	156.7	158.0	**121.6**	139.9	133.7	156.7	168.2	**141.2**	158.0	168.2
`0683`	**19.1**	51.4	21.9	20.1	42.2	125.9	125.9	72.7	125.9	**71.5**	**125.9**	**125.9**	**125.9**	**125.9**	**125.9**

[1] G: Gray
[2] SL: SumLog

5.5 Conclusions and Future Work

This chapter presented a method for learning RGB-to-grayscale colourspace mappings that, when used as a pre-processing step, explicitly maximize the number of inlier feature matches for a given input image pair, feature detector/matcher, and operating environment. Our key insight is that by training a deep neural network to predict the performance of a conventional non-differentiable feature detector/matcher, we can define a fully differentiable loss function that can be used to train image transformations that optimize a measure of localization performance. We evaluated our approach using both physically motivated and data-driven transformations and demonstrated substantially improved feature matching and localization performance on both synthetic and real-world long-term vision datasets that exhibit severe illumination change, potentially allowing experience-based localization to scale to long deployments with dramatically fewer bridging experiences. In particular, we found that continuous metric visual localization across day-night cycles is achievable using only a single mapping experience and a physically motivated weighted sum of log-responses with weights derived from a pairwise context encoder network trained to maximize the response of the feature detector/matcher proxy network. We further explored the impact of our proposed methods on improving localization performance across seasonal change and found that, while we did achieve major improvements in localization success over baseline methods, long-term map management remains necessary for coping with physical changes to unstructured operating environments over the course of months or years.

Future work might explore alternative loss functions such as pose estimation error relative to ground truth, the impact of context feature dimension on learned transformations, and the use of semantic information to identify and exploit persistent structures in image data while ignoring structures that are subject to physical changes. Another fruitful direction may be to investigate whether images optimized for feature matching could also be used to improve long-term direct (photometric) localization, or, alternatively, how our method might be adapted to accommodate a direct image alignment loss in the training process.

5.6 Novel Contributions

Our main contributions can be summarized as follows:

1. We proposed a technique for improving the robustness of a conventional visual localization pipeline to daily and seasonal appearance change by learning, in a self-supervised manner, a canonical appearance explicitly optimized for localization performance;

2. We developed a method for optimizing the performance of a non-differentiable localization pipeline by approximating the pipeline using a deep neural network;

3. We presented extensive experimental results on synthetic and real long-term vision datasets showing that our method enables continuous 6-dof metric localization across day-night cycles using a single mapping experience, and can reduce the number of bridging experiences needed to localize under seasonal appearance change.

4. We showed that our method compares favourably against a common empirically-tuned illumination-resistant image transformation; and

5. We released an open-source implementation of our method using PyTorch (Paszke et al., 2017), available at https://github.com/utiasSTARS/matchable-image-transforms.

5.7 Associated Publications

- Clement, L., Gridseth, M., Tomasi, J., and Kelly, J. (2019a). Learning maximally matchable image transformations for long-term metric visual localization. arXiv:1904.01080.
- Clement, L., Gridseth, M., Tomasi, J., and Kelly, J. (2019b). Matchable colorspace transformations for long-term metric visual localization. In *2019 CVPR Workshop on Image Matching*.

Chapter 6

Conclusion

6.1 Thesis Summary

In this thesis we have discussed several approaches to improving the accuracy and robustness of conventional visual localization pipelines by modelling aspects of environmental appearance and appearance change. These approaches were motivated by a desire to take full advantage of visual data, which encode a wealth of information about the environment and imaging sensor, most of which is discarded by traditional visual localization pipelines, but which can be exploited to improve the performance of vision-based robotic systems. We relied heavily on the increasingly prevalent tools of data-driven deep learning in the design of our systems, but took pains to avoid naïve approaches that replace well-understood conventional autonomy systems with black-box end-to-end learning. While learning-based methods are not a complete substitute for auxiliary sensors and long-term map management, our techniques are designed to be *complementary* to existing state-of-the-art methods in robotics, whether by extracting additional information from images or by curtailing the data and memory requirements of multi-experience localization frameworks. In this way we can make use of learning where appropriate, while preserving desirable properties of conventional systems such as interpretability and computational efficiency, and retaining the baseline performance of model-based algorithms. Through rigorous experimentation, we have demonstrated the effectiveness of using data-driven learning to augment the ability of existing autonomy systems to interpret real-world visual data, whether in the form of an auxiliary pseudo-sensor or a data pre-processing step, and have published several papers that we hope represent useful contributions towards robust long-term autonomous navigation.

6.2 Novel Contributions and Associated Publications

Chapter 3 presented a novel method for correcting orientation drift in visual odometry (VO) by estimating the direction of the sun from the same image stream used to compute VO. Our methods and experiments comparing hand-engineered visual sun detection algorithms and deep models on urban driving tasks appeared in the proceedings of two full-paper refereed conferences:

- Clement, L., Peretroukhin, V., and Kelly, J. (2017b). Improving the accuracy of stereo visual odometry using visual illumination estimation. In *Proceedings of the 2016 International Symposium*

on Experimental Robotics, Springer Proceedings in Advanced Robotics, pages 409–419, Tokyo, Japan. Springer International Publishing.

- Peretroukhin, V., Clement, L., and Kelly, J. (2017). Reducing drift in visual odometry by inferring sun direction using a bayesian convolutional neural network. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2035–2042, Singapore.
  Note: Peretroukhin and Clement contributed equally to this work and jointly claim first authorship.

These results were extended to planetary analogue environments, and a more detailed analysis was provided in a follow-up journal article:

- Peretroukhin, V., Clement, L., and Kelly, J. (2018). Inferring sun direction to improve visual odometry: A deep learning approach. *International Journal of Robotics Research*, 37(9):996–1016.
  Note: Peretroukhin and Clement contributed equally to this work and jointly claim first authorship.

The novel contributions of Chapter 3 can be summarized as follows:

1. We incorporated software-based visual sun detection as a 'pseudo-sensor' in a visual odometry pipeline, allowing for the extraction of global orientation information from the existing image stream and the correction of orientation drift in the motion estimate;
2. We demonstrated that Sun-BCNN, a Bayesian CNN with dropout layers after each convolutional and fully-connected layer, can achieve state-of-the-art accuracy on single-image sun detection at test time, and out-performs competing methods based on hand-engineered visual features;
3. We presented extensive experimental results on over 30 km of visual navigation data in urban and planetary analogue environments showing that indirect visual sun sensing significantly improves the accuracy of visual odometry over long trajectories;
4. We analyzed the sensitivity of Sun-BCNN to cloud cover, camera and environment changes, and measurement parameterization; and
5. We released an open-source implementation of Sun-BCNN, available at https://github.com/utiasSTARS/sun-bcnn.

Chapter 4 presented a novel method for improving the robustness of visual localization to environmental illumination change by training a deep convolutional encoder-decoder network to re-illuminate images such that they correspond to a previously-seen canonical appearance. Our methods and experiments demonstrating improved localization performance for localization pipelines based on direct photometric image alignment were published as a short journal paper:

- Clement, L. and Kelly, J. (2018). How to train a CAT: Learning canonical appearance transformations for direct visual localization under illumination change. *IEEE Robotics and Automation Letters*, 3(3):2447–2454.

The novel contributions of Chapter 4 can be summarized as follows:

1. We trained a convolutional encoder-decoder network to map images of a scene undergoing illumination change onto a previously-encountered canonical appearance, and incorporated the learned transformations as a pre-processing stage in a direct visual localization pipeline;

2. We demonstrated experimentally that our method yields significant improvements in visual odometry (VO) accuracy under time-varying illumination, as well as improved tracking performance in 6-dof metric localization under conditions of severe illumination change, where conventional methods fail;

3. We showed that our learning-based approach compares favourably against common illumination-robust analytical image transformations;

4. We investigated the possibility of transfer learning from synthetic to real environments in a localization context; and

5. We released an open-source implementation of our method, available at https://github.com/utiasSTARS/cat-net.

Chapter 5 presented a novel method for learning RGB-to-grayscale colourspace mappings that, when used as a pre-processing step, explicitly maximize the number of inlier feature matches for a given input image pair, feature detector/matcher, and operating environment. Our methods and experiments demonstrating improved localization performance over day-night cycles appeared at an abstract refereed workshop at a major international conference:

- Clement, L., Gridseth, M., Tomasi, J., and Kelly, J. (2019b). Matchable colorspace transformations for long-term metric visual localization. In *2019 CVPR Workshop on Image Matching*.

The novel contributions of Chapter 5 can be summarized as follows:

1. We proposed a technique for improving the robustness of a conventional visual localization pipeline to daily and seasonal appearance change by learning, in a self-supervised manner, a canonical appearance explicitly optimized for localization performance;

2. We developed a method for optimizing the performance of a non-differentiable localization pipeline by approximating the pipeline using a deep neural network;

3. We presented extensive experimental results on synthetic and real long-term vision datasets showing that our method enables continuous 6-dof metric localization across day-night cycles using a single mapping experience, and can reduce the number of bridging experiences needed to localize under seasonal appearance change.

4. We showed that our method compares favourably against a common empirically-tuned illumination-resistant image transformation; and

5. We released an open-source implementation of our method, available at https://github.com/utiasSTARS/matchable-image-transforms.

6.3 Outlook and Future Work

Vision has immense potential to be a major enabler of mobile robotic autonomy. As a multi-purpose, passive sensing modality, vision can simultaneously drive low-level localization and mapping tasks as well as high-level perception tasks, all at a fraction of the cost, weight, and power consumption of sophisticated active sensors such as 3D lidar. Future robotic vision systems will continue to benefit

from the marriage of performant model-based algorithms with data-driven learning that can assist in interpreting sensor data and account for situations where model-based approaches break down.

To enable data-efficient long-term autonomy over the course of months and years, new representations for maps and sensor data will be needed. These could take the form of learned feature descriptors, trained to express invariance to long-term appearance change, and some interesting recent work in this direction has already been presented by McManus et al. (2014), Carlevaris-Bianco and Eustice (2014), Krajnk et al. (2017), and Zhang et al. (2018). Alternatively, we could focus on learning image-to-image correspondences directly in the vein of Choy et al. (2016), Tang et al. (2018), and Melekhov et al. (2019). In either approach, the localization back-end is left untouched, but the front-end responsible for establishing visual correspondences is replaced with a learned model that relies on data-driven representations of visual appearance. The challenge of formulating such approaches is in generating appropriate training data, which often depends on establishing a large number of ground truth point correspondences across varying appearance conditions. Ideally, descriptor learning or direct correspondence learning should proceed in a self-supervised manner to avoid manually labelling training data, with the supervision signal deriving from geometric consistency or another easily verifiable validity test.

Digging deeper into the localization pipeline, future work might consider alternative representations of the operating environment and new ideas of what it means to build a map. Exciting recent work by Bloesch et al. (2018) and Brahmbhatt et al. (2018) points to the possibility of learning latent space representations for sensor data that can be combined into consistent maps and used for metric localization. By mapping images onto an appropriate latent space that encodes invariant properties of the environment (e.g., semantics), these methods could be adapted to enable metric localization across long-term appearance change while obviating the need to establish large numbers of point correspondences.

Finally, for learning-based vision systems to be suitable for the continuous operation of mobile robots outside the laboratory, the learning procedure should be formulated to enable online learning from sequential data as the robot continues to experience and interact with the operating environment. The problem of re-formulating deep learning to enable general-purpose continual learning remains an active area of research. Recent works including those of Sahoo et al. (2018) and Aljundi et al. (2019) present potentially fruitful approaches to this problem that could allow robotic systems to take advantage of new information as it arrives, without the need for costly and time-consuming offline optimization.

In summary, while this thesis presented several novel approaches to enhancing conventional vision-based localization pipelines using machine learning, reliable, robust, and data-efficient long-term visual localization remains far from a solved problem. The most exciting directions for future advances in this field lie at the intersection of computer vision, machine learning, and state estimation, as robotics researchers continue to innovate and apply new tools to old problems.

Bibliography

Agarwal, S., Mierle, K., et al. (2016). Ceres solver.

Alcantarilla, P. F. and Woodford, O. J. (2016). Noise models in feature-based stereo visual odometry. arXiv:1607.00273.

Aljundi, R., Kelchtermans, K., and Tuytelaars, T. (2019). Task-free continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11254–11263.

Anoosheh, A., Sattler, T., Timofte, R., Pollefeys, M., and Gool, L. V. (2019). Night-to-day image translation for retrieval-based localization. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5958–5964.

Baker, S. and Matthews, I. (2004). Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computuer Vision*, 56(3):221–255.

Barfoot, T. D. (2017). *State Estimation for Robotics*. Cambridge University Press.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (SURF). *Computuer Vision and Image Understanding*, 110(3):346–359.

Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., and Davison, A. J. (2018). CodeSLAM: Learning a compact, optimisable representation for dense visual SLAM. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2560–2568.

Brahmbhatt, S., Gu, J., Kim, K., Hays, J., and Kautz, J. (2018). Geometry-aware learning of maps for camera localization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2616–2625.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332.

Carlevaris-Bianco, N. and Eustice, R. M. (2014). Learning visual feature descriptors for dynamic lighting conditions. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2769–2776.

Carlevaris-Bianco, N., Ushani, A. K., and Eustice, R. M. (2015). University of Michigan North Campus long-term vision and lidar dataset. *International Journal of Robotics Research*, 35(9):1023–1035.

Cheng, Y., Maimone, M. W., and Matthies, L. (2006). Visual odometry on the mars exploration rovers. *IEEE Robotics and Automation Magazine*, 13(2):54–62.

Choi, Y., Choi, M., Kim, M., Ha, J., Kim, S., and Choo, J. (2018). StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. pages 8789–8797.

Choy, C. B., Gwak, J. Y., Savarese, S., and Chandraker, M. (2016). Universal correspondence network. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NeurIPS'16, pages 2414–2422, USA. Curran Associates Inc.

Churchill, W. and Newman, P. (2013). Experience-based navigation for long-term localisation. *International Journal of Robotics Research*, 32(14):1645–1661.

Clement, L., Gridseth, M., Tomasi, J., and Kelly, J. (2019a). Learning maximally matchable image transformations for long-term metric visual localization. arXiv:1904.01080.

Clement, L., Gridseth, M., Tomasi, J., and Kelly, J. (2019b). Matchable colorspace transformations for long-term metric visual localization. In *2019 CVPR Workshop on Image Matching*.

Clement, L. and Kelly, J. (2018). How to train a CAT: Learning canonical appearance transformations for direct visual localization under illumination change. *IEEE Robotics and Automation Letters*, 3(3):2447–2454.

Clement, L., Kelly, J., and Barfoot, T. D. (2016). Monocular visual teach and repeat aided by local ground planarity. In Wettergreen, D. S. and Barfoot, T. D., editors, *Proceedings of Field and Service Robotics*, Springer Tracts in Advanced Robotics, pages 547–561. Springer International Publishing, Toronto, Canada.

Clement, L., Kelly, J., and Barfoot, T. D. (2017a). Robust monocular visual teach and repeat aided by local ground planarity and color-constant imagery. *Journal of Field Robotics*, 34(1):74–97.

Clement, L., Peretroukhin, V., and Kelly, J. (2017b). Improving the accuracy of stereo visual odometry using visual illumination estimation. In *Proceedings of the 2016 International Symposium on Experimental Robotics*, Springer Proceedings in Advanced Robotics, pages 409–419, Tokyo, Japan. Springer International Publishing.

Corke, P., Paul, R., Churchill, W., and Newman, P. (2013). Dealing with shadows: Capturing intrinsic scene appearance for image-based outdoor localisation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2085–2092.

Costante, G., Mancini, M., Valigi, P., and Ciarfuglia, T. A. (2016). Exploring representation learning with CNNs for frame-to-frame ego-motion estimation. *IEEE Robotics and Automation Letters*, 1(1):18–25.

Cvišić, I. and Petrović, I. (2015). Stereo odometry based on careful feature selection and tracking. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–6.

Deng, J., Dong, W., Socher, R., Li, L., Kai Li, and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Eisenman, A. R., Liebe, C. C., and Perez, R. (2002). Sun sensing on the mars exploration rovers. In *Proceedings, IEEE Aerospace Conference*, volume 5, pages 5–5.

Engel, J., Koltun, V., and Cremers, D. (2018). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625.

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-Scale direct monocular SLAM. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, volume 8690 of *Lecture Notes in Computer Science*, pages 834–849. Springer International Publishing, Cham.

Engel, J., Stückler, J., and Cremers, D. (2015). Large-scale direct SLAM with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Fisher, R. (1953). Dispersion on a sphere. In *Proc. Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 217, pages 295–305. The Royal Society.

Fitzgibbon, A., Wexler, Y., and Zisserman, A. (2005). Image-based rendering using image-based priors. *International Journal of Compututer Vision*, 63(2):141–151.

Flynn, J., Neulander, I., Philbin, J., and Snavely, N. (2016). Deep stereo: Learning to predict new views from the world's imagery. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5515–5524.

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22.

Furgale, P. and Barfoot, T. D. (2010). Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560.

Furgale, P., Carle, P., Enright, J., and Barfoot, T. D. (2012). The Devon Island rover navigation dataset. *International Journal of Robotics Research*, 31(6):707–713.

Furgale, P., Enright, J., and Barfoot, T. (2011). Sun sensor navigation for planetary rovers: Theory and field testing. *IEEE Transactions on Aerospace and Electronic Systems*, 47(3):1631–1647.

Gaidon, A., Wang, Q., Cabon, Y., and Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4340–4349.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1050–1059.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 32(11):1231–1237.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361.

Geiger, A., Ziegler, J., and Stiller, C. (2011). StereoScan: Dense 3D reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968.

Gomez-Ojeda, R., Zhang, Z., Gonzalez-Jimenez, J., and Scaramuzza, D. (2018). Learning-based image enhancement for visual odometry in challenging HDR environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 805–811.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NeurIPS'14, pages 2672–2680, Cambridge, MA, USA. MIT Press.

Gridseth, M. and Barfoot, T. (2019). Towards direct localization for visual teach and repeat. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 97–104.

Grzeszczuk, R., Terzopoulos, D., and Hinton, G. (1998). Fast neural network emulation of dynamical systems for computer animation. In *Proceedings of the 11th International Conference on Neural Information Processing Systems*, NeurIPS'98, pages 882–888, Cambridge, MA, USA. MIT Press.

Haarnoja, T., Ajay, A., Levine, S., and Abbeel, P. (2016). Backprop KF: Learning discriminative deterministic state estimators. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 4383–4391, USA. Curran Associates Inc.

Hafner, D., Demetz, O., and Weickert, J. (2013). Why is the census transform good for robust optic flow computation? In Kuijper, A., Bredies, K., Pock, T., and Bischof, H., editors, *Scale Space and Variational Methods in Computer Vision*, pages 210–221, Berlin, Heidelberg. Springer Berlin Heidelberg.

Handa, A., Bloesch, M., Pătrăucean, V., Stent, S., McCormac, J., and Davison, A. (2016). gvnn: Neural network library for geometric computer vision. In *ECCV 2016 Workshops*, Lecture Notes in Computer Science, pages 67–82. Springer, Cham.

Handa, A., Whelan, T., McDonald, J., and Davison, A. J. (2014). A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1524–1531.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034.

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

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Huber, P., Wiley, J., and InterScience, W. (1981). *Robust statistics*. Wiley New York.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org.

Irani, M. and Anandan, P. (2000). About direct methods. In *Vision Algorithms: Theory and Practice*, pages 267–277. Springer Berlin Heidelberg.

Isola, P., Zhu, J., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, MM '14, pages 675–678, New York, NY, USA. ACM.

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision – ECCV 2016*, Lecture Notes in Computer Science, pages 694–711. Springer, Cham.

Kasper, M., Keivan, N., Sibley, G., and Heckman, C. (2016). Light source estimation in synthetic images. *ECCV 2016 Workshops*.

Kelly, J., Saripalli, S., and Sukhatme, G. S. (2008). Combined visual and inertial navigation for an unmanned aerial vehicle. In Laugier, C. and Siegwart, R., editors, *Field and Service Robotics: Results of the 6th International Conference*, pages 255–264. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kelly, J., Sibley, G., Barfoot, T., and Newman, P. (2012). Taking the long view: A report on two recent workshops on long-term autonomy [from the field]. *IEEE Robotics and Automation Magazine*, 19(1):109–111.

Kendall, A. and Cipolla, R. (2016). Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4762–4769.

Kendall, A., Grimes, M., and Cipolla, R. (2015). PoseNet: A convolutional network for real-time 6-dof camera relocalization. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2938–2946.

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

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE.

Koziel, S., Ciaurri, D. E., and Leifsson, L. (2011). *Surrogate-Based Methods*, pages 33–59. Springer Berlin Heidelberg, Berlin, Heidelberg.

Krajnk, T., Cristforis, P., Kusumam, K., Neubert, P., and Duckett, T. (2017). Image features for visual teach-and-repeat navigation in changing environments. *Robotics and Autonomous Systems*, 88(C):127–141.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NeurIPS'12, pages 1097–1105. Curran Associates Inc., USA.

Lalonde, J.-F., Efros, A. A., and Narasimhan, S. G. (2012). Estimating the natural illumination conditions from a single outdoor image. *International Journal of Computer Vision*, 98(2):123–145.

Lambert, A., Furgale, P., Barfoot, T. D., and Enright, J. (2012). Field testing of visual odometry aided by a sun sensor and inclinometer. *Journal of Field Robotics*, 29(3):426–444.

Latif, Y., Garg, R., Milford, M., and Reid, I. (2018). Addressing challenging place recognition tasks using generative adversarial networks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2349–2355.

LeCun, Y. (1989). Generalization and network design strategies. In Pfeifer, R., Schreter, Z., Fogelman, F., and Steels, L., editors, *Connectionism in perspective*. Elsevier.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Leshno, M. and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6:861–867.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. *International Journal of Robotics Research*, 34(3):314–334.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373.

Li, Q., Qian, J., Zhu, Z., Bao, X., Helwa, M. K., and Schoellig, A. P. (2017). Deep neural networks for improved, impromptu trajectory tracking of quadrotors. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5183–5189.

Linegar, C., Churchill, W., and Newman, P. (2015). Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 90–97.

Linegar, C., Churchill, W., and Newman, P. (2016). Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 787–794.

Liu, M.-Y., Breuel, T., and Kautz, J. (2017). Unsupervised image-to-image translation networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NeurIPS'17, pages 700–708. Curran Associates Inc., USA.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Compututer Vision*, 60(2):91–110.

Ma, W., Wang, S., Brubaker, M. A., Fidler, S., and Urtasun, R. (2017). Find your way by observing the sun and other semantic cues. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017). 1 year, 1000 km: The Oxford RobotCar dataset. *The International Journal of Robotics Research*, 36(1):3–15.

Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of visual odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, 24(3):169–186.

Matthies, L. H. (1989). *Dynamic Stereo Vision*. PhD thesis, Pittsburgh, PA, USA. AAI9023429.

McManus, C., Churchill, W., Maddern, W., Stewart, A. D., and Newman, P. (2014). Shady dealings: Robust, long-term visual localisation using illumination invariance. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 901–906.

McManus, C., Furgale, P., Stenning, B., and Barfoot, T. D. (2013). Lighting-invariant visual teach and repeat using appearance-based lidar. *Journal of Field Robotics*, 30(2):254–287.

McManus, C., Upcroft, B., and Newman, P. (2014). Scene signatures: Localised and point-less features for localisation. In *Robotics: Science and Systems X*.

McManus, C., Upcroft, B., and Newman, P. (2015). Learning place-dependant features for long-term vision-based localisation. *Autonomous Robots*, 39(3):363–387.

Melekhov, I., Tiulpin, A., Sattler, T., Pollefeys, M., Rahtu, E., and Kannala, J. (2019). DGC-Net: Dense geometric correspondence network. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1034–1042.

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

Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327.

Olson, C. F., Matthies, L. H., Schoppers, M., and Maimone, M. W. (2003). Rover navigation using stereo ego-motion. *Robotics and Autonomous Systems*, 43(4):215–229.

Omari, S., Bloesch, M., Gohl, P., and Siegwart, R. (2015). Dense visual-inertial navigation system for mobile robots. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2634–2640.

Park, S., Schöps, T., and Pollefeys, M. (2017). Illumination change robustness in direct visual SLAM. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4523–4530.

Pascoe, G., Maddern, W., and Newman, P. (2015). Robust direct visual localisation using normalised information distance. In *Procedings of the British Machine Vision Conference 2015*, pages 70.1–70.13. British Machine Vision Association.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*.

Paton, M. (2017). *Expanding the Limits of Vision-Based Autonomous Path Following*. PhD thesis, University of Toronto.

Paton, M., MacTavish, K., Berczi, L.-P., van Es, S. K., and Barfoot, T. D. (2018). I can see for miles and miles: An extended field test of visual teach and repeat 2.0. In *Proceedings of the Conference on Field and Service Robotics (FSR)*, pages 415–431.

Paton, M., MacTavish, K., Ostafew, C. J., and Barfoot, T. D. (2015). It's not easy seeing green: Lighting-resistant stereo visual teach & repeat using color-constant images. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1519–1526. IEEE.

Paton, M., MacTavish, K., Warren, M., and Barfoot, T. D. (2016). Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1918–1925.

Paton, M., Pomerleau, F., MacTavish, K., Ostafew, C. J., and Barfoot, T. D. (2017). Expanding the limits of vision-based localization for long-term route-following autonomy. *Journal of Field Robotics*, 34(1):98–122.

Peretroukhin, V., Clement, L., and Kelly, J. (2017). Reducing drift in visual odometry by inferring sun direction using a bayesian convolutional neural network. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2035–2042, Singapore.

Peretroukhin, V., Clement, L., and Kelly, J. (2018). Inferring sun direction to improve visual odometry: A deep learning approach. *International Journal of Robotics Research*, 37(9):996–1016.

Peretroukhin, V. and Kelly, J. (2018). DPC-Net: Deep pose correction for visual localization. *IEEE Robotics and Automation Letters*, 3(3):2424–2431.

Peretroukhin, V., Vega-Brown, W., Roy, N., and Kelly, J. (2016). PROBE-GK: Predictive robust estimation using generalized kernels. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 817–824.

Perez, R., Seals, R., and Michalsky, J. (1993). All-weather model for sky luminance distribution. *Solar Energy*, 50(3):235–245.

Peris, M., Martull, S., Maki, A., Ohkawa, Y., and Fukui, K. (2012). Towards a simulation driven stereo vision system. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1038–1042.

Pomerleau, D. A. (1989). ALVINN: An autonomous land vehicle in a neural network. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1*, pages 305–313. Morgan-Kaufmann.

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

Ratnasingam, S. and Collins, S. (2010). Study of the photodetector characteristics of a camera for color constancy in natural scenes. *Journal of the Optical Society of America A*, 27(2):286–294.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Sahoo, D., Pham, Q., Lu, J., and Hoi, S. C. H. (2018). Online deep learning: Learning deep neural networks on the fly. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2660–2666. International Joint Conferences on Artificial Intelligence Organization.

Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE Robotics and Automation Magazine*, 18(4):80–92.

Shum, H. and Kang, S. B. (2000). Review of image-based rendering techniques. In *Proceedings of SPIE 4067, Visual Communications and Image Processing*.

Skinner, J., Garg, S., Sünderhauf, N., Corke, P., Upcroft, B., and Milford, M. (2016). High-fidelity simulation for evaluating robotic vision performance. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2737–2744.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., and Corke, P. (2018). The limits and potentials of deep learning for robotics. *International Journal of Robotics Research*, 37(4-5):405–420.

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9.

Tang, J., Folkesson, J., and Jensfelt, P. (2018). Geometric correspondence network for camera motion estimation. *IEEE Robotics and Automation Letters*, 3(2):1010–1017.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30.

Torr, P. H. S. and Zisserman, A. (2000). Feature based methods for structure and motion estimation. In *Vision Algorithms: Theory and Practice*, pages 278–294. Springer Berlin Heidelberg.

Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380.

van der Merwe, R., Leen, T. K., Lu, Z., Frolov, S., and Baptista, A. M. (2007). Fast neural network surrogates for very high dimensional physics-based models in computational oceanography. *Neural Networks*, 20(4):462 – 478. Computational Intelligence in Earth and Environmental Sciences.

Wang, S., Clark, R., Wen, H., and Trigoni, N. (2018). End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research*, 37(4-5):513–542.

Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716.

Wolcott, R. W. and Eustice, R. M. (2014). Visual localization within lidar maps for automated urban driving. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–183.

Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In *Proceedings of the Third European Conference on Computer Vision (Vol. II)*, ECCV '94, pages 151–158, Berlin, Heidelberg. Springer-Verlag.

Zhang, J. and Singh, S. (2015). Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2174–2181.

Zhang, N., Warren, M., and Barfoot, T. D. (2018). Learning place-and-time-dependent binary descriptors for long-term visual localization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 828–835.

Zhang, Z., Forster, C., and Scaramuzza, D. (2017). Active exposure control for robust visual odometry in HDR environments. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3894–3901.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NeurIPS'14, pages 487–495, Cambridge, MA, USA. MIT Press.

Zhu, J., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251.