On learning pseudo-sensors to improve visual egomotion estimation

by

Valentin Peretroukhin

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Institute for Aerospace Studies
University of Toronto

Abstract

On learning pseudo-sensors to improve visual egomotion estimation

Valentin Peretroukhin
Doctor of Philosophy
Graduate Department of Institute for Aerospace Studies
University of Toronto
2019

The ability to estimate *egomotion* is at the heart of safe and reliable mobile autonomy. By inferring pose changes from sequential sensor measurements, egomotion estimation forms the basis of mapping and navigation pipelines, and permits mobile robots to self-localize within environments where external localization information may be intermittent or unavailable. Visual egomotion estimation, also known as *visual odometry*, has become ubiquitous in mobile robotics due to the availability of high-quality, compact, and inexpensive cameras that capture rich representations of the world. To remain computationally tractable, 'classical' visual odometry pipelines make simplifying assumptions that, while permitting reliable operation in ideal conditions, often lead to systematic error. In this dissertation, we present four ways in which conventional pipelines can be improved through the addition of a *pseudo-sensor*. In this context, a *pseudo-sensor* is a learned hyper-parametric model that infers latent information from sensor data. By fusing the output of such pseudo-sensors with traditional pipelines, we retain the performance of conventional techniques in nominal conditions while leveraging modern high-capacity data-driven models to improve uncertainty quantification, correct for systematic bias, and improve robustness to deleterious effects. We demonstrate the improvements derived from pseudo-sensors on data collected in sundry settings such as urban roads, indoor labs, and planetary analogue sites in the Canadian High Arctic.

Epigraph

A little learning is a dangerous thing; drink deep, or taste not the Pierian spring: there shallow draughts intoxicate the brain, and drinking largely sobers us again.

Alexander Pope

The universe is no narrow thing and the order within it is not constrained by any latitude in its conception to repeat what exists in one part in any other part. Even in this world more things exist without our knowledge than with it and the order in creation which you see is that which you have put there, like a string in a maze, so that you shall not lose your way. For existence has its own order and that no man's mind can compass, that mind itself being but a fact among others.

Cormac McCarthy

Elephants don't play chess.

Rodney Brooks

To all those who encouraged (or, at least, *never discouraged*) my intellectual wanderlust.

Acknowledgements

This document would not have been possible without the generous support and guidance of my supervisor[1], the perennial love of my family and friends[2], and the limitless patience of my lab mates[3]. Thank you all.

[1] as well as all of my collaborators and academic mentors (special thanks to Lee)
[2] especially the support and encouragement of Elyse
[3] in humouring my insatiable need for debate and banter (special thanks to Lee)

Contents

Notation

a	:	Symbols in this font are real scalars.
$\mathbf{a}$	:	Symbols in this font are real column vectors.
$\boldsymbol{a}$	:	Symbols in this font are real column vectors in homogeneous coordinates.
$\mathbf{A}$	:	Symbols in this font are real matrices.
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{R})$	:	Normally distributed with mean $\boldsymbol{\mu}$ and covariance $\mathbf{R}$.
$E[\cdot]$	:	The expectation operator.
$\underrightarrow{\boldsymbol{\mathcal{F}}}_a$	:	A reference frame in three dimensions.
$(\cdot)^\wedge$	:	An operator associated with the Lie algebra for rotations and poses. It produces a matrix from a column vector.
$(\cdot)^\vee$	:	The inverse operation of $(\cdot)^\wedge$.
$\mathbf{1}$	:	The identity matrix.
$\mathbf{0}$	:	The zero matrix.
$\mathbf{p}_a^{cb}$	:	A vector from point b to point c (denoted by the superscript) and expressed in $\underrightarrow{\boldsymbol{\mathcal{F}}}_a$ (denoted by the subscript).
$\mathbf{C}_{ab}$	:	The 3×3 rotation matrix that transforms vectors from $\underrightarrow{\boldsymbol{\mathcal{F}}}_b$ to $\underrightarrow{\boldsymbol{\mathcal{F}}}_a$: $\mathbf{p}_a^{cb} = \mathbf{C}_{ab}\mathbf{p}_b^{cb}$.
$\mathbf{T}_{ba}$	:	The 4×4 transformation matrix that transforms homogeneous points from $\underrightarrow{\boldsymbol{\mathcal{F}}}_a$ to $\underrightarrow{\boldsymbol{\mathcal{F}}}_b$: $\boldsymbol{p}_b^{cb} = \mathbf{T}_{ba}\boldsymbol{p}_a^{ca}$.

Chapter 1

Introduction

> What we call the beginning is often the end.
> And to make an end is to make a beginning.
>
> T.S. Eliot

This dissertation presents a general approach to improve visual egomotion estimation for mobile autonomous platforms. In the broadest sense, *egomotion estimation* refers to the process of computing the motion of a rigid body—relative to a fixed frame of reference—using measurements from sensors attached to the body (hence *ego*motion[1]).

Remark (*Autonomy through history*). Mobile *automata* have been a part of human culture since antiquity. In ancient India, the king Ajatashatru was said to use *bhuta vahana yanta* ('spirit movement machines') to protect the relics of Gautama Buddha after his death in the fourth century BCE. According to Burmese legend, the bhuta vahana yanta of Ajatashatru were made with stolen secrets from a group of Greco-Roman 'roboticists' named the *yantakara*. The methods of the yantakara were closely guarded, and mechanical assassins were said to pursue those who attempted to disseminate them (Mayor, 2019).

In the millennia since, mobile automata were relegated to isolated demonstrations (e.g., the programmable cart of Hero of Alexandria or the 'autonomous' knight of Leonardo da Vinci) or to imagined forms in cautionary tales (e.g., Mary Shelley's *Frankenstein*). Although the secrets of the yantakara may never be rediscovered, the ancient pursuit of helpful automata has found new life towards the turn of the twenty-first century and yielded machinery and algorithms that show great promise in aiding humanity.

[1] To the best of the author's knowledge, this term originates from experimental psychology (Warren, 1976), and was also referred to as *passive navigation* in the context of camera motion (Bruss and Horn, 1983).

One way to estimate egomotion is to measure rates (e.g., linear or rotational velocities) and integrate them to compute changes in position and orientation. This method, referred to as *dead reckoning*, was used by marine navigators of ancient times to determine the longitude of a ship at sea. Unlike latitude, which navigators could determine by measuring the altitude of the noon Sun, or the altitude of Polaris (the North Star) at night, longitude was not reliably computed from celestial measurements until the development of the marine chronometer in the 18th century (Barfoot, 2017). As a result, early seafarers could only *dead reckon* east-west motion by relying on estimates of local water currents, and by measuring the ship's magnetic heading and its speed relative to water (through a tool called a *chip log*, Figure 1.1).

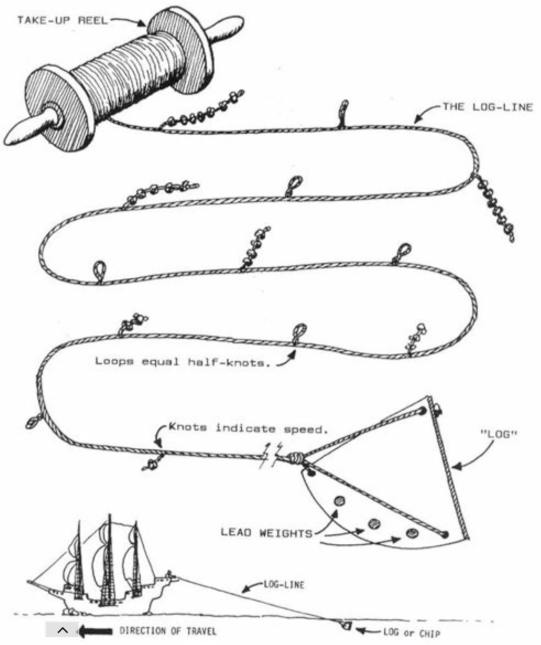


Figure 1.1: A *chip log* is a historical tool used to measure ship speed. The *log* or *chip* was tossed into the water, and speed was measured by the amount of *knots* that unravelled in a set time interval (*credit: oceonmotion.org*).

In a similar process, early aviators computed egomotion through magnetic heading, airspeed, and an estimate of prevailing winds. While even modern egomotion estimation methods exhibit unbounded error growth (Olson et al., 2003), these early techniques were particularly inaccurate and required regular corrections through observations of known landmarks.[2] In the mid twentieth century, the goals of inter-continental flight and space exploration necessitated the development of more accurate egomotion sensors (e.g., gimballed inertial platforms including accelerometers and gyroscopes) and an associated set of estimation techniques that could compute egomotion without human intervention (e.g., the Kalman filter (Grewal and Andrews, 2010)).

By the late twentieth century, unmanned Lunar and Martian exploration motivated the development of a new approach to egomotion estimation. Although ground vehicles like extra-planetary rovers could infer egomotion through *wheel odometry* (integrating wheel rates and using wheel orientation to drive a kinematic model), this approach was highly inaccurate on surfaces that induce wheel slip (e.g., the rock and sand covered surface of Mars, Figure 1.3).

To address this, a number of researchers in the 1980s developed the technique of *visual* odometry (Scaramuzza and Fraundorfer, 2011) (or VO), as a way to infer egomotion from sequentially-collected images. The mathematical basis of VO is closely tied to the technique of photogrammetric *bundle adjustment* (Triggs et al., 2000) which originates from 19th century photogrammetry (Albertz, 2007).

[2] Perhaps the most famous example of dead-reckoning error was made by Christopher Columbus in 1492 when he arrived in modern-day Bahamas, but believed he had reached the *Indies* (modern Indonesia).

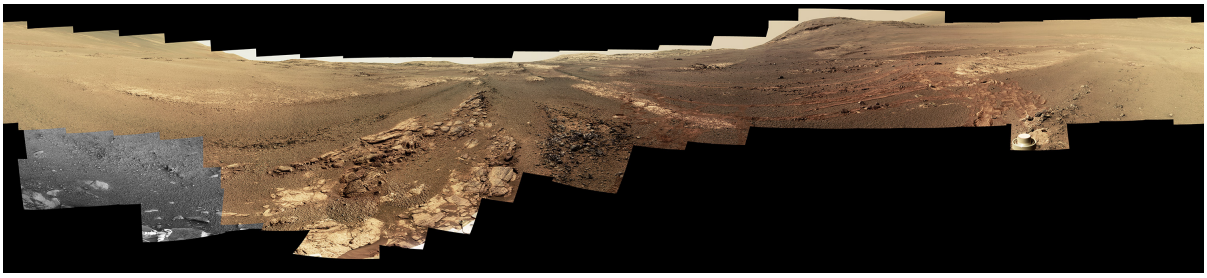


Figure 1.3: The last 360 degree panorama of the rocky Martian surface taken by the Pancam apparatus of the Mars Exploration Rover, *Opportunity*, at its final resting place, the western rim of the Endeavour Crater (*credit: NASA/JPL-Caltech/Cornell/ASU*).

In the nearly four decades since the first development of VO, the proliferation of compact, relatively-inexpensive, high-resolution cameras has made vision-based sensing techniques ubiquitous in mobile autonomous applications. In addition to computing egomotion, camera data can be used to build detailed maps of an environment (which can be computed in tandem with egomotion through a technique called simultaneous localization and mapping, or SLAM (Cadena et al., 2016)), as well as detect, track and avoid other objects (Figure 1.2).

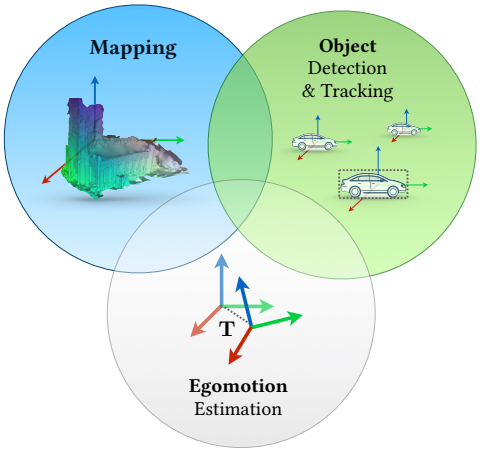


Figure 1.2: Visual egomotion estimation can be useful to create maps and to detect and track other objects.

1.1 A Visual *Pipeline*

Central to *classical* visual odometry algorithms (which, in this context, refers to the bulk of VO research published during what Cadena et al. (2016) call the *classical* and *algorithmic-analysis* ages of VO and SLAM research between 1986 and 2015) is the idea of a processing pipeline. A pipeline consists of several connected computational 'blocks' or 'stages' that have interpretable inputs and outputs. By carefully processing information contained within raw sensor data, pipelines facilitate the construction of complex state estimation architectures that can fuse visual observations with other sensors of varied modality to create maps and models of the external world and infer the egomotion of a mobile platform within it. In this dissertation, we will largely deal with the improvement of a canonical visual odometry pipeline—we illustrate its major components in Figure 1.4.

Broadly, VO solutions based on the idea of hand-crafted pipelines (Leutenegger et al., 2015; Cvišić and Petrović, 2015; Tsotsos et al., 2015; Alcantarilla and Woodford, 2016; Forster et al., 2014; Wang et al., 2017a; Engel et al., 2018) have achieved impressive localization accuracy within a variety of settings. They have been used to compute the egomotion of self-driving cars through urban roads, and track autonomous drones through aggressive flying maneuvers.

Building on momentum from the computer vision community, a significant part of the visual

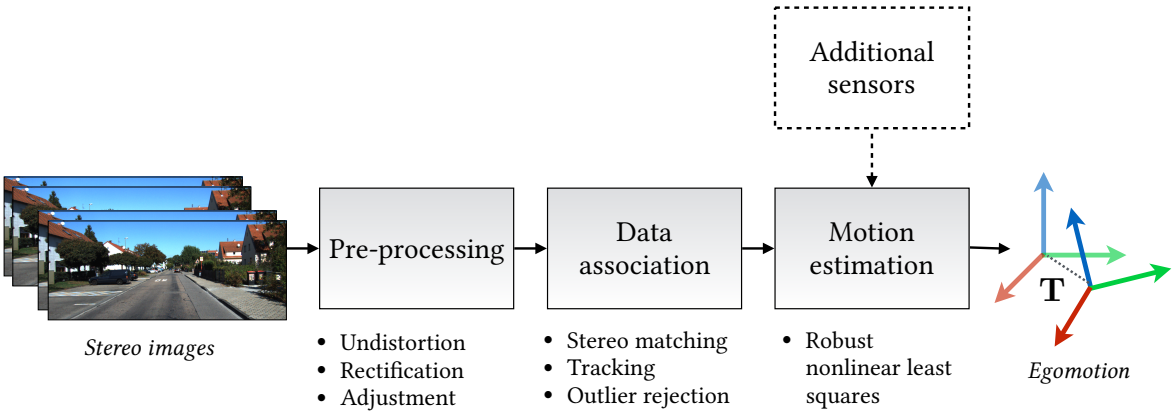


Figure 1.4: A 'classical' visual odometry pipeline consists of several distinct components with interpretable inputs and outputs. We list common examples of each component.

state estimation literature has also considered replacing classical pipelines with parametric modelling through deep convolutional neural networks (CNNs) and data-driven learning. Although initially developed for image classification (LeCun et al., 2015), CNN-based measurement models have been applied to numerous problems in visual state estimation including homography estimation (DeTone et al., 2016), single image depth reconstruction (Garg et al., 2016), camera re-localization (Kendall and Cipolla, 2016), and place recognition (Sünderhauf et al., 2015). A number of CNN-based approaches have also tackled the problem of visual egomotion estimation, often purporting to obviate the need for classical visual localization pipelines by learning pose changes *end-to-end*, without requiring intermediate outputs (Melekhov et al., 2017; Handa et al., 2016; Oliveira et al., 2017).

In light of these new approaches, debate has emerged within the robotics and computer vision communities regarding the extent to which data-driven parametric models should replace pipelines. Although classical approaches can achieve high accuracy under nominal conditions, deep data-driven networks have the potential to improve upon them in two respects.

First, owing to their representational power, deep parametric networks have the potential to learn bespoke data associations that are robust to (1) large viewpoint changes, (2) moving objects, and (3) self-similar visual textures (e.g., indoor walls, grass, sky, etc.). Such robust, invariant data associations have been the holy-grail of hand-crafted pipelines, and traditional data association methods must be carefully tuned to work well in a given setting (Schonberger et al., 2017).

Second, deep parametric regression of egomotion has the potential to more efficiently use dense high-dimensional visual data. To remain computationally tractable, classical VO pipelines typically take one of two approaches. First, some pipelines (Leutenegger et al., 2015; Cvišić and Petrović, 2015) choose to indirectly summarize visual data by extracting and matching a set of sparsely-distributed salient *features*. These features may be robust to some of the effects mentioned above, but, by construction, they discard large portions of visual data that may inform more accurate egomotion estimates. Alternatively, other methods (Forster et al., 2014; Wang et al., 2017a; Engel et al., 2018) may instead choose to match photometric intensity directly through an assumption of photometric consistency.

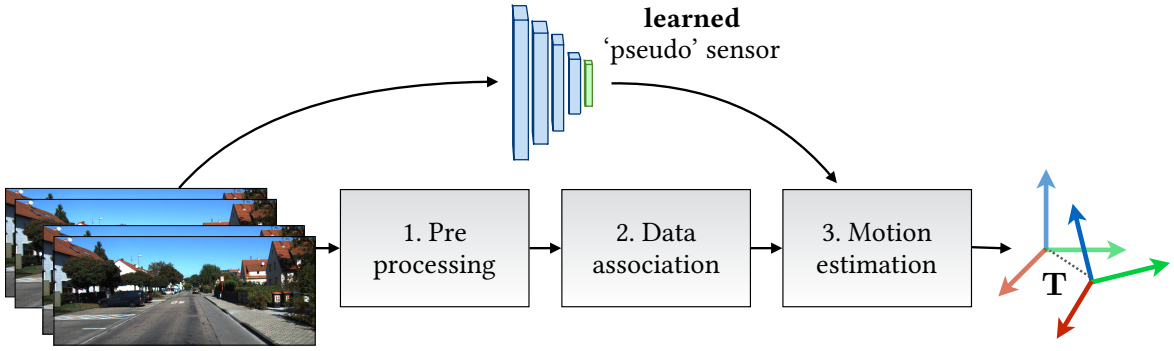


Figure 1.5: A learned *pseudo-sensor* extracts latent information from the same data stream.

This relatively simple schema permits dense data association but can be a poor assumption in the presence of large viewpoint changes or non-Lambertian surfaces. Further, this approach yields a high number of data associations and produces a highly non-convex objective that requires care to optimize.

In contrast, deep networks built on basic image operations with modern computationally-efficient implementations have the potential to avoid the pitfalls of both of these approaches while remaining tractable (assuming appropriate hardware). Despite this potential, current end-to-end learning techniques for egomotion estimation have a number of disadvantages (see Table 1.1). They often generalize poorly to new environments, come with few analytical guarantees, provide only point estimates of latent parameters, and do not allow for intermediate representations that have been shown to improve generalization performance on visual tasks (Zhou et al., 2019). Indeed, according to one benchmark, the most accurate visual egomotion approach at the time of writing[3] remains a classical pipeline based on carefully selected sparse features.

1.2 The Pseudo-Sensor

As mobile autonomy enters the robust-perception age (Cadena et al., 2016), classical pipelines that work in limited contexts will need to be adapted and augmented to ensure they can operate over longer time periods, and through challenging environments. However, replacing these performant pipelines completely with learned approaches, is in our view, unnecessary. Instead, we propose the paradigm of the *pseudo-sensor* (Figure 1.5). A *pseudo-sensor* is a machine-learning-based (hyper-)parametric model that extracts latent information from existing sensor data that is difficult to model analytically. Instead of replacing an entire pipeline, pseudo-sensors leverage the representational power of modern data-driven learning techniques to extract useful quantities that make existing classical pipelines more consistent and accurate in a given environment. In this dissertation, we present four such pseudo-sensors. In each case, the output of the pseudo-sensor is fused with a classical visual odometry pipeline to produce better motion estimates. To accomplish this fusion, we rely on two approaches. The first

[3]Based on the KITTI Odometry benchmark leaderboard at http://www.cvlibs.net/datasets/kitti/eval_odometry.php.

Table 1.1: A comaprison of pipelines and end-to-end deep models for visual egomotion estimation.

	Classical Pipelines	**Deep Models**
Maturity	Decades of literature & domain knowledge	Nascent with few uses in mobile autonomy
Interpretability	Good, each component has interpretable input and output	Poor, often with no interpretable intermediate outputs
Uncertainty	Foundational to *probabilistic robotics*	Few nascent methods (Monte-carlo Dropout (Gal and Ghahramani, 2016b), Bootstrap (Osband et al., 2016))
Robustness	Empirically generalizable (Zhou et al., 2019)	Highly dependant on training data
Flexibility	Limited by ingenuity of designer	Limited by training data

(Predictive Robust Estimation, or PROBE, Chapter 4), uses the paradigm of a pseudo-sensor to infer a heteroscedastic noise model based on sensor data. By predicting uncertainty information, PROBE effectively re-scales a robust loss function to better account non-stationary noise models and deleterious visual effects. The second approach (used by Sun-BCNN, DPC-Net, and HydraNet, Chapters 5 to 7 respectively) produces geometric quantities (probabilistic estimates of an illumination source, $\mathrm{SE}(3)$ corrections to existing egomotion estimates, and independent probabilistic rotation estimates, respectively), that can be fused with the egomotion estimate produced by the pipeline through pose graph optimization.

1.3 Original Contributions

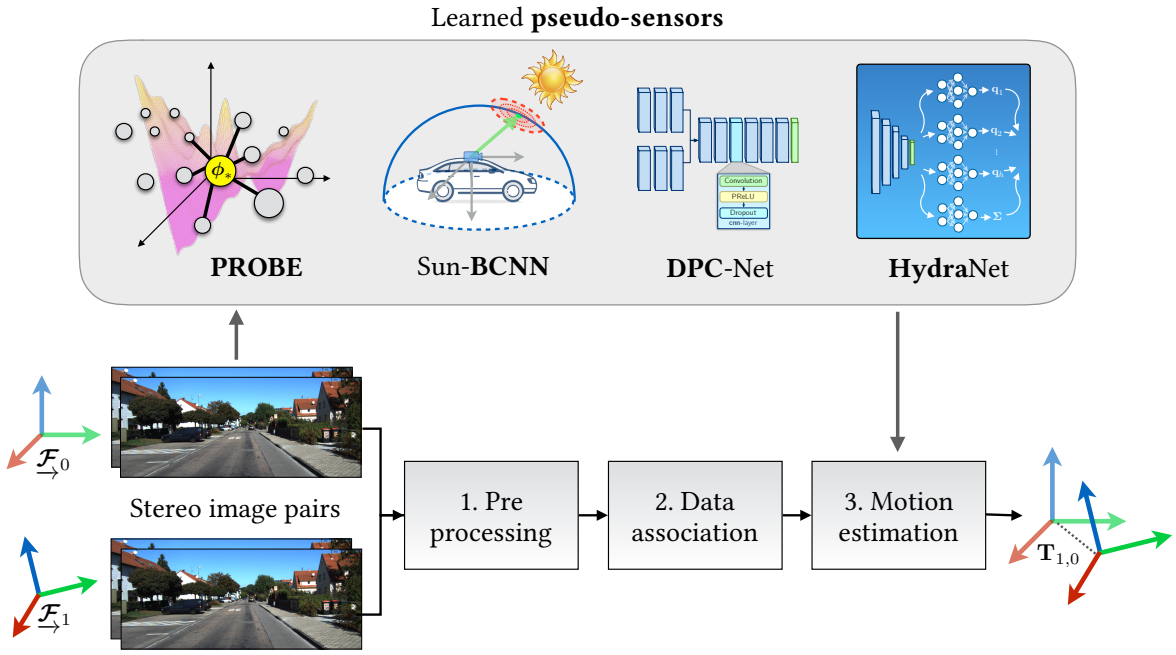


Figure 1.6: Four *pseudo-sensors* that improve 'classical' egomotion estimation through data-driven learning.

This dissertation consists of several published contributions under the umbrella of *learned pseudo-sensors* that improve a canonical visual egomotion pipeline (Figure 1.6). Before detailing each pseudo-sensor, we present some mathematical foundations (Chapter 2) and a common baseline for a feature-based stereo visual odometry pipeline (Chapter 3) which all four methods build upon. In total, there are two journal papers and five conference papers associated with our work. Below, we briefly summarize each of the pseudo-sensors and list the publications that are associated with each.

1. **PROBE: Predictive Robust Estimation**,
   Predictive Robust Estimation (Chapter 4, Appendix A) builds a predictive model for observation uncertainty from training data. To do this, we collaborate with William Vega-Brown at MIT to adapt the technique of Generalized Kernel (GK) estimation (Vega-Brown et al., 2014) to visual odometry. Generalized Kernel estimation allows us to build an efficient Bayesian model for stereo tracking uncertainty. By setting a prior on covariance, we derive a 'robust' objective that can be *predictively* scaled to improve the accuracy and consistency of a feature-based stereo visual odometry pipeline. PROBE is associated with three publications: Peretroukhin et al. (2015a,b, 2016). The first two publications explore useful predictors for uncertainty and build a non-Bayesian isotropic covariance model. The latter publication presents the Bayesian GK approach.

2. **Sun-BCNN: Learned Probabilistic Sun Sensor**
   Sun-BCNN (Chapter 5) is a virtual sun sensor based on a Bayesian Convolutional Neural Net-

work (BCNN) that was developed in collaboration with Lee Clement. Much like a dedicated hardware sun sensor, Sun-BCNN infers a probabilistic estimate of the direction of the sun that can be used to inject orientation information into an egomotion pipeline. However, unlike dedicated sensors, Sun-BCNN requires no additional hardware and can predict both a mean and uncertainty from a single RGB image. It is associated with three publications: Clement et al. (2017); Peretroukhin et al. (2017, 2018). The first publication consists of initial exploratory work on virtual sun sensors, while the second presents the BCNN formulation. The final publication is an extended journal article that summarizes the results of the prior two conference publications and adds two novel contributions: (1) further experimental validation on visual data collected in the Canadian High Arctic and around Oxford, UK, and (2) investigations into the effect of cloud cover and the possibility of generalization across datasets.

3. **DPC-Net: Learned Pose Corrections**

   Deep Pose Correction (Chapter 6) is an approach to improving egomotion estimates through $\mathrm{SE}(3)$ pose corrections learned with deep networks. As part of this work, we derive a novel loss function based on Lie theory that permits the learning six degree-of-freedom pose residuals in a supervised learning framework. After training, our Deep Pose Correction Network (DPC-Net) predicts low-rate, 'small' *corrections* that can be fused with egomotion estimates from a canonical pipeline. DPC-Net does not require any modification to an exisiting pipeline, and can learn to correct multi-faceted errors from estimator bias, sensor mis-calibration or environmental effects. It is associated with one journal publication, Peretroukhin and Kelly (2018).

4. **HydraNet: Learned Probabilistic Rotation Estimation**

   Finally, HydraNet (Chapter 7) is a multi-headed network structure that can regress probabilistic estimates of rotation (elements of the matrix Lie group, $\mathrm{SO}(3)$) that account for both aleatoric and epistemic uncertainty. HydraNet builds upon results from both Sun-BCNN and DPC that show that correcting rotation is critical to accurate egomotion estimation. Towards this end, HydraNet is designed to produce well-calibrated notions of uncertainty over $\mathrm{SO}(3)$ that facilitate fusion with classical egomotion pipelines through a probabilistic factor graph formulation. It is associated with the publication, Peretroukhin et al. (2019).

Chapter 2

Mathematical Foundations

> By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems, and, in effect, increases the mental power of the race.
>
> — Alfred North Whitehead

2.1 Coordinate Frames

Before we can present the main contributions of this dissertation, it will be useful to first outline the notation and mathematical foundations that underly the work. Throughout this dissertation, we largely follow the notation of Barfoot (2017) when dealing with three-dimensional rigid-body kinematics.

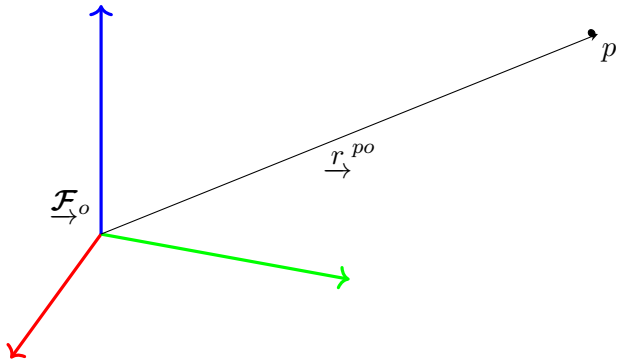


Figure 2.1: A position vector expressed in a coordinate frame.

We refer to a three-dimensional position vector, $\underrightarrow{r}^{po}$, as one that originates at the origin of a coordinate reference frame, $\underrightarrow{\mathcal{F}}_o$, and terminates at the point p. This geometric quantity has the numerical coordinates $\mathbf{r}_o^{po}$ when expressed in $\underrightarrow{\mathcal{F}}_o$. Often, we will refer to two reference frames such as a world or *inertial* frame, $\underrightarrow{\mathcal{F}}_i$, and a vehicle frame, $\underrightarrow{\mathcal{F}}_v$. Rotation matrices or rigid-body transformations that

convert coordinates from $\underrightarrow{\mathcal{F}}_i$ to $\underrightarrow{\mathcal{F}}_v$ will be represented as $\mathbf{T}_{vi}$, and $\mathbf{C}_{vi}$[1], respectively.

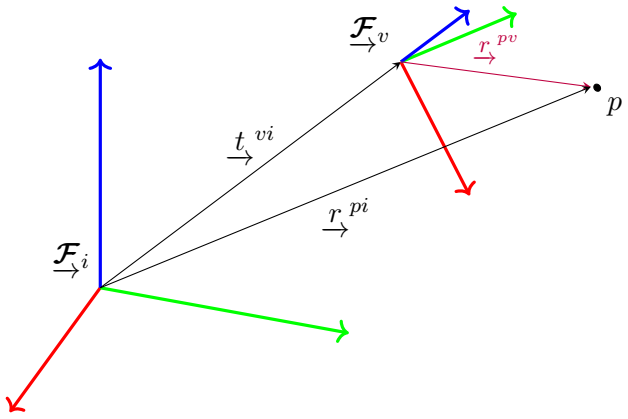


Figure 2.2: Two common references frames used throughout this thesis.

2.2 Rotations

The rotation matrix $\mathbf{C}$ is a member of the matrix Lie group $\mathrm{SO}(3)$ (the Special Orthogonal group). We can define it as follows:

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

Remark (*Matrix Lie Groups*). A group is a set with an operation (which combines two elements to form a third element that is part of the group) that satisfies the four group axioms: closure, associativity, identity and invertibility. A Lie group is a group that is also a differentiable manifold. Finally, a *matrix* Lie group is a group whose elements are matrices and whose operation is matrix multiplication. See Barfoot (2017); Solà et al. (2018) for more details.

2.2.1 Axis-Angle Parameters

Any rotation can be represented (non-uniquely) as a rotation of angle ϕ about an axis $\mathbf{a}$ (Hartley et al., 2013). Concretely, we can map an axis-angle vector, $\boldsymbol{\phi} = \phi\mathbf{a}, \quad \phi \in \mathbb{R}, \mathbf{a} \in S^2$, to a rotation matrix, $\mathbf{C}$, using the surjective exponential map

$$\mathbf{C} = \mathrm{Exp}\left(\boldsymbol{\phi}\right) = \exp\left(\boldsymbol{\phi}^\wedge\right) = \sum_{n=0}^{\infty} \frac{1}{n!}(\boldsymbol{\phi}^\wedge)^n \tag{2.2}$$

$$= \cos\phi\mathbf{1} + (1-\cos\phi)\mathbf{a}\mathbf{a}^T + \sin\phi\mathbf{a}^\wedge, \tag{2.3}$$

[1] We use $\mathbf{C}$ and not $\mathbf{R}$ for rotation matrices to avoid confusion with common notation for measurement model covariance.

where the wedge operator $(\cdot)^{\wedge}$[2] is defined as

$$\mathbf{a}^{\wedge} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}^{\wedge} = \begin{bmatrix} 0 & -a_2 & a_1 \\ a_2 & 0 & -a_0 \\ -a_1 & a_0 & 0 \end{bmatrix}. \tag{2.4}$$

The vector $\boldsymbol{\phi}$ is sometimes called the *rotation vector* (Barfoot, 2017) or referred to as the *exponential coordinates* (Gallego and Yezzi, 2015) of rotation. Equation (2.3) is often referred to as the Euler-Rodriguez formula. Although the map in Equation (2.2) is surjective, we can define an inverse map if we restrict its domain to $0 \leq \phi < \pi$:

$$\boldsymbol{\phi} = \operatorname{Log}(\mathbf{C}) = \log(\mathbf{C})^{\vee} = \frac{\phi(\mathbf{C} - \mathbf{C}^T)^{\vee}}{2\sin\phi}, \tag{2.5}$$

where $\phi = \arccos\left(\frac{\operatorname{tr}(\mathbf{C})-1}{2}\right)$ and the *vee* operator, $(\cdot)^{\vee} : \mathbb{R}^{3\times 3} \to \mathbb{R}^3$, is defined as the unique inverse of the wedge operator $(\cdot)^{\wedge}$. Note Equation (2.5) is undefined at both $\phi = 0$ and at $\phi = \pi$. In the former case, we can use a small-angle approximation and define

$$\operatorname{Log}(\mathbf{C}) \approx (\mathbf{C} - \mathbf{1})^{\vee} \text{ when } \phi \approx 0. \tag{2.6}$$

The latter case (when $\phi = \pi$) defines the *cut locus* of the space where $\operatorname{Exp}(\cdot)$ is not a covering map and both $+\boldsymbol{\phi}$ and $-\boldsymbol{\phi}$ map to the same $\mathbf{C}$. This is a fundamental drawback of using three parameters to represent rotation.

2.2.2 Unit Quaternions

We can also represent a rotation with unit quaternion, $\mathbf{q}$. A unit quaternion consists of a scalar value q_ω, and a three-dimensional vector component, $\mathbf{q}_v$:

$$\mathbf{q} = \begin{bmatrix} q_\omega \\ \mathbf{q}_v \end{bmatrix} \in S^3, \quad (\|\mathbf{q}\| = 1). \tag{2.7}$$

Unit quaternions also form a Lie group (Solà et al., 2018) (but not a matrix Lie group) and and lie on a three-dimensional unit sphere within $\mathbb{R}^4$. We will overload our exponential map operator to use the same notation to define a surjective map from three parameters to unit quaternions,

$$\mathbf{q} = \operatorname{Exp}(\boldsymbol{\phi}) = \begin{bmatrix} \cos(\phi/2) \\ \mathbf{a}\sin(\phi/2) \end{bmatrix}. \tag{2.8}$$

By inspection, we can see that both $\mathbf{q}$ and $-\mathbf{q}$ represent the same axis-angle pair, $\{\phi, \mathbf{a}\}$. As a result, unit quaternions represent a *double cover* of $\mathrm{SO}(3)$ and we must be careful to account for this when using them as a rotation parametrization. In particular, when computing the (also overloaded)

[2] This operator is sometimes also expressed as $(\cdot)^{\times}$ or $[\cdot]_{\times}$ and is known as the skew-symmetric operator.

logarithmic map,

$$\boldsymbol{\phi} = \operatorname{Log}(\mathbf{q}) = 2\mathbf{q}_v \frac{\arctan\left(\|\mathbf{q}_v\|, q_\omega\right)}{\|\mathbf{q}_v\|}, \tag{2.9}$$

we can must account for the double cover by replacing $\mathbf{q}$ with $-\mathbf{q}$ if q_ω is negative. Also note that as with rotation matrices Equation (2.9) is undefined when $\phi = 0$, but, importantly, we do not face any issues when $\phi = \pi$ due to the half-angle. In the former case, we can again rely on small angle approximations to define:

$$\operatorname{Log}(\mathbf{q}) \approx \frac{\mathbf{q}_v}{q_\omega}\left(1 - \frac{\|\mathbf{q}_v\|^2}{3q_\omega^2}\right) \quad \text{when} \quad \phi \approx 0. \tag{2.10}$$

A fantastic summary of the history of rotation parameterizations, unit quaternions and the story of Hamilton and Rodriguez can be found in Altmann (1989).

2.2.3 Topology and Parameterizations

Topologically, $\mathrm{SO}(3)$ is *diffeomorphic*[3] to the real projective space, $\mathbb{RP}^3$, the space of all lines passing through the origin in $\mathbb{R}^4$ (Hartley et al., 2013). As a result, any global n-parametrization of $\mathrm{SO}(3)$ will incur some cost. If we use rotation matrices ($n = 9$), we need to ensure orthonormality and that $\det \mathbf{C} = 1$. With unit quaternions ($n = 4$), we need to account for the unit-norm constraint and take note of the double cover. Parameterizations with $n = 3$ (like axis-angle parameters or Euler angles) will be bounded, but unconstrained. However, due to the topological structure of $\mathrm{SO}(3)$ all three-parameter parameterizations will not be invertible for certain rotations. With Euler angles, one has to be wary of *gimbal lock*, wherein two angles become indeterminate from each other. For axis-angle parameters, it is not possible to uniquely represent rotations whose angle is π.

Remark (*SO(3) Topology*). To see why this is the case, consider that a direct result of Euler's rotation theorem is that we can represent any element in $\mathrm{SO}(3)$ by a (non-unique) axis-angle pair $\{\mathbf{a}, \phi\}$, $\mathbf{a} \in S^2$, and $0 \leq \phi \leq \pi$. We can partially represent this space by the closed ball of radius π in $\mathbb{R}^3$ using the combined axis-angle coordinates $\boldsymbol{\phi} = \phi\mathbf{a}$. However, at the boundary ($\phi = \pi$), we must account for the fact that rotations represented by $\{\mathbf{a}, \pi\}$ are identical to those represented by $\{-\mathbf{a}, \pi\}$. In other words, we must *identify* all antipodal points, $\boldsymbol{\phi}$ and $-\boldsymbol{\phi}$ when $\|\boldsymbol{\phi}\| = \pi$. This closed 3-ball with identified antipodal points on its boundary is topologically equivalent to the 3-sphere (S^3) with its antipodal points identified. In turn, this space is equivalent to $\mathbb{RP}^3$ since any line passing through the origin in $\mathbb{R}^4$ can be mapped to two unit normals, $\pm\mathbf{n} \in S^3$. This *identification* makes rotation representation particularly tricky in $\mathbb{R}^3$ and clearly explains why unit quaternions, $\mathbf{q} \in S^3$, are a *double* cover of $\mathrm{SO}(3)$, since we must add the relation $\mathbf{q} = -\mathbf{q}$ to make these two spaces equivalent. We direct the reader to Hartley et al. (2013) who use the *gnonomic* projection to provide further geometric

[3] A diffeomorphism is a smooth invertible function that maps one differentiable manifold to another.

intuition for why $\mathrm{SO}(3)$ is a projective space.

Accordingly, in this dissertation, we parametrize rotations as the constrained quantities, $\mathbf{q}$ or $\mathbf{C}$. When dealing with perturbations about a given rotation (e.g., to compute updates to a state, or to propagate uncertainty), we use *small* rotations, $\delta\mathbf{C}$ or $\delta\mathbf{q}$, parametrized using three unconstrained parameters that we can assume are a one-to-one mapping to elements in $\mathrm{SO}(3)$ (since, for small rotations, $\|\boldsymbol{\phi}\| << \pi$).

2.3 Spatial Transforms

The rigid body transform $\mathbf{T}$ is a also a member of the matrix Lie group, the Special Euclidian group $\mathrm{SE}(3)$ and can be defined as a 4×4 matrix as follows:

$$\mathrm{SE}(3) = \{\mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} | \ \mathbf{C} \in \mathrm{SO}(3), \mathbf{t} \in \mathbb{R}^3\}. \tag{2.11}$$

As a member of a matrix Lie group, it also admits a surjective exponential map,

$$\mathbf{T} = \mathrm{Exp}\left(\boldsymbol{\xi}\right) = \exp\left(\boldsymbol{\xi}^\wedge\right) = \sum_{n=0}^{\infty} \frac{1}{n!}(\boldsymbol{\xi}^\wedge)^n \tag{2.12}$$

where $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho}^T & \boldsymbol{\phi}^T \end{bmatrix}^T \in \mathbb{R}^6$ are the exponential coordinates of rigid-body transforms and the wedge operator is overloaded (following Barfoot (2017)) as follows:

$$\boldsymbol{\xi}^\wedge \triangleq \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix}. \tag{2.13}$$

In practice, we can evaluate the exponential map through the Euler-Rodriguez formula (Equation (2.3)) and by computing the left-Jacobian of $\mathrm{SO}(3)$, $\mathbf{J}$,

$$\mathbf{T} = \mathrm{Exp}\left(\begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{C}(\boldsymbol{\phi}) & \mathbf{J}(\boldsymbol{\phi})\boldsymbol{\rho} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{2.14}$$

where

$$\mathbf{J}(\boldsymbol{\phi}) = \frac{\sin\phi}{\phi}\mathbf{1} + (1 - \frac{\sin\phi}{\phi})\mathbf{a}\mathbf{a}^T + \frac{1-\cos\phi}{\phi}\mathbf{a}^\wedge. \tag{2.15}$$

2.3.1 Applying Transforms

Applying our notation for coordinate frames (and referring back to Section 2.1), a transform, $\mathbf{T}_{vi}$ can be expressed as

$$\mathbf{T}_{vi} = \begin{bmatrix} \mathbf{C}_{vi} & \mathbf{t}_v^{iv} \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{2.16}$$

This allows us to use the homogenous point representation for $\mathbf{r}_i^{pi}$ and express the relation $\boldsymbol{r}_v^{pv} = \mathbf{T}_{vi}\boldsymbol{r}_i^{pi}$, or

$$\begin{bmatrix} \mathbf{r}_v^{pv} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{C}_{vi} & \mathbf{t}_v^{iv} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\mathbf{T}_{vi}} \begin{bmatrix} \mathbf{r}_i^{pi} \\ 1 \end{bmatrix}, \tag{2.17}$$

which is numerically equivalent to

$$\mathbf{r}_v^{pv} = \mathbf{C}_{vi}\mathbf{r}_i^{pi} + \mathbf{t}_v^{iv}. \tag{2.18}$$

2.4 Perturbations and Tangent Spaces

When solving optimization problems that involve rotations or rigid-body transforms, it is often useful to consider a small *perturbation* about an operating point. By leveraging a core property of Lie groups (that they are locally 'Euclidian'), we can construct an iterative algorithm that successively updates an operating point through optimal perturbations in some unconstrained tangent space. Using rotations as an example, we can perturb an operating point, $\overline{\mathbf{C}} \triangleq \operatorname{Exp}\left(\overline{\boldsymbol{\phi}}\right)$, in three different ways:

$$\mathbf{C} \leftarrow \begin{cases} \operatorname{Exp}\left(\delta\boldsymbol{\phi}^{\ell}\right)\overline{\mathbf{C}} & \text{left perturbation,} \\ \operatorname{Exp}\left(\overline{\boldsymbol{\phi}} + \delta\boldsymbol{\phi}^{m}\right) & \text{middle perturbation,} \\ \overline{\mathbf{C}}\operatorname{Exp}\left(\delta\boldsymbol{\phi}^{r}\right) & \text{right perturbation.} \end{cases} \tag{2.19}$$

In this dissertation, we will use the left and middle perturbations when appropriate. Using small angle approximations, the Euler-Rodriguez formula (Equation (2.3)) yields $\operatorname{Exp}\left(\delta\boldsymbol{\phi}\right) \approx \mathbf{1} + \delta\boldsymbol{\phi}^{\wedge}$, which allows us to write the useful approximation for the left perturbation:

$$\mathbf{C} \leftarrow \operatorname{Exp}\left(\delta\boldsymbol{\phi}^{\ell}\right)\overline{\mathbf{C}} \approx (\mathbf{1} + (\delta\boldsymbol{\phi}^{\ell})^{\wedge})\overline{\mathbf{C}}. \tag{2.20}$$

Similarly, we can write analogous expressions for a rigid body transform, $\mathbf{T} \in \mathrm{SE}(3)$, as composed of an operating point $\overline{\mathbf{T}} \triangleq \operatorname{Exp}\left(\overline{\boldsymbol{\xi}}\right)$, and a small perturbation about that operating point:

$$\mathbf{T} \leftarrow \begin{cases} \operatorname{Exp}\left(\delta\boldsymbol{\xi}^{\ell}\right)\overline{\mathbf{T}} & \text{left perturbation,} \\ \operatorname{Exp}\left(\overline{\boldsymbol{\xi}} + \delta\boldsymbol{\xi}^{m}\right) & \text{middle perturbation,} \\ \overline{\mathbf{T}}\operatorname{Exp}\left(\delta\boldsymbol{\xi}^{r}\right) & \text{right perturbation.} \end{cases} \tag{2.21}$$

We may form a similar expression,

$$\mathbf{T} \leftarrow \operatorname{Exp}\left(\delta\boldsymbol{\xi}^{\ell}\right)\overline{\mathbf{T}} = (\mathbf{1} + (\delta\boldsymbol{\xi}^{\ell})^{\wedge})\overline{\mathbf{T}}. \tag{2.22}$$

To derive locally linear systems from sets of rigid-body transforms, or 'poses', we can apply Equation (2.22). To update an operating point, we solve for $\delta\boldsymbol{\xi}^{\ell}$ and then use the constraint-sensitive update $\mathbf{T} \leftarrow \operatorname{Exp}\left(\delta\boldsymbol{\xi}^{\ell}\right)\overline{\mathbf{T}}$. Finally, we note that we will often drop the perturbation superscripts $(\cdot)^{\ell}$ and $(\cdot)^{m}$ after defining the perturbation scheme.

Remark (*Relating Pertubations*). We can relate all the left and middle perturbations through the left Jacobian of $\mathrm{SO}(3)$ with the following useful identity (Barfoot, 2017),

$$\operatorname{Exp}\left(\left(\boldsymbol{\phi}+\delta\boldsymbol{\phi}^{m}\right)\right) \approx \operatorname{Exp}\left(\mathbf{J}(\boldsymbol{\phi})\delta\boldsymbol{\phi}^{m}\right)\operatorname{Exp}\left(\boldsymbol{\phi}\right). \tag{2.23}$$

From this it follows that $\delta\boldsymbol{\phi}^{\ell} \approx \mathbf{J}(\boldsymbol{\phi})\delta\boldsymbol{\phi}^{m}$ and elucidates why $\mathbf{J}$ is called the *left* Jacobian. Similarly, for $\mathrm{SE}(3)$, $\delta\boldsymbol{\xi}^{\ell} \approx \boldsymbol{\mathcal{J}}(\boldsymbol{\xi})\delta\boldsymbol{\xi}^{m}$, since

$$\operatorname{Exp}\left(\left(\boldsymbol{\xi}+\delta\boldsymbol{\xi}^{m}\right)\right) \approx \operatorname{Exp}\left(\left(\boldsymbol{\mathcal{J}}(\boldsymbol{\xi})\delta\boldsymbol{\xi}^{m}\right)\right)\operatorname{Exp}\left(\boldsymbol{\xi}\right), \tag{2.24}$$

where $\boldsymbol{\mathcal{J}}$, the left Jacobian of $\mathrm{SE}(3)$, is defined as

$$\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}) \triangleq \begin{bmatrix} \mathbf{J}(\boldsymbol{\phi}) & \mathbf{Q}(\boldsymbol{\xi}) \\ \mathbf{0} & \mathbf{J}(\boldsymbol{\phi}) \end{bmatrix}, \tag{2.25}$$

with $\mathbf{Q}(\boldsymbol{\xi})$ having an analytic expression (see Barfoot (2017)).

2.5 Uncertainty Through Injection

We can also use perturbation theory to implicitly define uncertainty on constrained manifolds (see Barfoot and Furgale (2014) for a thorough discussion). Given a concentrated normal density, $\delta\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{6\times 6})$, we can *inject* this unconstrained density onto the Lie group through left perturbations about some mean using

$$\mathbf{T} = \operatorname{Exp}\left(\delta\boldsymbol{\xi}\right)\overline{\mathbf{T}}. \tag{2.26}$$

This allows us to keep track of a random variable, $\mathbf{T}$, by keeping its mean in group form, $\overline{\mathbf{T}}$, while its second statistical moment is stored as a standard 6×6 covariance matrix, $\boldsymbol{\Sigma}$.

We can define an analogous density for rotation matrices given normal densities over rotation perturbations $\delta\boldsymbol{\phi} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{3\times 3})$,

$$\mathbf{C} = \operatorname{Exp}\left(\delta\boldsymbol{\phi}\right)\overline{\mathbf{C}}, \tag{2.27}$$

and also, for unit quaternions,

$$\mathbf{q} = \operatorname{Exp}\left(\delta\boldsymbol{\phi}\right) \otimes \overline{\mathbf{q}} \tag{2.28}$$

where $\otimes$ refers to the standard quaternion product operator Sola (2017).

2.6 Deep Learning

Three of the four pseudo-sensors (Sun-BCNN, DPC, and HydraNet—Chapters 5 to 7) in this dissertation are built using neural networks and the tools of *deep learning* (LeCun et al., 2015). We briefly summarize these concepts here and refer the reader to Goodfellow et al. (2016) for a more thorough treatment.

2.6.1 Feed-forward Neural Networks

The basic premise of deep learning is that certain kinds of data are naturally decomposed into hierarchical structure. For instance, images may have local textures (e.g. edges), which compose basic primitives (e.g., leaves), which then combine to form semantic objects (e.g, a tree). Consequently, for this type of data, latent information may be efficiently extracted through an analogous hierarchical model. Typically, this is done through a neural network composed of multiple *layers* (the term *neural* comes from the loose biological basis for each layer, and *deep* refers to the amount of layers contained in state-of-the-art models). A standard neural layer computes a non-linear transformation from an input $\mathbf{z}_\ell$ to an output $\mathbf{z}_{\ell+1}$ as

$$\mathbf{z}_{\ell+1} = \mathbf{f}_\ell(\mathbf{z}) = \boldsymbol{\sigma}\left(\mathbf{W}_\ell \mathbf{z}_\ell + \mathbf{b}_\ell\right), \tag{2.29}$$

where $\mathbf{W}_\ell \in \mathbb{R}^{D_{\ell_{out}} \times D_{\ell_{in}}}$ and $\mathbf{b}_\ell \in \mathbb{R}^{D_{\ell_{out}}}$ are the parameters of the layer (often referred to as the *weight matrix* and *bias*, respectively) and $\boldsymbol{\sigma}(\cdot)$ is an element-wise *non-linearity*.

Remark (*Non-linearities*). The optimal choice of non-linearity is an area of active research. Some common examples include,

$$\sigma_i(x) = \begin{cases} \frac{1}{1+e^{-x}} & \text{Logistic,} \\ \frac{e^x - e^{-x}}{e^x + e^{-x}} & \text{Hyperbolic tangent, } \textit{tanh}, \\ \max(x, 0) & \text{Rectified Linear Unit, } \textit{ReLU}, \\ \begin{cases} x & \text{if } x \geq 0 \\ -\alpha x & \text{if } x < 0 \end{cases} & \text{Parametric ReLU, } \textit{PReLU}. \end{cases} \tag{2.30}$$

To produce a *feed-forward neural network*, L layers are composed in a hierarchy to create a single parametric function,

$$NN(\mathbf{x}; \boldsymbol{\pi}) = \mathbf{f}_L \circ \mathbf{f}_{L-1} \circ \cdots \circ \mathbf{f}_1(\mathbf{x}), \tag{2.31}$$

where $f \circ g$ refers to function composition, $f \circ g \triangleq f(g(\cdot))$, and $\boldsymbol{\pi} = \{\mathbf{W}_\ell, \mathbf{b}_\ell\}_{\ell=1}^{L}$ are the parameters of the network.

2.6.2 Convolutional Neural Networks

Convolutional Neural Networks (LeCun et al., 1989), or CNNs, are a particular form of neural network where at least one of the layers uses *convolution* in place of matrix multiplication. Due to their effi-

ciency in processing high-dimensional image data, they have become ubiquitous in computer vision applications (LeCun et al., 2015). Briefly, a convolution is an operation that transforms a continuous input signal $x(t)$ into a new signal $y(t)$ as

$$y(t) = \int x(s)k(t-s)ds, \tag{2.32}$$

where k is often called the *kernel* of the convolution. For two-dimensional discrete input signals, convolution is defined as

$$y(i,j) = \sum_p \sum_q I(i-p, j-q)K(p,q). \tag{2.33}$$

Remark (*Convolution vs. Cross-correlation*). Most deep learning libraries implement convolution as the cross-correlation operation

$$y(i,j) = \sum_p \sum_q I(i+p, j+q)K(p,q) \tag{2.34}$$

which is convolution with the kernel 'flipped.' Unlike convolution, cross-correlation is not commutative with respect to the input and kernel (but this is typically not important in deep learning contexts). We note that kernels learned with cross-correlation will be flipped relative to those learned with true convolution (Goodfellow et al., 2016).

In this discrete case, a kernel can be represented by a kernel matrix, $\mathbf{K}$, which can have different *size* (e.g., $\mathbf{K} \in \mathbb{R}^{3\times3}$), and we may compute Equation (2.33) at different spatial intervals called *strides*. To ensure that the convolution is defined at the boundaries of the signal, one can also use different *padding* strategies. Please refer to Goodfellow et al. (2016) for further information about these hyper-parameters.

By limiting the size of a kernel, we can use convolutional layers (parametrized by $\mathbf{K}$) to efficiently process high-dimensional signals. For images, a standard feed-forward layer would require a scalar weight for every pixel. In contrast, a convolutional layer can rely on a single kernel (with significantly less parameters than a weight matrix) to process an entire image with *shared weights*. This reduces the number of parameters of our network, but limits the type of spatial correlations we can model (since kernels can only capture local relations). To increase the potential modelling capacity of our network, we can use several independent kernels to process and output multiple *channels* (Figure 2.3). Finally, it is important to note that the convolution operator is particularly suited to visual data, as kernels can act as *filters* that pick out salient visual features like corners and edges.

2.6.3 Supervised Training

Given a (convolutional) neural network with a set of parameters $\boldsymbol{\pi}$ (where $\boldsymbol{\pi}$ may include weight and bias parameters, $\mathbf{W}_\ell, \mathbf{b}_\ell$, as well as convolutional kernel parameters, $\mathbf{K}_{k\ell}$) we can use *supervised*

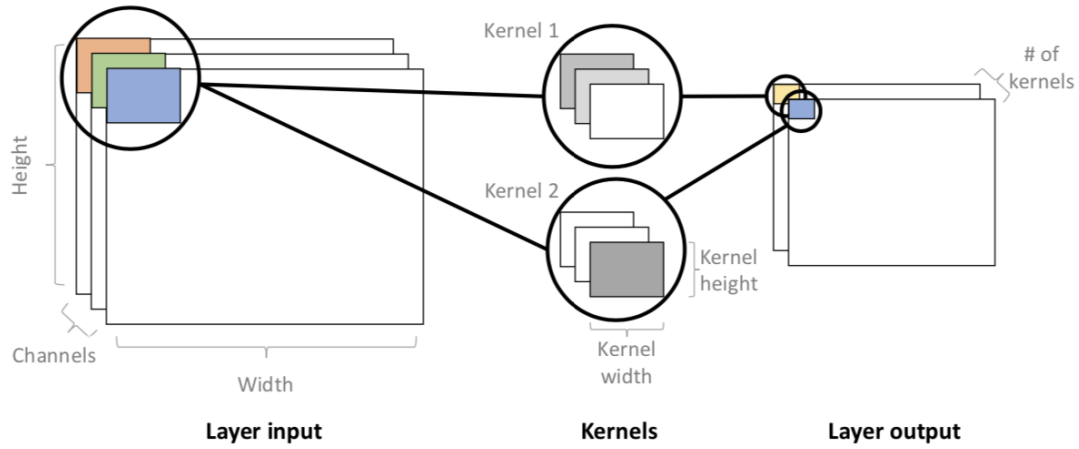


Figure 2.3: A convolutional layer with two kernels that operate over different channels of a two-dimensional input. Figure from Gal (2016).

training to obtain an optimal parameter set $\boldsymbol{\pi}^*$. Supervised training requires set of training pairs $\{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$ where $\mathbf{x} \in \mathbb{R}^{D_x}$ is an *input* and $\mathbf{y} \in \mathbb{R}^{D_y}$ is a *target* that corresponds to the desired output of our parametric model. To *train* the network, we find the parameters which minimize a *loss function* over this dataset,

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}}{\operatorname{argmin}} \mathcal{L}\left(\{NN(\mathbf{x}_i; \boldsymbol{\pi}), \mathbf{y}_i\}_{i=1}^N\right). \tag{2.35}$$

For example, a common loss function for regression problems is *mean squared error*,

$$\boldsymbol{\pi}^* = \underset{\boldsymbol{\pi}}{\operatorname{argmin}} \frac{1}{2N} \sum_{i=1}^{N} \|NN(\mathbf{x}_i; \boldsymbol{\pi}) - \mathbf{y}_i\|_2^2 . \tag{2.36}$$

. In this dissertation, we will use supervised training in Chapters 5 to 7 and we will explore different forms of loss functions for geometric quantities that do not allow for a simple Euclidian norms.

2.6.4 Practical Considerations

Optimization

To train a deep network—that is, to solve Equation (2.35) for optimal parameters—modern approaches rely on stochastic gradient descent (SGD) through back-propagation (LeCun et al., 2015). SGD avoids the computationally prohibitive task of computing a gradient over an entire dataset by approximating the gradient using a randomly selecting subset of training data called a *mini-batch*. In the majority of this dissertation, we rely on Adam, a modern SGD-based approach that also maintains an estimate of second-order curvature (Kingma and Ba, 2017).

Regularization

In order to prevent *overfitting* to a dataset (i.e., to obtain a model that generalizes to unseen data), the literature provides a number of methods. An important approach that is used in this dissertation is *dropout* (Srivastava et al., 2014). Dropout stochastically 'zeros-out' inputs of a particular layer with probability p,

$$\mathbf{z}_{\ell+1} = \mathbf{f}_\ell(\mathbf{z}) = \boldsymbol{\sigma}\left(\mathbf{W}_\ell \tilde{\mathbf{z}}_\ell + \mathbf{b}_\ell\right), \tag{2.37}$$

where $\tilde{\mathbf{z}}_\ell = \mathbf{b} \circledcirc \mathbf{z}_\ell$, $b_i \sim \text{Bernoulli}(p)$ and $\circledcirc$ refers to element-wise multiplication, also known as the Hadamard product. The intuition behind dropout is that it effectively creates an ensemble of smaller-parameter networks that will generally be less prone to overfitting than a commensurately-sized monolithic network. Importantly, dropout has a fundamental connection variational inference, which we exploit in Chapter 5.

Pooling and Spatial Invariance

A common technique in convolutional neural networks is *pooling* (Goodfellow et al., 2016). Pooling is a non-parametric operation that summarizes the output of a kernel in a particular region. For example, *max pooling* selects the maximum response of a kernel in demarcated regions of an input channel, thereby downsampling the resulting output. Pooling operations are designed to make convolution operations invariant to the spatial location of a particular kernel response. This is important in many classification tasks (e.g., a tree classifier should be invariant to the location of a particular leaf) but is a detriment to regression tasks where we would like to preserve spatial information. We explore building convolutional neural networks without pooling in Chapter 6.

Chapter 3

Classical Visual Odometry

> Eventually, my eyes were opened, and I really understood nature.
>
> — Claude Monet

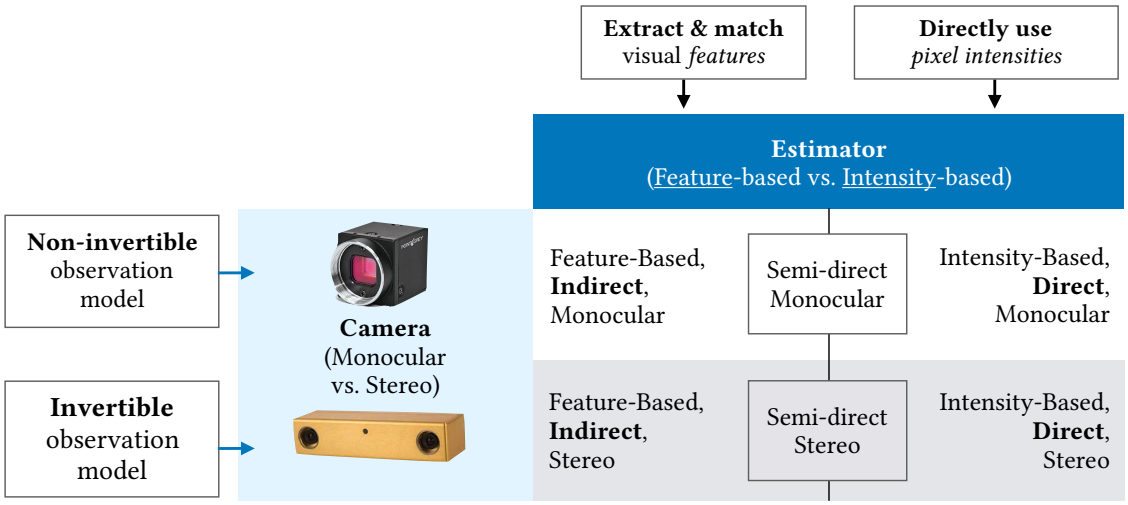


Figure 3.1: A taxonomy of different types of visual odometry.

Visual odometry (VO) has a rich history in mobile robotics and computer vision. As this dissertation largely deals with the improvement of a baseline visual odometry pipeline, we first outline the components of what we have chosen to be a canonical VO system. For a seminal tutorial on visual odometry and its more general cousin, visual SLAM, we refer the reader to two seminal papers: Scaramuzza and Fraundorfer (2011) and Cadena et al. (2016).

Remark (*VO Taxonomy*). VO can be largely divided along two dimensions (Figure 3.1): (1) the type of camera used to capture images and (2) the type of data association used to compute motion estimates.

Monocular vs. Stereo Camera: Monocular VO methods (Engel et al., 2018; Tsotsos et al., 2015) use a single camera to infer motion and can use a single compact, low-power vision sensor. They do not require any extrinsic calibration but must rely on known visual cues or external information (e.g., wheel odometry, inertial measurements) to provide metric egomotion estimates. Conversely, stereo VO methods (Engel et al., 2018; Leutenegger et al., 2015; Cvišić and Petrović, 2015) use a stereo camera to triangulate objects with metric scale. This allows stereo VO to provide metrically-accurate egomotion estimates. However, stereo methods rely on accurate extrinsic calibration, and their ability to resolve depth is limited by the baseline distance between the stereo pair and by the quality of stereo matches (which can be degraded by self-similar textures, occlusions, and foreshortening effects).

Direct vs. Indirect Data Association: The second distinction is based on the type of data association used to match sequential images and infer motion. Direct methods (Engel et al., 2018; Wang et al., 2017a) make the assumption of brightness constancy, and attempt to find the egomotion estimate that *directly* maximizes the similarity of pixel intensities between images. Indirect methods (Leutenegger et al., 2015; Cvišić and Petrović, 2015), conversely, rely on image features detectors to extract a set of salient landmarks or features, and then match these landmarks across images (typically by relying on a view-invariant descriptor).

3.1 Canonical VO Pipeline

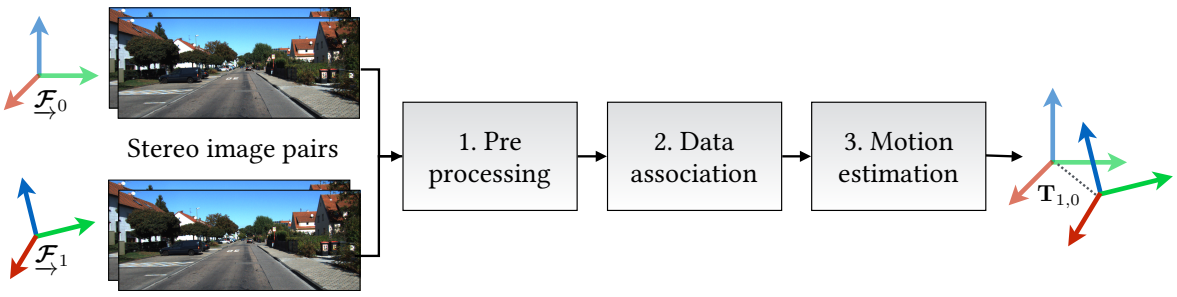
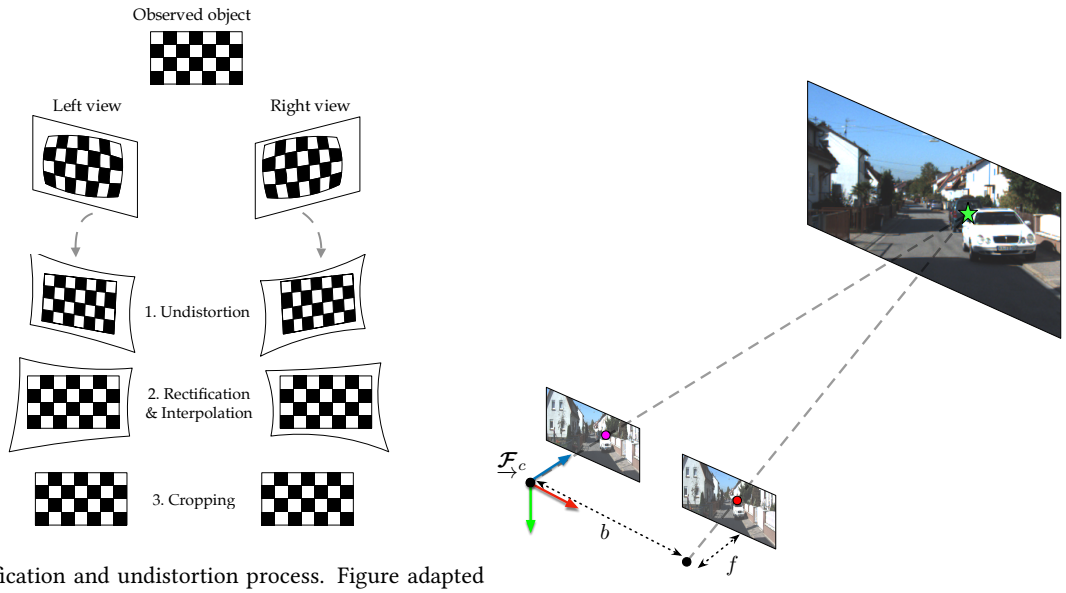


Figure 3.2: A 'classical' stereo visual odometry pipeline consists of several distinct components that have interpretable inputs and outputs.

In this dissertation, we apply our learned pseudo-sensors to a baseline stereo, indirect visual odometry pipeline (Figure 3.2). We choose this baseline system for its computational efficiency and robustness. We briefly summarize the main components of the pipeline here.



(a) Rectification and undistortion process. Figure adapted from Florez (2010).

(b) Ideal stereo camera.

Figure 3.3: We pre-process stereo images (left) to simulate an ideal stereo camera (right).

3.1.1 Preprocessing

During preprocessing, we use a lens model (assumed to be known apriori) to undistort each stereo image. Next, using the camera extrinsic parameters (also assumed to be known), we *rectify* the stereo pair such that the images can be assumed to come from two cameras whose principal axes are parallel (Figure 3.3). Finally, we also assume that the stereo camera intrinsics are known a priori or compute them through a calibration process (Furgale et al., 2013).

3.1.2 Data Association

Feature Extraction and Matching

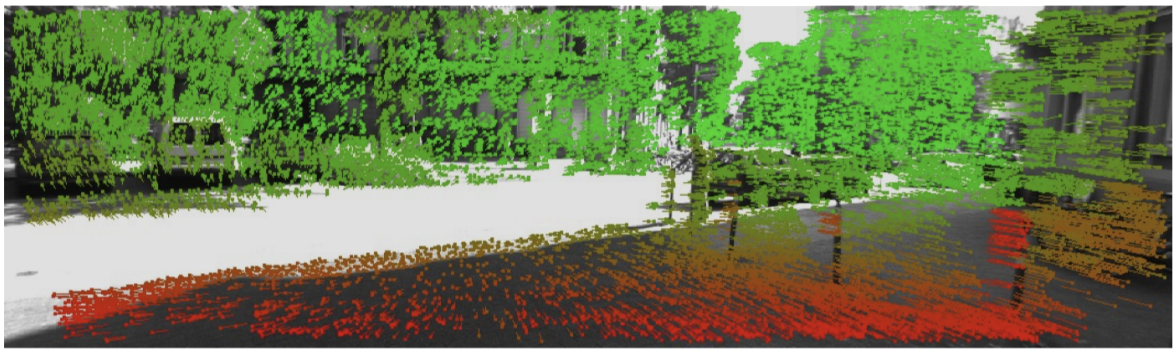


Figure 3.4: Feature tracking using `libviso2`, taken from Geiger et al. (2011). Colours correspond to depth.

Although a number of different types of indirect feature extraction and matching methods can be used towards this end, we choose to use the viso2 (Geiger et al., 2011) image feature extraction and matching algorithm as it is especially designed for sequential feature matching. In viso2, features are extracted using blob and corner masks with non-minimum and non-maximum suppression. Unlike other features detectors that do not assume a particular camera motion, viso2 assumes a smooth camera trajectory that permits fast matching through a simple sum-of-absolute-difference error metric based on Sobel filter responses. Features are matched across a stereo-pair and forward in time to ensure that a single feature exists across two consecutive stereo camera poses.

We refer to the ith feature as a point in space expressed in homogeneous coordinates in the camera frame as $\boldsymbol{p}_{i,c} \in \mathbb{P}^3$. Given our intrinsics and extrinsic calibration parameters, our idealized stereo-camera model, $\mathbf{f}$, projects a landmark expressed in homogeneous coordinates into image space, so that $\mathbf{y}_{i,c}$, the stereo pixel coordinates of landmark i in the camera frame, are given by

$$\mathbf{y}_{i,c} = \begin{bmatrix} u_l \\ v_l \\ d \end{bmatrix} = \mathbf{f}(\boldsymbol{p}_{i,c}) = \mathbf{f}(\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}) = \mathbf{M}\frac{1}{z}\boldsymbol{p}_{i,c}, \tag{3.1}$$

where

$$\mathbf{M} = \begin{bmatrix} f & 0 & c_u & 0 \\ 0 & f & c_v & 0 \\ 0 & 0 & 0 & fb \end{bmatrix}. \tag{3.2}$$

Here, $\{c_u, c_v\}$, f, and b are the principal points, focal length and baseline of the stereo camera respectively (computed through intrinsic calibration) and $d \triangleq u_l - u_r$ is the *disparity* of the feature. Note that in this formulation, the stereo camera frame is in the left optical centre. We can also define the inverse operation, $\mathbf{f}^{-1}$ (triangulation) as:

$$\boldsymbol{p}_{i,c} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{f}^{-1}\left(\begin{bmatrix} u_l \\ v_l \\ d \end{bmatrix}\right) = \begin{bmatrix} \frac{b}{d}(u_l - c_u) \\ \frac{b}{d}(v_l - c_v) \\ \frac{b}{d}f \\ 1 \end{bmatrix}. \tag{3.3}$$

Outlier Rejection

To filter out any stereo tracks that are *outliers*, we use a three-point random sample consensus algorithm (RANSAC, Fischler and Bolles (1981)) based on an analytic solution to the six degree-of-freedom motion (Umeyama, 1991) (refer to Appendix B for more details).

3.1.3 Maximum Likelihood Motion Solution

Finally, we compute the rigid-body transform between to stereo camera frames using maximum likelihood estimation. We define the rigid-body transform, $\mathbf{T}_t \in \mathrm{SE}(3)$, to be the rigid-body transform

between two subsequent stereo camera poses, $\underrightarrow{\mathcal{F}}_{c_0}$ and $\underrightarrow{\mathcal{F}}_{c_1}$,

$$\mathbf{T}_t = \mathbf{T}_{c_1 w}\mathbf{T}_{c_0 w}^{-1}, \tag{3.4}$$

where $\underrightarrow{\mathcal{F}}_w$ is a predefined world frame. After data association, we assume we have a set of N_t matches, $\{\mathbf{y}_{i,c_0}, \mathbf{y}_{i,c_1}\}_{i=1}^{N_t}$, between visual landmarks in the two camera frames. For each match, we define an error function, $\mathbf{e}_i(\mathbf{T}_t)$, that relates the rigid transform to these stereo feature matches. Throughout this dissertation, we assume that these errors are corrupted by zero-mean independent Gaussian noise with the (potentially heteroscedastic) covariance, $\mathbf{\Sigma}_{i,t}$;

$$\mathbf{e}_i(\mathbf{T}_t) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma}_{i,t}\right). \tag{3.5}$$

Under this noise model, the maximum likelihood transform, $\mathbf{T}_t^*$, is given by

$$\mathbf{T}_t^* = \underset{\mathbf{T}\in \mathrm{SE}(3)}{\operatorname{argmax}} \prod_{i=1}^{N_t} p(\mathbf{e}_i(\mathbf{T}_t)) = \underset{\mathbf{T}\in \mathrm{SE}(3)}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{N_t} \mathbf{e}_i(\mathbf{T}_t)^T \mathbf{\Sigma}_{i,t}^{-1}\mathbf{e}_i(\mathbf{T}_t). \tag{3.6}$$

We will define the error function in two different ways.

Point Cloud Error

First, we can follow classical approach (Maimone et al., 2007) and define $\mathbf{e}_i(\mathbf{T}_t)$ based on a three-dimensional point cloud error. To do this, we invert our stereo camera model to triangulate pairs of points in each frame, $\boldsymbol{p}_{i,c_0} = \mathbf{f}^{-1}(\mathbf{y}_{i,c_0})$ and $\boldsymbol{p}_{i,c_1} = \mathbf{f}^{-1}(\mathbf{y}_{i,c_1})$, and then define a three-dimensional error,

$$\mathbf{e}_i(\mathbf{T}_t) = \mathbf{D}(\boldsymbol{p}_{i,c_1} - \mathbf{T}_t\boldsymbol{p}_{i,c_0}) \in \mathbb{R}^3, \tag{3.7}$$

where $\mathbf{D} = \begin{bmatrix}\mathbf{1}_{3\times 3} & \mathbf{0}\end{bmatrix} \in \mathbb{R}^{3\times 4}$ converts homogenous coordinates into Euclidian coordinates.
We can then follow Maimone et al. (2007) and assume each stereo projection is corrupted by additive Gaussian noise,

$$\mathbf{y}_{i,c} \sim \mathcal{N}\left(\bar{\mathbf{y}}_{i,c}, \mathbf{R}_{i,c}\right), \tag{3.8}$$

and compute a density on the error function itself through first order noise propagation. This gives the density

$$\mathbf{e}_i(\mathbf{T}_t) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma}_{i,t}\right), \tag{3.9}$$

where

$$\mathbf{\Sigma}_{i,t} = \mathbf{D}\mathbf{G}_{i,c_1}\mathbf{R}_{i,c_1}\mathbf{G}_{i,c_1}^T\mathbf{D}^T + \mathbf{D}\mathbf{T}_t\mathbf{G}_{i,c_0}\mathbf{R}_{i,c_0}\mathbf{G}_{i,c_0}^T\mathbf{T}_t^T\mathbf{D}^T, \tag{3.10}$$

with $\mathbf{G}_{i,c} = \left.\dfrac{\partial \mathbf{f}^{-1}}{\partial \mathbf{y}}\right|_{\mathbf{y}_{i,c}}$.

Reprojection Error

Alternatively, we can represent reprojection errors in the second frame directly as

$$\mathbf{e}_i(\mathbf{T}_t) = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}_t\mathbf{f}^{-1}(\mathbf{y}_{i,c_0})), \tag{3.11}$$

and assume the following simple noise model

$$\mathbf{e}_i(\mathbf{T}_t) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Sigma}_{i,t}\right) = \mathcal{N}\left(\mathbf{0}, \mathbf{R}_{i,t}\right), \tag{3.12}$$

where we abuse notation (slightly) and replace the index for the camera frames c_0 or c_1 with t to indicate that this covariance refers to the reprojection error that involves both sets of features.

Importantly, Sibley et al. (2007) show that using reprojection error (as compared to 3D point cloud error) results in less biased estimates for long-range stereo triangulation. Consequently, we favour this latter formulation in the large majority of our work (the one exception being the initial work on isotropic PROBE described in Appendix A).

Solution via Gauss-Newton Optimization

In either case, we have now defined a weighted nonlinear least squares problem which can be solved iteratively using standard techniques. For our purposes, we opt to use Gauss-Newton optimization and follow Barfoot (2017) to optimize constrained poses.

Namely, at a given iteration n, we linearize the error function $\mathbf{e}_i(\mathbf{T}_t)$, about an operating point $\mathbf{T}_t^{(n)} \in \mathrm{SE}(3)$, which results in a quadratic approximation to Equation (3.6). To linearize, we consider the left perturbations $\delta\boldsymbol{\xi} \in \mathbb{R}^6$ represented in exponential coordinates:

$$\mathbf{T}_t = \mathrm{Exp}\left(\delta\boldsymbol{\xi}\right)\mathbf{T}_t^{(n)} \approx (\mathbf{1} + \delta\boldsymbol{\xi}^{\wedge})\mathbf{T}_t^{(n)}. \tag{3.13}$$

This allows us to transform Equation (3.6) into a linear least squares objective in $\delta\boldsymbol{\xi}$:

$$\mathcal{L}(\delta\boldsymbol{\xi}) = \frac{1}{2}\sum_{i=1}^{N_t}\left(\mathbf{e}_i - \mathbf{J}_i\delta\boldsymbol{\xi}\right)^T\mathbf{\Sigma}_i^{-1}\left(\mathbf{e}_i - \mathbf{J}_i\delta\boldsymbol{\xi}\right) \tag{3.14}$$

where $\mathbf{J}_i = \left.\dfrac{\partial\mathbf{e}_i}{\partial\delta\boldsymbol{\xi}}\right|_{\mathbf{T}_t^{(n)}}$, $\mathbf{e}_i = \mathbf{e}_i(\mathbf{T}_t^{(n)})$, and $\mathbf{\Sigma}_i = \mathbf{\Sigma}_{i,t}(\mathbf{T}_t^{(n)})$. The minimum to this objective can be solved for analytically by solving the normal equations. This results in the optimal parameters,

$$\delta\boldsymbol{\xi}^{\star} = \left(\sum_{i=1}^{N_t}\mathbf{J}_i^T\mathbf{\Sigma}_i^{-1}\mathbf{J}_i\right)^{-1}\sum_{i=1}^{N_t}\mathbf{J}_i^T\mathbf{\Sigma}_i^{-1}\mathbf{e}_i. \tag{3.15}$$

Given $\delta\boldsymbol{\xi}^{\star}$, we can update the operating point using the constraint-sensitive update

$$\mathbf{T}^{(n+1)} = \mathrm{Exp}\left(\delta\boldsymbol{\xi}^{\star}\right)\mathbf{T}^{(n)}, \tag{3.16}$$

and iterate until convergence. See Appendix B for more details and an analytic expression for $\mathbf{J}_i$. There are many reasonable choices for both the initial transform $\mathbf{T}^{(0)}$ and for the conditions under which we terminate iteration. For most visual odometry applications, it suffices to initialize the estimated transform to identity, and iteratively perform the update given by Equation (3.16) until we see a relative change in the squared error of less than one percent after an update.

3.2 Robust Estimation

Since Equation (3.14) assigns cost values that grow quadratically with measurement error, it is very sensitive to outlier measurements that may persist through RANSAC. A common solution to this problem is to replace the quadratic loss function with one that is less sensitive to large measurement errors (MacTavish and Barfoot, 2015). These robust cost functions are collectively known as M-estimators[1], and many variants exist. Each uses a re-weighting function, $\rho(\cdot)$, to define a new optimization problem,

$$\mathbf{T}^*_{\text{RLS}} = \underset{\mathbf{T}\in\text{SE}(3)}{\text{argmin}} \sum_{i=1}^{N} \rho\left(\sqrt{\mathbf{e}_i{}^T \boldsymbol{\Sigma}_i^{-1} \mathbf{e}_i}\right) = \underset{\mathbf{T}\in\text{SE}(3)}{\text{argmin}} \sum_{i=1}^{N} \rho(\epsilon_i) = \underset{\mathbf{T}\in\text{SE}(3)}{\text{argmin}}\, \mathcal{L}_{\text{RLS}}(\mathbf{T}), \tag{3.17}$$

where we have defined $\epsilon_i \triangleq \sqrt{\mathbf{e}_i{}^T \boldsymbol{\Sigma}_i^{-1} \mathbf{e}_i}$ (and dropped the t subscript for clarity). The basic idea with M-estimation is to use a $\rho(\cdot)$ that reduces the influence of large ϵ below that of the quadratic $\rho(\epsilon) = \frac{1}{2}\epsilon^2$. There are several examples of such functions, including,

$$\rho(\epsilon) = \begin{cases} \frac{c^2}{2}\log\left(1+\frac{\epsilon^2}{c^2}\right) & \text{Cauchy,} \\ \frac{1}{2}\frac{\epsilon^2}{c^2+\epsilon^2} & \text{Geman-McClure (Geman et al., 1992),} \\ \begin{cases} \frac{\epsilon^2}{2} & \text{if } \epsilon < c \\ c\epsilon - \frac{c^2}{2} & \text{if } \epsilon \geq c \end{cases} & \text{Huber (Huber, 1964).} \end{cases} \tag{3.18}$$

where the constant c can be set with reference to asymptotic efficiency relative to a unit Gaussian (Holland and Welsch, 1977). To solve Equation (3.17), it is common in the literature to apply the technique of *iteratively reweighted least squares* (IRLS) (Holland and Welsch, 1977). To do this, we define a new non-linear least squares minimization problem,

$$\mathbf{T}^*_{\text{IRLS}} = \underset{\mathbf{T}\in\text{SE}(3)}{\text{argmin}} \frac{1}{2} \sum_{i=1}^{N} \mathbf{e}_i{}^T \mathbf{M}_i \mathbf{e}_i = \underset{\mathbf{T}\in\text{SE}(3)}{\text{argmin}}\, \mathcal{L}_{\text{IRLS}}(\mathbf{T}) \tag{3.19}$$

where we define these new weights, $\mathbf{M}_i$, based on an *influence function*, $\psi(\cdot)$ as

$$\mathbf{M}_i = \frac{1}{\epsilon_i} \underbrace{\left.\frac{\partial \rho}{\partial \epsilon}\right|_{\epsilon_i}}_{\psi(\cdot)} \boldsymbol{\Sigma}_i^{-1}, \tag{3.20}$$

[1] M, for *maximum-likelihood-type* since they generalize the basic maximum likelihood solution (Barfoot, 2017).

and solve it using the Gauss-Newton approach presented in Section 3.1.3. We claim that upon convergence, $\mathbf{T}^*_{\text{IRLS}}$ will also minimize Equation (3.17). To see why, consider that

$$\frac{\partial \mathcal{L}_{\text{RLS}}}{\partial \delta \boldsymbol{\xi}} = \sum_i^N \frac{\partial \rho}{\partial \epsilon_i} \frac{\partial \epsilon_i}{\partial \mathbf{e}_i} \frac{\partial \mathbf{e}_i}{\partial \delta \boldsymbol{\xi}} = \sum_i^N \frac{1}{\epsilon_i} \frac{\partial \rho}{\partial \epsilon_i} \, {\mathbf{e}_i}^T \boldsymbol{\Sigma}_i^{-1} \frac{\partial \mathbf{e}_i}{\partial \delta \boldsymbol{\xi}}, \tag{3.21}$$

where he have used the fact that $\frac{\partial \epsilon_i}{\partial \mathbf{e}_i} = \frac{1}{\epsilon_i} \, {\mathbf{e}_i}^T \boldsymbol{\Sigma}_i^{-1}$. Now using our definition of $\mathbf{M}_i$, we can write,

$$\frac{\partial \mathcal{L}_{\text{RLS}}}{\partial \delta \boldsymbol{\xi}} = \sum_i^N {\mathbf{e}_i}^T \underbrace{\frac{1}{\epsilon_i} \frac{\partial \rho}{\partial \epsilon_i} \boldsymbol{\Sigma}_i^{-1}}_{\mathbf{M}_i(\mathbf{T})} \frac{\partial \mathbf{e}_i}{\partial \delta \boldsymbol{\xi}} = \sum_i^N {\mathbf{e}_i}^T \mathbf{M}_i(\mathbf{T}) \frac{\partial \mathbf{e}_i}{\partial \delta \boldsymbol{\xi}}, \tag{3.22}$$

where we have made the dependence on $\mathbf{T}$ explicit. We could potentially proceed to set this gradient to $\mathbf{0}$ and attempt to solve for an optimal update $\delta \boldsymbol{\xi}$. However, due to $\mathbf{M}_i(\mathbf{T})$, this may be difficult in general. Instead, we note that if we evaluate $\mathbf{M}_i(\mathbf{T})$ at the current operating point, $\mathbf{T}^{(n)}$, (i.e., we *re-weight* the loss) we are then left with the equivalent normal equations that solve $\frac{\partial \mathcal{L}_{\text{IRLS}}}{\partial \delta \boldsymbol{\xi}} = \mathbf{0}$.

Furthermore, upon convergence, our solution to the iteratively re-weighted problem $\mathbf{T}^{(n)} = \mathbf{T}^*_{\text{IRLS}}$ will also minimize the robust objective Equation (3.17), since we must have that,

$$\left. \frac{\partial \mathcal{L}_{\text{IRLS}}}{\partial \delta \boldsymbol{\xi}} \right|_{\mathbf{T}^*_{\text{IRLS}}} = \left. \frac{\partial \mathcal{L}_{\text{RLS}}}{\partial \delta \boldsymbol{\xi}} \right|_{\mathbf{T}^*_{\text{IRLS}}} = \mathbf{0}. \tag{3.23}$$

3.3 Outstanding Issues

Finally, we summarize three high-level limitations of such a canonical visual odometry pipeline that we will address with learned pseudo-sensors: efficiency, systematic bias and homoscedastic uncertainnty.

Table 3.1: Data efficiency vs. computational efficiency

Synopsis	Addressed by
Classical VO pipelines face a difficult-to-optimize trade-off between using all of the information contained within image and while still remaining computationally tractable.	PROBE, DPC-Net, SuperBCNN, HydraNet

Table 3.2: Systematic bias

Synopsis	Addressed by
Stereo visual odometry can incur systematic bias through poor extrinsic or intrinsic calibration, stereo triangulation errors, poor feature *spread* (i.e., concentration of features on one side of an image), and poor data association due self-similar textures.	DPC-Net

Table 3.3: Homoscedastic uncertainty

Synopsis	Addressed by
Stationary, homoscedastic noise in observation models can often reduce the consistency and accuracy of state estimates. This is especially true for complex, inferred measurement models.	PROBE, Sun-BCNN, HydraNet

Chapter 4

Predictive Robust Estimation

> Information is the resolution of uncertainty.
>
> Claude Shannon

4.1 Introduction

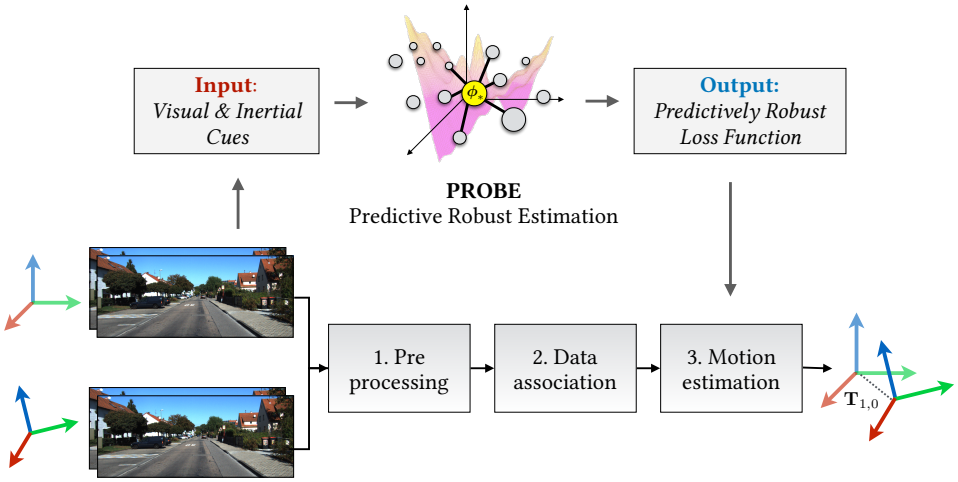


Figure 4.1: PROBE builds a predictive noise model for stereo visual odometry.

The first pseudo-sensor we present is a technique we call PRedictive ROBust Estimation, or PROBE. This approach uses non-parametric learning to build a model for anisotropic observation covariances for a stereo visual odometry pipeline. Namely, we apply the method of Generalized Kernel (GK) estimation to a Bayesian treatment of covariance estimation. We show that by assuming a particular covariance prior over re-projection errors, we can then naturally derive a robust least squares objective that resembles the widely-used Cauchy loss. The parameters of this robust loss are predicted (hence *predictive* robust estimation) for each error term as a function of a prediction space that we define.

Remark (*Associated Publications*). PROBE was initially published as a simpler non-Bayesian technique that learned isotropic covariances through a k-nearest-neighbours approach (see Appendix A for more details). The following two publications summarize this initial technique:

1. Peretroukhin, V., Clement, L., Giamou, M., and Kelly, J. (2015a). PROBE: Predictive robust estimation for visual-inertial navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'15)*, pages 3668–3675, Hamburg, Germany

2. Peretroukhin, V., Clement, L., and Kelly, J. (2015b). Get to the point: Active covariance scaling for feature tracking through motion blur. In *Proceedings of the IEEE International Conference on Robotics and Automation Workshop on Scaling Up Active Perception*, Seattle, Washington, USA.

We significantly extended this technique to full anisotropic covariances and through GK estimation in collaboration with William Vega-Brown at MIT. William was primarily responsible for the mathematical formulation of Generalized Kernels for VO and provided code to perform efficient GK estimation. I was responsible for the VO formulation, the integration of GK estimation into the VO pipeline, and all of the experimental work.

3. Peretroukhin, V., Vega-Brown, W., Roy, N., and Kelly, J. (2016). PROBE-GK: Predictive robust estimation using generalized kernels. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 817–824 .

We will present this latter technique in this chapter.

4.2 Motivation

Robot navigation relies on an accurate quantification of sensor noise or uncertainty in order to produce reliable state estimates. In practice, this uncertainty is often fixed for a given sensor and experiment, whether by automatic calibration or by manual tuning. Although a fixed measure of uncertainty may be reasonable in certain static environments, dynamic scenes frequently exhibit many effects that corrupt a portion of the available observations. For visual sensors, these effects include, for example, self-similar textures, variations in lighting, moving objects, and motion blur. Further, there may be useful information available in these observations that would normally be rejected by a fixed-threshold outlier rejection scheme. Ideally, we would like to retain some of these observations in our estimator, while still placing more trust in observations that do not suffer from such effects.

4.3 Related Work

There is a large and growing body of work on the task of deriving consistent uncertainty estimates for state estimation. It possible, for example, to develop better bespoke covariance estimates for a particu-

lar estimator modality by choosing a higher-fidelity covariance propagation schema (e.g., applying the implicit function theorem to propagate uncertainties in a laser-scan registration optimization problem, Censi (2007)).

Alternatively, time-varying uncertainties can be formulated using the Adaptive Kalman Filter and related work (Gustafsson and Gustafsson, 2000). This classical approach adapts measurement and process covariances based on past estimation errors. Since it is adaptive, it cannot cope with fast-changing environments (since it must *adapt* to past measurements, instead of *predicting* the uncertainty of current measurements). Further, these techniques generally do not provide a straight-forward mechanism to leverage rich high dimensional data like images.

To deal with such high-dimensional sensor data, an alternative set of approaches build predictive uncertainty models by mapping sensor data to a lower dimensional *prediction space*. By carefully choosing the components of this space, it is possible to learn covariances parametrically through weight vectors (e.g., for range/bearing sensors like the work of Hu and Kantor (2015)) or neural networks (for cameras, like Liu et al. (2018)) or non-parametrically through kernel based methods (Vega-Brown et al., 2013; Vega-Brown and Roy, 2013) or Gaussian Processes (GPs) (Ko and Fox, 2009).

Non-parametric models can also be extended to more general covariance matrix models that infer densities over positive-definite matrices (Wilson and Ghahramani, 2011) or apply GP inference for each element of these matrices (Melkumyan and Ramos, 2011). In both cases, however, inference is often too slow to perform in real-time because they are not designed to handle high-dimensional feature spaces. Conversely, the inspiration for this work showed that is possible to build non-GP-based kernel-based models that predict covariances efficiently for laser-based scan matching odometry with (Vega-Brown et al., 2013) and without (Vega-Brown and Roy, 2013) ground-truth pose information .

In the visual domain, estimators often cope with varying uncertainty only indirectly through (static) robust loss functions (Kerl et al., 2013) or modified RANSAC procedures with statistical hypothesis testing (as a way to ensure that tracked visual features have normally distributed residuals) (Tsotsos et al., 2015). Unlike our predictive approach which can adjust covariances from a single observation, these techniques are not designed to cope with heteroscedastic measurement noise that may occur within a single image. Alternatively, it is also possible to carefully select an optimally observable feature subset based on the highest *informativeness* - a measure calculated based on the observability metrics of the camera motion (Zhang and Vela, 2015). Observability, however, is governed by the 3D location of the features, and therefore cannot predict systematic feature degradation due to environmental or sensor-based effects that are independent of location (e.g., self-similar textures). Finally, since this work was published, a technique that is very close in spirit was published in the context of robust fundamental matrix estimation (Ranftl and Koltun, 2018). This approach uses deep learning to predict the weights of an iteratively reweighted least squares routine–effectively mimicking our predictively robust loss but without the information-theoretic basis.

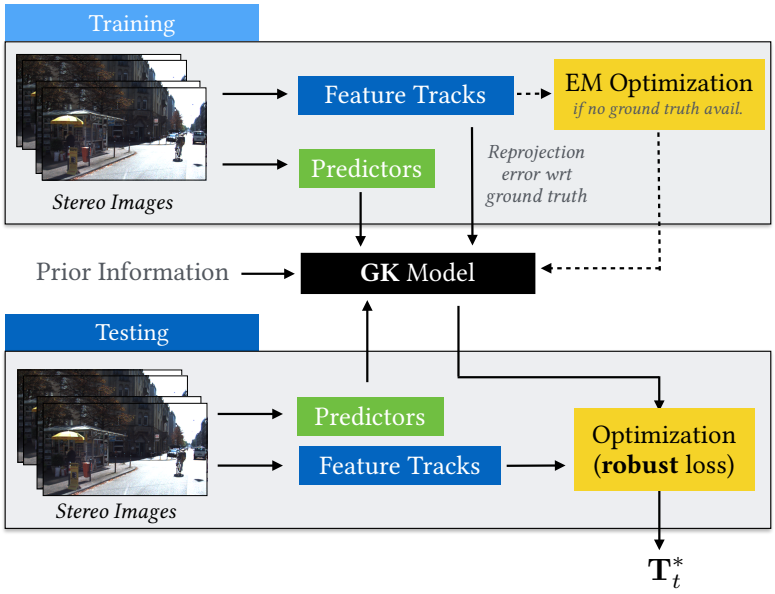


Figure 4.2: PROBE builds a predictive noise model using Generalized Kernels (GK) for stereo visual odometry. The model can be trained with and without groundtruth egomotion.

4.4 Predictive Robust Estimation for VO

We present a principled, data-driven way to build a noise model for visual odometry. We leverage recent advances in covariance estimation (Vega-Brown and Roy, 2013; Vega-Brown et al., 2013) to formulate a predictive robust estimator for a stereo visual odometry pipeline. We frame the traditional non-linear least squares optimization problem as a problem of maximum likelihood estimation with a Gaussian noise model, and infer a distribution over the covariance matrix of the Gaussian noise from a predictive model learned from training data. This results in a Student's t distribution for reprojection error, and naturally yields a robust nonlinear least-squares optimization problem. In this way, we can predict, in a principled manner, how informative each visual feature is with respect to the final state estimate, which allows our approach to intelligently weight observations to produce more accurate odometry estimates.

4.4.1 Bayesian Noise Model for Visual Odometry

We adopt the motion solution for visual odometry based on reprojection errors presented in Section 3.1.3. In brief, this technique assumes independent Gaussian errors on stereo reprojections of a landmark from one frame, $\underrightarrow{\mathcal{F}}_{c_0}$ into a subsequent frame, $\underrightarrow{\mathcal{F}}_{c_1}$:

$$\mathbf{e}_i(\mathbf{T}_t) = \mathbf{e}_{i,t} = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}_t \mathbf{f}^{-1}(\mathbf{y}_{i,c_0})) \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{i,t}) \,. \tag{4.1}$$

Maximizing the likelihood of these errors is then equivalent to solving the following weighted non-linear least squares objective for $\mathbf{T}_t \in \mathrm{SE}(3)$ the rigid-body transform that transforms points in $\underrightarrow{\mathcal{F}}_{c_0}$

to those in $\underrightarrow{\mathcal{F}}_{c_1}$:

$$\mathbf{T}_t^* = \underset{\mathbf{T}\in \mathrm{SE}(3)}{\operatorname{argmax}} \prod_{i=1}^{N_t} p(\mathbf{e}_{i,t}; \mathbf{T}_t, \mathbf{R}_{i,t}) \tag{4.2}$$

$$= \underset{\mathbf{T}\in \mathrm{SE}(3)}{\operatorname{argmin}} \frac{1}{2} \sum_{i=1}^{N_t} {\mathbf{e}_{i,t}}^T \mathbf{R}_{i,t}^{-1} \mathbf{e}_{i,t}. \tag{4.3}$$

With PROBE, instead of treating $\mathbf{R}_{i,t}$ as fixed, we build a model for it as a function of some useful *predictor*, $\boldsymbol{\phi}_{i,t}$,

$$\mathbf{R}_{i,t} = \mathbf{R}(\boldsymbol{\phi}_{i,t}). \tag{4.4}$$

Each predictor can be computed based on the stereo track ($\{\mathbf{y}_{i,c_0}, \mathbf{y}_{i,c_1}\}$) and additional visual[1] and inertial cues, allowing us to model effects like motion blur and self-similar textures. Further, instead of treating the covariance as a point function $\mathbf{R}(\boldsymbol{\phi}_{i,t})$, we instead build a non-parametric model of covariance *density* based on a training dataset, $\mathcal{D}$,

$$p(\mathbf{R}_{i,t}) = p(\mathbf{R}|\mathcal{D}, \boldsymbol{\phi}_{i,t}). \tag{4.5}$$

We will seek the transform that then maximizes the posterior predictive distribution of the errors, given this posterior:

$$\mathbf{T}_t^* = \underset{\mathbf{T}\in \mathrm{SE}(3)}{\operatorname{argmax}} \prod_{i=1}^{N_t} \int p(\mathbf{e}_{i,t}; \mathbf{T}_t|\mathbf{R}_{i,t}) p(\mathbf{R}|\mathcal{D}, \boldsymbol{\phi}_{i,t}) d\mathbf{R}. \tag{4.6}$$

Although at first this may seem unwieldy, we present an efficient method for computing the posterior and show that a particular formulation allows it to be marginalized out analytically to arrive at a simple posterior predictive distribution with a straight-forward objective for achieving a maximum likelihood egomotion transform.

4.4.2 Generalized Kernels

To do this, we leverage Generalized Kernel (GK) estimation (Vega-Brown et al., 2014). GK estimation combines the benefits of kernel density estimation with Bayesian inference. The basic idea is as follows. Consider a dataset of inputs, $\mathbf{x}$, outputs, $\mathbf{y}$, and a dataset of independent observations $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$. We are given a new 'test' input $\mathbf{x}^*$, and are asked to infer the likelihood of a observing a given output at this input:

$$p(\mathbf{y}|\mathbf{x}^*, \mathcal{D}). \tag{4.7}$$

If we associate a set of latent parameters, $\boldsymbol{\pi}$, with each input $\mathbf{x}$, and assume a known likelihood function $p(\mathbf{y}|\boldsymbol{\pi})$, we can infer a distribution over $\boldsymbol{\pi}$ and then marginalize it out to arrive at the desired likelihood

[1] Including potentially data from all four images in the pair of stereo images.

$$p(\mathbf{y}|\mathbf{x}^*, \mathcal{D}) = \int_{\boldsymbol{\pi}} \underbrace{p(\mathbf{y}|\boldsymbol{\pi}^*)}_{\text{Known likelihood function}} \underbrace{p(\boldsymbol{\pi}^*|\mathbf{x}^*, \mathcal{D})}_{\text{Parameter posterior}} d\boldsymbol{\pi}^*. \tag{4.8}$$

This is called the posterior predictive distribution. Using Bayes rule, we can write

$$p(\boldsymbol{\pi}^*|\mathbf{x}^*, \mathcal{D}) \propto \int \left(\prod_{i=1}^{N} p(\mathbf{y}_i|\boldsymbol{\pi}_i) d\boldsymbol{\pi}_i \right) p(\boldsymbol{\pi}_{1:N}, \boldsymbol{\pi}^*|\mathbf{x}^*, \mathbf{x}_{1:N}) \tag{4.9}$$

The technique of generalized kernels makes the assumption that the parameters $\boldsymbol{\pi}_{1:N}$ are conditionally independent given the target parameters, $\boldsymbol{\pi}^*$. This gives the distribution:

$$p(\boldsymbol{\pi}_{1:N}, \boldsymbol{\pi}^*|\mathbf{x}^*, \mathbf{x}_{1:N}) = \left(\prod_{i=1}^{N} p(\boldsymbol{\pi}_i|\boldsymbol{\pi}^*\mathbf{x}^*, \mathbf{x}_i) \right) p(\boldsymbol{\pi}^*|\mathbf{x}^*), \tag{4.10}$$

which combined with Equation (4.9) results in

$$p(\boldsymbol{\pi}^*|\mathbf{x}^*, \mathcal{D}) \propto \prod_{i=1}^{N} \underbrace{p(\mathbf{y}_i|\boldsymbol{\pi}^*, \mathbf{x}_i, \mathbf{x}^*)}_{\text{extended likelihood}} \underbrace{p(\boldsymbol{\pi}^*|\mathbf{x}^*)}_{\text{Prior}}. \tag{4.11}$$

Now, the pièce de résistance of generalized kernels is that the *extended* likelihood (the first factor above) can be written as function of the known likelihood $p(\mathbf{y}_i|\boldsymbol{\pi}_i)$ if we assume it is the maximum entropy distribution whose information divergence from the likelihood is bounded by the metric $\rho(\mathbf{x}^*, \mathbf{x}_i)$. Specifically, in Vega-Brown et al. (2014), it is shown that under these assumptions, the extended likelihood must have the form:

$$p(\mathbf{y}|\boldsymbol{\pi}^*, \mathbf{x}, \mathbf{x}^*) \propto p(\mathbf{y}|\boldsymbol{\pi})^{k(\mathbf{x}^*, \mathbf{x})}, \tag{4.12}$$

where $k(\cdot, \cdot)$ is a kernel function[2] that is uniquely defined by ρ. The intuition behind this is that we expect the extended likelihood to equal the known likelihood if $\mathbf{x}^* = \mathbf{x}_i$ (and therefore $\boldsymbol{\pi}^* = \boldsymbol{\pi}_i$, resulting in $p(\mathbf{y}_i|\boldsymbol{\pi}^*, \mathbf{x}_i, \mathbf{x}^*) = p(\mathbf{y}_i|\boldsymbol{\pi}_i)$) and diverge in some smooth way when $\mathbf{x}^* \neq \mathbf{x}_i$. Combining Equation (4.11) with Equation (4.12), we arrive at an expression for the posterior over parameters as

$$p(\boldsymbol{\pi}|\mathbf{x}, \mathcal{D}) \propto \prod_{i=1}^{N} p(\mathbf{y}|\boldsymbol{\pi})^{k(\mathbf{x}, \mathbf{x}_i)} p(\boldsymbol{\pi}|\mathbf{x}), \tag{4.13}$$

which can be evaluated in closed form for appropriate an appropriate likelihood and prior. Namely, for PROBE, we will assume Gaussian likelihoods for the reprojection errors (and therefore the observations $\mathbf{y}_{i,c_1}$), and inverse Wishart priors for covariance matrices (this will result in inverse Wishart posteriors due to conjugacy). The input, $\mathbf{x}$, will be the vector of predictors $\boldsymbol{\phi}$.

[2] i.e., $k(\mathbf{x}, \mathbf{x}) = 1 \forall \mathbf{x}$ and $k(\mathbf{x}, \mathbf{x}') \in [0, 1] \forall \mathbf{x}, \mathbf{x}'$.

4.4.3 Generalized Kernels for Visual Odometry

In order to exploit conjugacy to a Gaussian noise model, we formulate our prior knowledge about this function using an inverse Wishart (IW) distribution over positive definite $d \times d$ matrices (the IW distribution has been used as a prior on covariance matrices in other robotics and computer vision contexts, see for example, (Fitzgibbon et al., 2007)). This distribution is defined by a scale matrix $\mathbf{\Psi} \in \mathbb{R}^{d \times d} \succ 0$ and a scalar quantity called the degrees of freedom $\nu \in \mathbb{R} > d-1$:

$$\begin{aligned} p(\mathbf{R}) &= \mathrm{IW}(\mathbf{R}; \mathbf{\Psi}, \nu) \\ &= \frac{|\mathbf{\Psi}|^{\nu/2}}{2^{\frac{\nu d}{2}} \Gamma_d(\frac{\nu}{2})} |\mathbf{R}|^{-\frac{\nu+d+1}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\mathbf{\Psi}\mathbf{R}^{-1}\right)\right). \end{aligned} \tag{4.14}$$

We use the scale matrix to encode our prior estimate of the covariance, and the degrees of freedom to encode our confidence in that estimate. Specifically, if we estimate the covariance $\mathbf{R}$ associated with predictor $\boldsymbol{\phi}$ to be $\hat{\mathbf{R}}$ with a confidence equivalent to seeing n independent samples of the error from $\mathcal{N}(\mathbf{0}, \hat{\mathbf{R}})$, we would choose $\nu(\boldsymbol{\phi}) = n$ and $\mathbf{\Psi}(\boldsymbol{\phi}) = n\hat{\mathbf{R}}$. Given a sequence of observations and ground truth transformations,

$$\mathcal{D} = \{\mathcal{I}_t, \mathbf{T}_t\}, \quad t \in [1, N] \tag{4.15}$$

where

$$\mathcal{I}_t = \{\mathbf{y}_{i,c_0}, \mathbf{y}_{i,c_1}, \boldsymbol{\phi}_{i,t}\} \quad i \in [1, N_t], \tag{4.16}$$

we can use the procedure of generalized kernel estimation as described above to infer a posterior distribution over the covariance matrix $\mathbf{R}_*$ associated with some query predictor vector $\boldsymbol{\phi}_*$:

$$\begin{aligned} p(\mathbf{R}_*|\mathcal{D}, \boldsymbol{\phi}_*) &\propto \prod_{i,t} \mathcal{N}(\mathbf{e}_{i,t}|\mathbf{0}, \mathbf{R}_*)^{k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t})} \\ &\quad \times \mathrm{IW}(\mathbf{R}_*; \mathbf{\Psi}(\boldsymbol{\phi}_*), \nu(\boldsymbol{\phi}_*)) \qquad (4.17) \\ &= \mathrm{IW}(\mathbf{R}_*; \mathbf{\Psi}_*, \nu_*). \qquad (4.18) \end{aligned}$$

Here, $\mathbf{e}_{i,t} = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}_t\mathbf{f}^{-1}(\mathbf{y}_{i,c_0}))$ as before. The function $k : \mathbb{R}^M \times \mathbb{R}^M \to [0, 1]$ is a kernel function which measures the similarity of two points in predictor space. Note also that the posterior parameters $\mathbf{\Psi}_*$ and ν_* can be computed in closed form (see Vega-Brown et al. (2014)) as

$$\mathbf{\Psi}_* = \mathbf{\Psi}(\boldsymbol{\phi}_*) + \sum_{i,t} k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t}) \mathbf{e}_{i,t}\, \mathbf{e}_{i,t}{}^T, \tag{4.19}$$

$$\nu_* = \nu(\boldsymbol{\phi}_*) + \sum_{i,t} k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t}). \tag{4.20}$$

If we marginalize over the covariance matrix, we find that the posterior predictive distribution is a multivariate Student's t distribution:

$$p(\mathbf{y}_{i,c_1}|\mathbf{T}_t, \mathbf{y}_{i,c_0}, \mathcal{D}, \boldsymbol{\phi}_{i,t}) \tag{4.21}$$

$$= \int \mathrm{d}\mathbf{R}_{i,t} \mathcal{N}\left(\mathbf{e}_{i,t}; \mathbf{0}, \mathbf{R}_{i,t}\right) \mathrm{IW}(\mathbf{R}_{i,t}; \boldsymbol{\Psi}_*, \nu_*) \tag{4.22}$$

$$= \mathrm{t}_{\nu_*-d+1}\left(\mathbf{e}_{i,t}; \mathbf{0}, \frac{1}{\nu_* - d + 1}\boldsymbol{\Psi}_*\right) \tag{4.23}$$

$$= \frac{\Gamma(\frac{\nu_*+1}{2})}{\Gamma(\frac{\nu_*-d+1}{2})} |\boldsymbol{\Psi}_*|^{-\frac{1}{2}} \pi^{-\frac{d}{2}} \left(1 + \mathbf{e}_{i,t}{}^T \boldsymbol{\Psi}_*{}^{-1} \mathbf{e}_{i,t}\right)^{-\frac{\nu_*+1}{2}}. \tag{4.24}$$

Given a new landmark and predictor vector, we can infer a noise model by evaluating eqs. (4.19) and (4.20). In order to accelerate this computation, it is helpful to choose a kernel function with finite support: that is, $k(\boldsymbol{\phi}, \boldsymbol{\phi}') = 0$ if $\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|_2 > \gamma$ for some γ. Then, by indexing our training data in a spatial index such as a k-d tree, we can identify the subset of samples relevant to evaluating the sums in eqs. (4.19) and (4.20) in $\mathcal{O}(\log N + \log N_t)$ time. Algorithm 1 describes the procedure for building this model.

Algorithm 1 Build the covariance model given a sequence of observations, $\mathcal{D}$.

```
function BuildCovarianceModel(D)
    Initialize an empty spatial index M
    for all I_t, T_t in D do
        for all {y_{i,c0}, y_{i,c1}, φ_{i,c0}} in I_t do
            e_{i,t} = y_{i,c1} − f(T_t f^{-1}(y_{i,c0}))
            Insert φ_{i,t} into M and store e_{i,t} at its location
        end for
    end for
    return M
end function
```

Once we have inferred a noise model for each landmark in a new image pair, the maximum likelihood optimization problem is given by

$$\mathbf{T}_t^* = \underset{\mathbf{T}_t \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} (\nu_{i,t} + 1) \log\left(1 + \mathbf{e}_{i,t}{}^T \boldsymbol{\Psi}_{i,t}^{-1} \mathbf{e}_{i,t}\right). \tag{4.25}$$

The final optimization problem thus emerges as a nonlinear least squares problem with a rescaled Cauchy-like loss function, with error term $\mathbf{e}_{i,t}{}^T (\frac{1}{\nu_{i,t}+1}\boldsymbol{\Psi}_{i,t})^{-1} \mathbf{e}_{i,t}$ and outlier scale $\nu_{i,t} + 1$. This is a common robust loss function which is approximately quadratic in the reprojection error for $\mathbf{e}_{i,t}{}^T \boldsymbol{\Psi}_{i,t}^{-1} \mathbf{e}_{i,t} \ll \nu_{i,t} + 1$, but grows only logarithmically for $\mathbf{e}_{i,t}{}^T \boldsymbol{\Psi}_{i,t}^{-1} \mathbf{e}_{i,t} \gg \nu_{i,t} + 1$. It follows that in the limit of large $\nu_{i,t}$—in regions of predictor space where there are many relevant samples—our optimization problem becomes the original least-squares optimization problem.

Solving nonlinear optimization problems with the form of Equation (4.25) is a well-studied and

well-understood task, and software packages to perform this computation are readily available. Algorithm 2 describes the procedure for computing the transform between a new image pair, treating the optimization of Equation (4.25) as a subroutine.

Algorithm 2 Compute the transform between two images, given a set, $\mathcal{I}_t$, of landmarks and predictors extracted from an image pair and a covariance model $\mathcal{M}$.

function COMPUTETRANSFORM($\mathcal{I}_t$, $\mathcal{M}$)
  for all $\{\mathbf{y}_{i,c_0}, \mathbf{y}_{i,c_1}, \boldsymbol{\phi}_{i,c_0}\}$ in $\mathcal{I}_t$ **do**
    $\boldsymbol{\Psi}, \nu \leftarrow$ INFERNOISEMODEL($\mathcal{M}$, $\boldsymbol{\phi}_{i,t}$)
    $g(\mathbf{T}) = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0}))$
    $\mathcal{L} \leftarrow \mathcal{L} + (\nu + 1) \log\left(1 + g(\mathbf{T})^T \boldsymbol{\Psi}^{-1} g(\mathbf{T})\right)$
  end for
  return $\operatorname{argmin}_{\mathbf{T} \in \mathrm{SE}(3)} \mathcal{L}(\mathbf{T})$
end function
function INFERNOISEMODEL($\mathcal{M}$, $\boldsymbol{\phi}_*$)
  NEIGHBORS $\leftarrow$ GETNEIGHBORS($\mathcal{M}$, $\boldsymbol{\phi}_*$, ρ) ▷ ρ is the radius of support of kernel k
  $\boldsymbol{\Psi}_* \leftarrow \boldsymbol{\Psi}(\boldsymbol{\phi}_*)$
  $\nu_* \leftarrow \nu(\boldsymbol{\phi}_*)$
  for $(\boldsymbol{\phi}_{i,t}, \mathbf{e}_{i,t})$ in NEIGHBORS **do**
    $\boldsymbol{\Psi}_* \leftarrow \boldsymbol{\Psi}_* + k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t}) \mathbf{e}_{i,t}\, \mathbf{e}_{i,t}^T$
    $\nu_* \leftarrow \nu_* + k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t})$
  end for
  return $\boldsymbol{\Psi}_*, \nu_*$
end function

We observe that Algorithm 2 is predictively robust, in the sense that it uses past experiences not just to predict the reliability of a given image landmark, but also to introspect and estimate its own knowledge of that reliability. Landmarks which are not known to be reliable are trusted less than landmarks which look like those which have been observed previously, where "looks like" is defined by our prediction space and choice of kernel.

4.4.4 Inference without ground truth

Algorithm 1 requires access to the true transform between training image pairs. In practice, such ground truth data may be difficult to obtain. In these cases, we can instead formulate a likelihood model $p(\mathcal{D}'|\mathbf{T}_1, \ldots, \mathbf{T}_t)$, where $\mathcal{D}' = \{\mathcal{I}_t\}$ is a dataset consisting only of landmarks and predictors for each training image pair. We can construct a model for future queries by inferring the most likely sequence of transforms for our training images. The likelihood has the following factorized form:

$$p(\mathcal{D}'|\mathbf{T}_{1:T}) \propto \int \prod_{i,t} \mathrm{d}\mathbf{R}_{i,t}\, p(\mathbf{y}_{i,c_1}|\mathbf{y}_{i,c_0}, \mathbf{T}_t, \mathbf{R}_{i,t}) p(\mathbf{R}_{i,t}|\boldsymbol{\phi}_{i,t}, \mathcal{D}, \mathbf{T}_{1:T}). \tag{4.26}$$

We cannot easily maximize this likelihood, since marginalizing over the noise covariances removes the independence of the transforms between each image pair. To render the optimization tractable, we fol-

low previous work (Vega-Brown and Roy, 2013) and formulate an iterative expectation-maximization (EM) procedure. Given an estimate $\mathbf{T}_t^{(n)}$ of the transforms, we can compute the expected log-likelihood conditioned on our current estimate:

$$Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)}) = \int \left(\prod_{i,t} \mathrm{d}\mathbf{R}_{i,t}\, p(\mathbf{R}_{i,t}|\mathcal{D}_{\setminus i,t}, \mathbf{T}_{1:T}^{(n)}) \right) \log \prod_{i,t} p(\mathbf{y}_{i,c_1}|\mathbf{y}_{i,c_0}, \mathbf{T}_t, \mathbf{R}_{i,t}). \tag{4.27}$$

This has the effect of rendering the likelihood of each transform to be estimated independently. Moreover, the expected log-likelihood can be evaluated in closed form:

$$Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)}) \cong -\frac{1}{2} \sum_{t=1}^{T} \sum_{i=1}^{N_t} \mathbf{e}_{i,t}{}^T \left(\frac{1}{\nu_{i,t}^{(n)}} \mathbf{\Psi}_{i,t}^{(n)} \right)^{-1} \mathbf{e}_{i,t}. \tag{4.28}$$

The symbol $\cong$ is used to indicate equality up to an additive constant. We can iteratively refine our estimate by maximizing the expected log-likelihood

$$\mathbf{T}_{1:T}^{(n+1)} = \underset{\mathbf{T}_{1:T} \in \mathrm{SE}(3)^T}{\operatorname{argmax}} Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)}). \tag{4.29}$$

Due to the additive structure of $Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)})$, this takes the form of T separate nonlinear least-squares optimizations:

$$\mathbf{T}_t^{(n+1)} = \underset{\mathbf{T}_t \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} \mathbf{e}_{i,t}{}^T \left(\frac{1}{\nu_{i,t}^{(n)}} \mathbf{\Psi}_{i,t}^{(n)} \right)^{-1} \mathbf{e}_{i,t}. \tag{4.30}$$

Algorithm 3 describes the process of training a model without ground truth. We refer to this process as PROBE-GK-EM, and distinguish it from PROBE-GK-GT (Ground Truth). We note that the sequence of estimated transforms, $\mathbf{T}_{1:T}^{(n)}$, is guaranteed to converge to a local maxima of the likelihood function (Dempster et al., 1977). It is also possible to use a robust loss function (Equation (4.25)) in place of Equation (4.30) during EM training. Although not formally motivated by the derivation above, this approach often leads to lower test errors in practice. Characterizing when and why this robust learning process outperforms its non-robust alternative is outside the scope of this dissertation.

4.5 Prediction Space

A crucial component of our technique is the choice of the vector of predictors ϕ. In practice, feature tracking quality is often degraded by a variety of effects such as motion blur, moving objects, and textureless or self-similar image regions. The challenge is in determining predictors that account for such effects without requiring excessive computation. In our implementation, we use the following predictors, but stress that the choice of predictors can be tailored to suit particular applications and environments:

Algorithm 3 Build the covariance model without ground truth given a sequence of observations, $\mathcal{D}'$, and an initial odometry estimate $\mathbf{T}_{1:T}^{(0)}$.

function BUILDCOVARIANCEMODEL($\mathcal{D}'$, $\mathbf{T}_{1:T}^{(0)}$)
  Initialize an empty spatial index $\mathcal{M}$
  for all $\mathcal{I}_t$ in $\mathcal{D}'$ **do**
    for all $\{\mathbf{y}_{i,c_0}, \mathbf{y}_{i,c_1}, \boldsymbol{\phi}_{i,t}\}$ in $\mathcal{I}_t$ **do**
      $\mathbf{e}_{i,t} = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}_t^{(0)}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0}))$
      Insert $\boldsymbol{\phi}_{i,t}$ into $\mathcal{M}$ and store $\mathbf{e}_{i,t}$ at its location
    end for
  end for
  repeat
    for all $\mathcal{I}_t$ in $\mathcal{D}'$ **do**
      for all $\{\mathbf{y}_{i,c_0}, \mathbf{y}_{i,c_1}, \boldsymbol{\phi}_{i,t}\}$ in $\mathcal{I}_t$ **do**
        $\boldsymbol{\Psi}, \nu \leftarrow$ INFERNOISEMODEL($\mathcal{M}, \boldsymbol{\phi}_{i,t}$)
        $g(\mathbf{T}) = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0}))$
        $\mathcal{L} \leftarrow \mathcal{L} + g(\mathbf{T})^T \left(\frac{1}{\nu}\boldsymbol{\Psi}\right)^{-1} g(\mathbf{T})$
      end for
      $\mathbf{T}_t \leftarrow \operatorname{argmin}_{\mathbf{T}\in \mathrm{SE}(3)} \mathcal{L}(\mathbf{T})$
      $\mathbf{e}_{i,t} = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}_t^{(0)}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0}))$
      Update the error stored at $\boldsymbol{\phi}_{i,t}$ in $\mathcal{M}$ to $\mathbf{e}_{i,t}$
    end for
  until converged
  return $\mathcal{M}$
end function

- Angular velocity and linear acceleration magnitudes
- Local image entropy
- Blur (quantified by the blur metric of Crete et al. (2007))
- Optical flow variance score
- Image frequency composition

We discuss each of these predictors in turn.

4.5.1 Angular velocity and linear acceleration

While most of the predictors in our system are computed directly from image data, the magnitudes of the angular velocities and linear accelerations reported by an IMU (if available) are in themselves good predictors of image degradation (e.g., image blur) and hence poor feature tracking. We do not explicitly correct for bias in linear accelerations because we expect real motion-induced acceleration to trump bias at the timescales of our test trials. As a result, there is virtually no computational cost involved in incorporating these quantities as predictors.

4.5.2 Local image entropy

Entropy is a statistical measure of randomness that can be used to characterize the texture in an image or patch. Since the quality of feature detection is strongly influenced by the strength of the texture in the vicinity of the feature point, we expect the entropy of a patch centered on the feature to be a good predictor of its quality. We evaluate the entropy S in an image patch by sorting pixel intensities into N bins and computing

$$S = -\sum_{i=1}^{N} c_i \log_2(c_i), \tag{4.31}$$

where c_i is the number of pixels counted in the i^{th} bin.

4.5.3 Blur

Blur can arise from a number of sources including motion, dirty lenses, and sensor defects. All of these have deleterious effects on feature tracking quality. To assess the effect of blur in detail, we performed a separate experiment. We recorded images of 32 interior corners of a standard checkerboard calibration target using a low frame-rate (20 FPS) Skybotix VI-Sensor stereo camera and a high frame-rate (125 FPS) Point Grey Flea3 monocular camera rigidly connected by a bar (Figure 4.3). Prior to the experiment, we determined the intrinsic and extrinsic calibration parameters of our rig using the KALIBR[3] package Furgale et al. (2013). The apparatus underwent both slow and fast translational and rotational motion, which induced different levels of motion blur as quantified by the blur metric proposed by Crete et al. (2007).

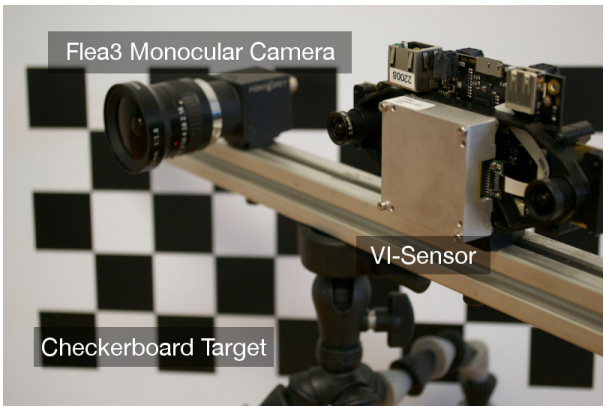


Figure 4.3: The Skybotix VI-Sensor, Point Grey Flea3, and checkerboard target used in our motion blur experiments.

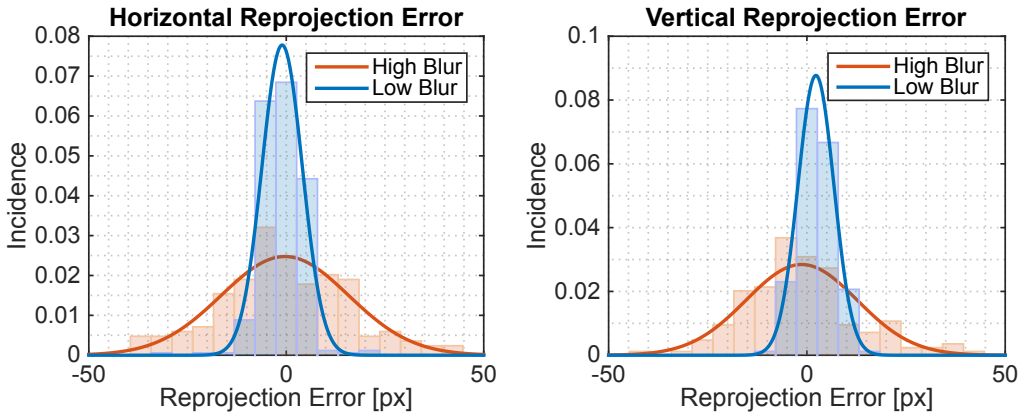


Figure 4.4: Reprojection error of checkerboard corners triangulated from the VI-Sensor and reprojected into the Flea3.We distinguish between high and low blur by thresholding the blur metric Crete et al. (2007).

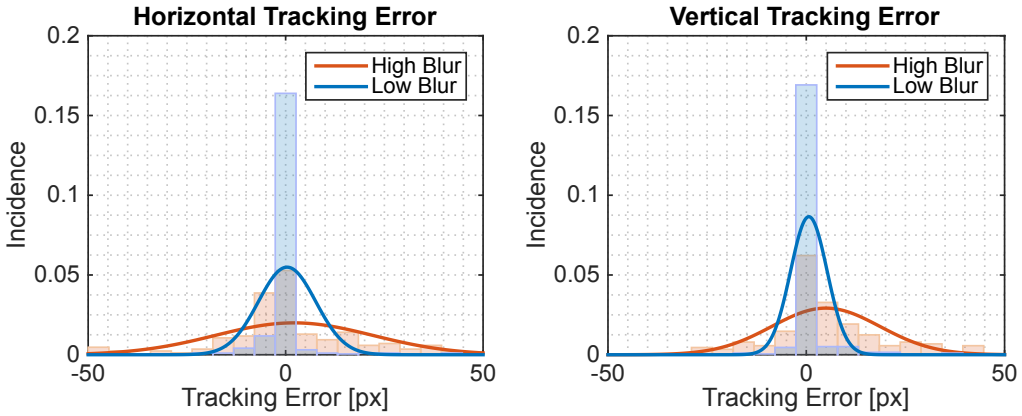


Figure 4.5: Effect of blur on reprojection and tracking error for the slow-then-fast checkerboard dataset. We distinguish between high and low blur by thresholding the blur metric Crete et al. (2007). The variance in both errors increases with blur.

We detected checkerboard corners in each camera at synchronized time steps, computed their 3D coordinates in the VI-Sensor frame, then reprojected these 3D coordinates into the Flea3 frame. We then computed the reprojection error as the distance between the reprojected image coordinates and the true image coordinates in the Flea3 frame. Since the Flea3 operated at a much higher frame rate than the VI-Sensor, it was less susceptible to motion blur and so we treated its observations as ground truth. We also computed a tracking error by comparing the image coordinates of checkerboard corners in the left camera of the VI-Sensor computed from both KLT tracking Lucas and Kanade (1981) and re-detection.

Figure 4.5 shows histograms and fitted normal distributions for both reprojection error and tracking error. From these distributions we can see that the errors remain approximately zero-mean, but that their variance increases with blur. This result is compelling evidence that the effect of blur on feature tracking quality can be accounted for by scaling the feature covariance matrix by a function of the blur metric.

4.5.4 Optical flow variance

To detect moving objects, we compute a score for each feature based on the ratio of the variance in optical flow vectors in a small region around the feature to the variance in flow vectors of a larger region. Intuitively, if the flow variance in the small region differs significantly from that in the larger region, we might expect the feature in question to belong to a moving object, and we would therefore like to trust the feature less. Since we consider only the variance in optical flow vectors, we expect this predictor to be reasonably invariant to scene geometry.

We compute this optical flow variance score according to

$$\log\left(\frac{\bar{\sigma}_s^2}{\bar{\sigma}_l^2}\right), \tag{4.32}$$

where $\bar{\sigma}_s^2, \bar{\sigma}_l^2$ are the means of the variance of the vertical and horizontal optical flow vector components in the small and large regions respectively. Figure 4.6 shows sample results of this scoring procedure for two images in the KITTI dataset. Our optical flow variance score generally picks out moving objects such as vehicles and cyclists in diverse scenes.

4.5.5 Image frequency composition

Reliable feature tracking is often difficult in textureless or self-similar environments due to low feature counts and false matches. We detect textureless and self-similar image regions by computing the Fast Fourier Transform (FFT) of each image and analyzing its frequency composition. For each feature, we compute a coefficient for the low- and high-frequency regimes of the FFT. Figure 4.7 shows the result of the high-frequency version of this predictor on a sample image from the KITTI dataset. Our high-frequency predictor effectively distinguishes between textureless regions (e.g., shadows and roads) and

[3] https://github.com/ethz-asl/kalibr

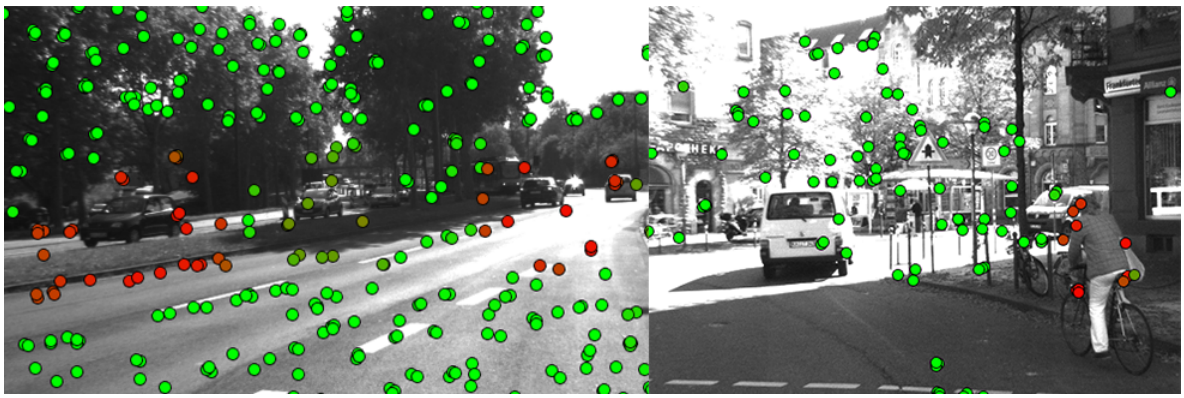


Figure 4.6: The optical flow variance predictor can help in detecting moving objects. Red circles correspond to higher values of the optical flow variance score (i.e., features more likely to belong to a moving object).

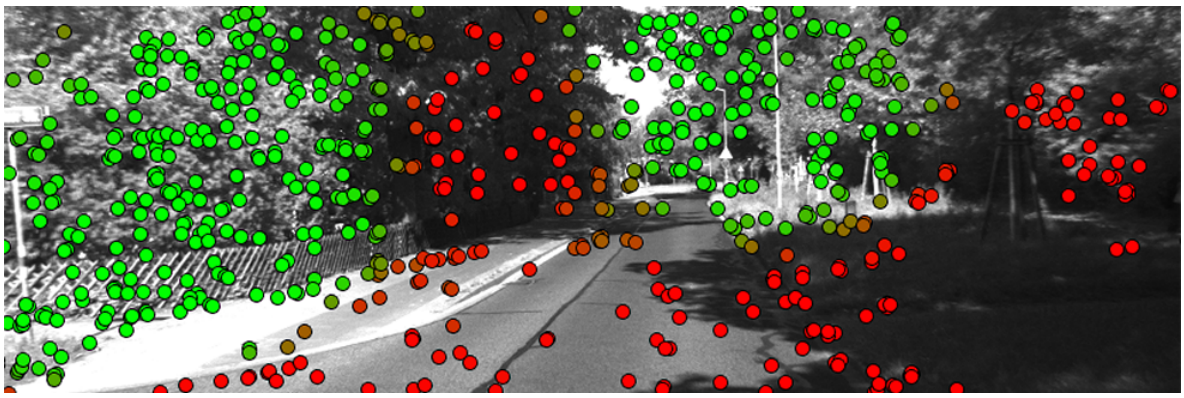


Figure 4.7: A high-frequency predictor can distinguish between regions of high and low texture such as foliage and shadows. Green indicates higher values.

texture-rich regions (e.g., foliage).

4.6 Experiments

To validate PROBE-GK, we used three types of data: synthetic simulations, the KITTI dataset, and our own experimental data collected at the University of Toronto.

4.6.1 Simulation

Monte-Carlo Verification

To begin, we verified that PROBE-GK can predict increasingly accurate estimates of the true error covariance as more training data is added. We developed a basic simulation environment consisting of a large amount of point landmarks being observed by a stereo camera. In our simulation, the camera traversed a single step in one direction, and recorded empirical reprojection errors based on ground truth poses. We simulated additive Gaussian noise on image coordinates, and used Monte Carlo simulations to estimate the true covariances. Figure 4.8 shows the mean Frobenius norm (as defined in

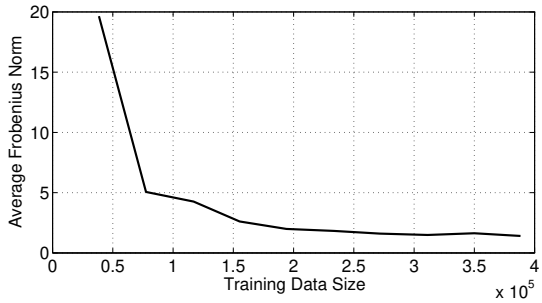


Figure 4.8: Mean Frobenius norm of the error between the estimated and true noise covariance as a function of training data size. The norm tends to zero as training data is added which indicates that PROBE-GK is learning the correct covariances.

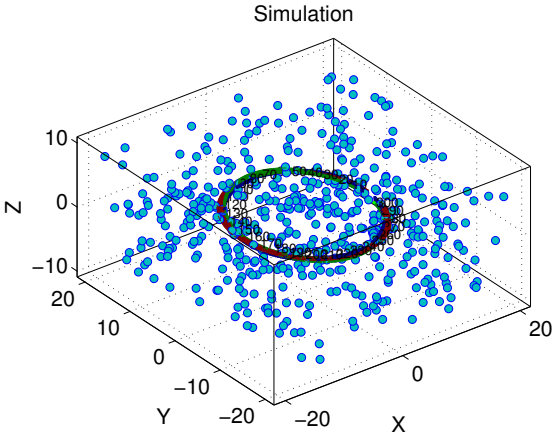


Figure 4.9: Our synthetic world. A stereo camera rig moves through a world with 2000 point features.

Barfoot and Furgale (2014)) between the covariances estimated by PROBE-GK and the true covariances for a test trial. The mean norm tends to zero as more landmarks are added, indicating that PROBE-GK does learn the correct covariances.

Synthetic

Next, we formulated a synthetic dataset wherein a stereo camera traverses a circular path observing 2000 randomly distributed point features. We added Gaussian noise to each of the ideal projected pixel co-ordinates for visible landmarks at every step. We varied the noise variance as a function of the vertical pixel coordinate of the feature in image space. In addition, a small subset of the landmarks received an error term drawn from a uniform distribution to simulate the presence of outliers. The prediction space was composed of the vertical and horizontal pixel locations in each of the stereo cameras.

We simulated independent training and test traversals, where the camera moved for 30 and 60 seconds respectively (at a forward speed of 3 metres per second for final path lengths of 90 and 180 meters). Figure 4.10 and Table 4.1 document the qualitative and quantitative comparisons of PROBE-GK (trained with and without ground-truth) against two baseline stereo odometry frameworks. Both

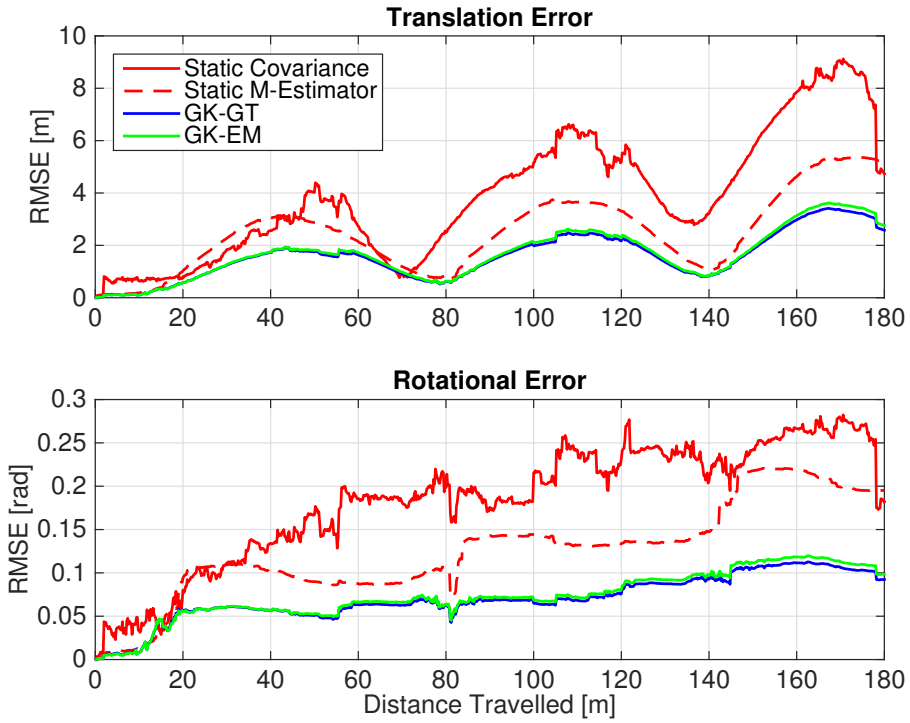


Figure 4.10: A comparison of translational and rotational Root Mean Square Error on simulated data (RMSE) for four different stereo-visual odometry pipelines: two baseline bundle adjustment procedures with and without a robust Student's t cost with a fixed and hand-tuned covariance and degrees of freedom (M-Estimation), a robust bundle adjustment with covariances learned from ground truth with algorithm 1 (GK-GT), and a robust bundle adjustment using covariances learned without ground truth using expectation maximization, with algorithm 3 (GK-EM). Note in this experiment, the RMSE curves for GK-GT and GK-EM very nearly overlap. The overall translational and rotational ARMSE values are shown in Table 4.1.

baseline estimators were implemented based on the reprojection-error-based VO pipeline described in Chapter 3. The first utilized fixed covariances for all reprojection errors, while the second used a modified robust cost (i.e. M-estimation) based on Student's t weighting, with $\nu = 5$ (as suggested in Kerl et al. (2013)). These benchmarks served as baseline estimators (with and without robust costs) that used fixed covariance matrices and did not include a predictive component.

Using PROBE-GK with ground truth data for training, we significantly reduced both the translation and rotational Average Root Mean Squared Error (ARMSE) by approximately 50%. In our synthetic data, the Expectation Maximization approach was able to achieve nearly identical results to the ground-truth-aided model within 5 iterations.

4.6.2 KITTI

To evaluate PROBE-GK on real environments, we trained and tested several models on the KITTI Vision Benchmark suite (Geiger et al., 2013), a series of datasets collected by a car outfitted with a number of sensors driven around different parts of Karlsruhe, Germany. Within the dataset, ground truth pose information is provided by a high grade inertial navigation unit which also fuses measurements from differential GPS. Raw data is available for different types of environments through which the car was

Figure 4.11: The KITTI dataset contains three different environments. We validate PROBE-GK by training on each type and testing against a baseline stereo visual odometry pipeline.

Table 4.1: Comparison of average root mean squared errors (ARMSE) for rotational and translational components. Each trial is trained and tested from a particular category of raw data from the synthetic and KITTI datasets.

		Trans. ARMSE [m]				Rot. ARMSE [rad]			
	Length [m]	Fixed Covar.	Static M-Estimator	GK-GT	GK-EM	Fixed Covar.	Static M-Estimator	GK-GT	GK-EM
Synthetic	180	3.87	2.49	1.59	1.66	0.18	0.13	0.070	0.073
City	332.9	3.84	2.99	1.69	2.87	0.032	0.021	0.0046	0.018
Residential	714.1	13.48	9.37	1.97	8.80	0.068	0.050	0.013	0.044
Road	723.8	17.69	9.38	5.24	8.87	0.060	0.027	0.015	0.024

driving; for our work, we focused on the city, residential and road categories (Figure 4.11). From each category, we chose two separate trials for training and testing.

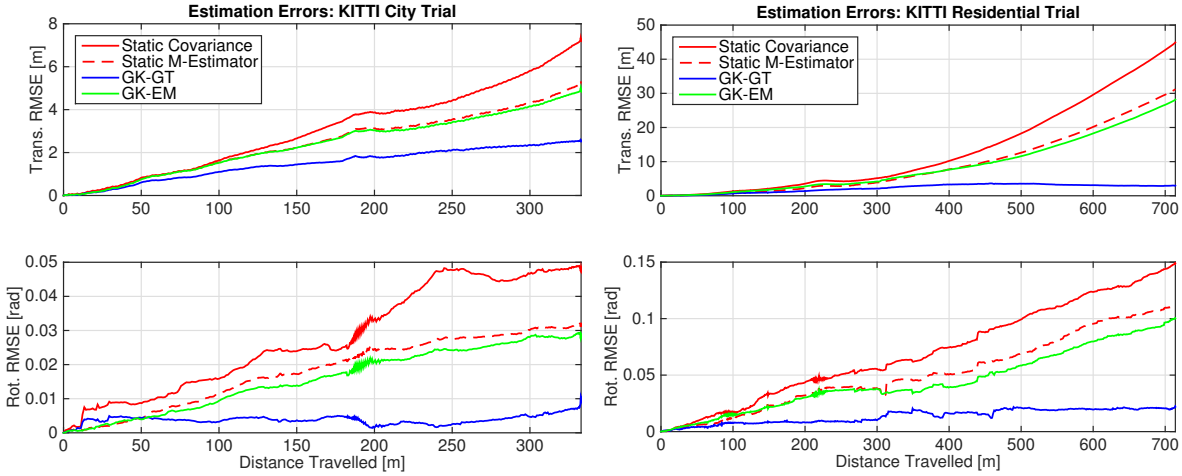
Figures 4.12a to 4.12c show typical results; Table 4.1 presents a quantitative comparison. PROBE GK-GT produced significant reductions in ARMSE, reducing translational ARMSE by as much as 80%. In contrast, GK-EM showed more modest improvements; this is unlike our synthetic experiments, where both GK-EM and GK-GT achieved similar performance. We note that although our simulated data is drawn from a mixture of Gaussian distributions, the underlying noise distribution for real data may be far more complex. With no ground truth, EM has to jointly optimize the camera poses and sensor uncertainty. It is unclear whether this is feasible in the general case with no ground truth information.

Further, we observe that the performance of PROBE-GK depends on the similarity of the training data to the final test trials. A characteristic training dataset was important for consistent improvements on test trials.

4.6.3 UTIAS

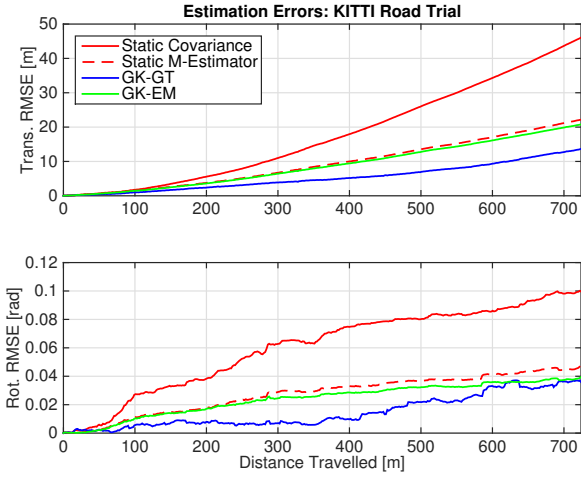
To further investigate the capability of our EM approach, we evaluated PROBE-GK on experimental data collected at the University of Toronto Institute for Aerospace Studies (UTIAS). For this experiment, we drove a Clearpath Husky rover outfitted with an Ashtech DG14 Differential GPS, and a PointGrey XB3 stereo camera around the MarsDome (an indoor Mars analog testing environment) at UTIAS (Figure 4.13a) for five trials of a similar path. Each trial was approximately 250 m in length and we made an effort to align the start and end points of each loop. We used the wide baseline (25 cm) of the XB3 stereo camera to record the stereo images. The approximate trajectory for all 5 trials, as recorded by GPS, is shown in Figure 4.13b. Note that the GPS data was not used during training, and only recorded for reference.

For the prediction space in our experiments, we mimicked the KITTI experiments, omitting inertial magnitudes as no inertial data was available. We trained PROBE-GK without ground truth, using the



(a) City category.

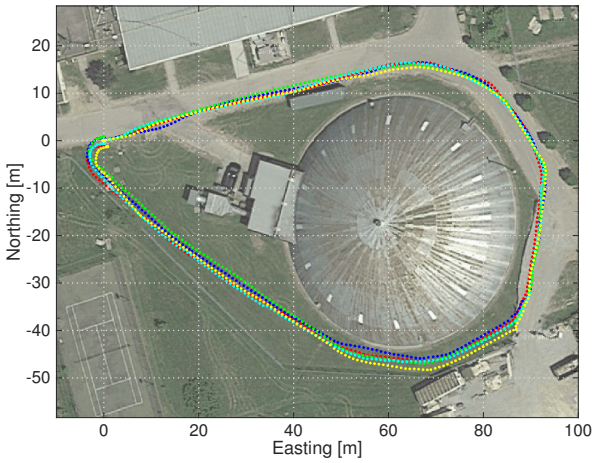
(b) Residential category.



(c) Road category.

Figure 4.12: RMSE comparison of stereo VO estimators evaluated on data in the KITTI odometry benchmark. See Table 4.1 for a quantitative summary.

(a) Our experimental apparatus: a Clearpath Husky rover outfitted with a PointGrey XB3 stereo camera and a differential GPS receiver and base station.



(b) GPS ground truth for 5 experimental trials collected near the UTIAS Mars Dome. Each trial is approximately 250 m long.

Figure 4.13: Our experimental setup.

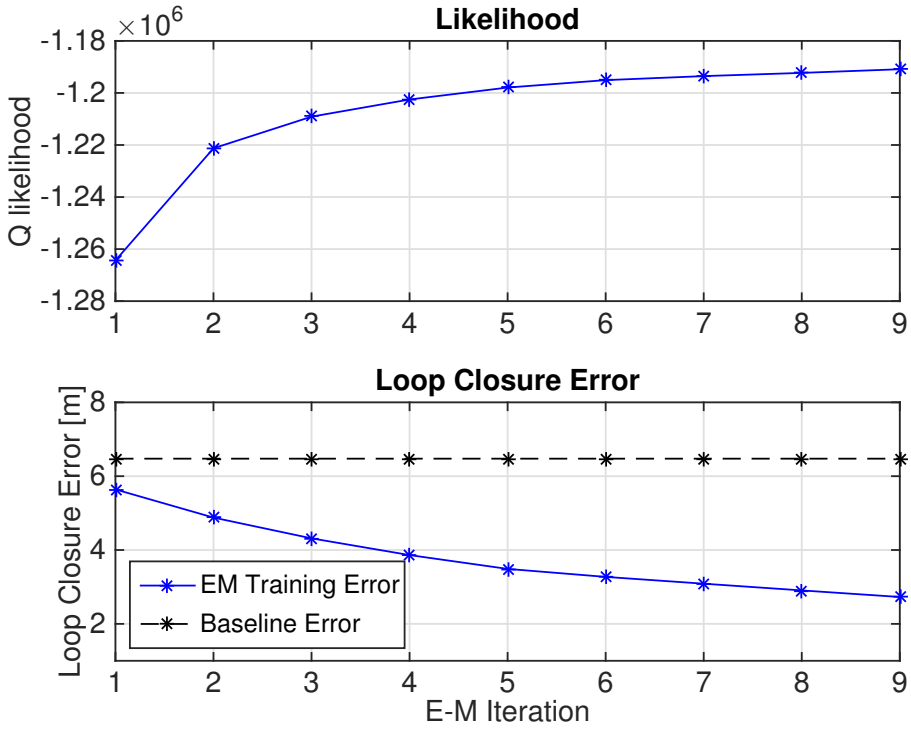


Figure 4.14: Training without ground truth using PROBE-GK-EM on a 250.2m path around the Mars Dome at UTIAS. The likelihood of the data increases with each iteration, and the loop closure error decreases, improving significantly from a baseline static M-estimator.

Table 4.2: Comparison of loop closure errors for 4 different experimental trials with and without a learned PROBE-GK-EM model.

		Loop Closure Error [m]	
Trial	Path Length [m]	PROBE-GK-EM	Static M-Estimator
2	250.3	3.88	8.07
3	250.5	3.07	6.64
4	205.4	2.81	7.57
5	249.9	2.34	7.75

Expectation Maximization approach. Figure 4.14 shows the likelihood and loop closure error as a function of EM iteration.

The EM approach indeed produced significant error reductions on the training dataset after just a few iterations. Although it was trained with no ground truth information, our PROBE-GK model was used to produce significant reductions in the loop closure errors of the remaining 4 test trials. This reinforced our earlier hypothesis: the EM method works well when the training trajectory more closely resembles the test trials (as was the case in this experiment). Table 4.2 lists the statistics for each test.

4.7 Summary

Predictive Robust Estimation (PROBE) applied Generalized Kernel estimation to improve on the uncorrelated and static Gaussian error models typically employed in stereo odometry. By building a nonparametric predictive model for the density of reprojection errors, we derived a robust least squares objective whose parameters were predicted based on training data. In summary, this chapter contributed

1. a probabilistic model for indirect stereo visual odometry, leading to a predictive robust algorithm for inference on that model,
2. an efficient approach to constructing the robust algorithm based on Generalized Kernel (GK) estimation,
3. a procedure for training our model using pairs of stereo images with known relative transforms, and
4. an iterative, expectation-maximization approach to train our GK model when the relative ground truth egomotion was unavailable.

Chapter 5

Learned Probabilistic Sun Sensor

> He stepped down, avoiding any long look at her as one avoids long looks at the sun, but seeing her as one sees the sun, without looking.
>
> Leo Tolstoy, *Anna Karenina*

5.1 Introduction

Building on the results of PROBE, wherein we showed that it is possible to extract useful uncertainty information from images, we turn to the problem of constructing a pseudo-sensor that can infer a concrete geometric quantity that could be used to improve egomotion estimates. Specifically, in this chapter, we address the problem of replacing a costly dedicated hardware-based sun-sensor with a *sun pseudo-sensor*. We show, through extensive experiments, that it is possible to train a deep parametric model (a Bayesian Convolutional Neural Network, or BCNN) to act as a virtual sun sensor that can predict the direction of the sun from a single RGB image where the sun is not directly observed. Further, we leverage recent advances in the theory of approximate variational inference of deep networks to extract uncertainty estimates for each prediction.

Remark (*Associated Publications*). This project was a collaboration with Lee Clement. Broadly, Lee lead the integration of sun information into a visual odometry pipeline, while I lead the design and implementation of the learning architecture. It is associated with three publications. The first publication was exploratory work on (non-probabilistic) virtual sun sensors lead by Lee,

1. Clement, L., Peretroukhin, V., and Kelly, J. (2017). Improving the accuracy of stereo visual odometry using visual illumination estimation. In Kulic, D., Nakamura, Y., Khatib, O., and Venture, G., editors, *2016 International Symposium on Experimental Robotics*, volume 1 of *Springer Proceedings in Advanced Robotics*, pages 409–419. Springer International Publishing, Berlin Heidelberg.

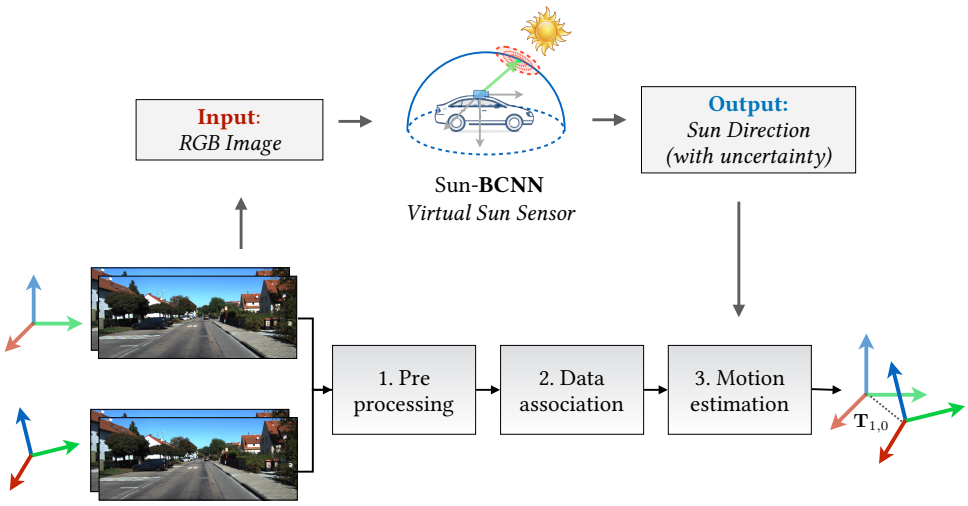


Figure 5.1: Sun-BCNN is a learned virtual sun sensor that outputs sun direction with an associated uncertainty based on a single RGB image. We use this as a source of orientation information within a privileged reference frame.

Invited to Journal Special Issue,

while the latter two publications were a joint collaboration where I developed and investigated the BCNN formulation and we collaborated on the experimental validation,

2. 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 IEEE International Conference on Robotics and Automation (ICRA'17)*, pages 2035–2042, Singapore,

3. 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.

This chapter is largely a reproduction of the latter journal publication which summarizes the approach, with additional elaboration on the pseudo-sensor architecture.

5.2 Motivation

This accumulated error, or drift, of egomotion estimation can be limited by incorporating external global information such as observations of known landmarks, loop closures, or position updates from the global navigation satellite system (GNSS). In many situations, however, a globally consistent map may be unavailable or prohibitively expensive to compute, loop closures may not occur, or GNSS information may be unavailable or inaccurate (e.g., on the surface of Mars). In such cases, outdoor navigation can still extract external information from measurements of celestial objects (c.f. early maritime navigation described in Chapter 1). During the day, for example, observations of the sun can

provide global orientation information since the sun's apparent motion in the sky is well-described by ephemeris models.

For visual odometry, 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). Accordingly, sun-based orientation corrections have been successfully used to improve VO estimates 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 motion estimation pipeline can significantly reduce VO drift over long trajectories.

In this work, we build a virtual pseudo-sensor that can replicate the performance of a dedicated sun-sensor while requiring only the image stream that is already being used to compute VO. In particular, we leverage recent advances in approximate variational inference of Bayesian Convolutional Neural Networks to demonstrate how we can build and train a deep model capable of inferring the direction of the sun from a single RGB image, and moreover, produce a covariance estimate for each observation.

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 is an adapted version of the pipeline describe in Chapter 3, 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 variety of environments. These include experiments on 21.6 km of urban driving data from the KITTI odometry benchmark training set (Geiger et al., 2013), as well as a 10 km traverse through a planetary analogue site taken from the Devon Island Rover Navigation Dataset collected in a planetary analogue site in the Canadian High Arctic (Furgale et al., 2012). Then, 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., 2016). Finally, we 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.

5.3 Related Work

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., 2017). Some of these methods

have been based on hand-crafted illumination cues such as shadows and variation in sky brightness (Lalonde et al., 2011; Clement et al., 2017), while others have attempted to learn such cues from data using deep Convolutional Neural Networks (CNNs) (Ma et al., 2016).

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., 2016; 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 (2016b) showed that it is possible to achieve 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 (Kendall et al., 2015) on camera pose regression.

We build on on previous work by Clement et al. (2017), who demonstrated empirically that techniques for single-image sun estimation based on hand-crafted models (Lalonde et al., 2011) and Convolutional Neural Networks (CNNs) (Ma et al., 2016) could be incorporated into a stereo visual odometry pipeline to reduce estimation error in the manner of Lambert et al. (2012). We also build on the work of Peretroukhin et al. (2017), who presented preliminary experimental results comparing Sun-BCNN against the method of Lalonde et al. (2011) and its VO-informed variant (Clement et al., 2017) as well as the Sun-CNN of Ma et al. (2016) on the KITTI odometry benchmark (Geiger et al., 2013), both in terms of raw measurement accuracy and in terms of their impact on VO accuracy.

While our method is similar in spirit to the work of Ma et al. (2016), who built a CNN-based sun sensor as part of a relocalization pipeline, our model makes three important improvements: 1) in addition to a point estimate of the sun direction, we output a principled covariance estimate that is incorporated into our estimator; 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. (2016)); and 3) we incorporate the sun direction covariance into a VO estimator that accounts for growth in pose uncertainty over time (unlike Clement et al. (2017)). Furthermore, our Bayesian CNN includes a dropout layer after every convolutional and fully connected layer (as outlined by Gal and Ghahramani (2016b) but not done by Kendall and Cipolla (2016)).

5.4 Sun-Aided Stereo Visual Odometry

For this work, we adopt a sliding window sparse stereo VO technique (derived from the frame-to-frame pipeline described in Chapter 3) 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 $\mathrm{SE}(3)$ 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 GPS ground truth pose such that $\underrightarrow{\mathcal{F}}_0$ is a local East-North-Up (ENU) coordinate system with its origin at the first GPS position.

5.4.1 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 3.1.2).

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 $\mathbf{g}(\cdot)$ (defined with disparity, unlike the function $\mathbf{f}(\cdot)$ in Section 3.1.2) can then be formulated as

$$\mathbf{y}_{k,j} = \mathbf{g}\left(\mathbf{p}_k^j\right) = \begin{bmatrix} u \\ v \\ d \end{bmatrix} = \begin{bmatrix} f_u p_{k,x}^j / p_{k,z}^j + c_u \\ f_v p_{k,y}^j / p_{k,z}^j + c_v \\ f_u b / p_{k,z}^j \end{bmatrix}, \tag{5.1}$$

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

5.4.2 Sliding Window Bundle Adjustment

Like with PROBE, we use the open-source `viso2` package (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 interframe 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 using the nonlinear least-squares solver Ceres (Agarwal et al., 2016). The objective function to be minimized can be written as

$$\mathcal{J} = \mathcal{J}_{\text{reprojection}} + \mathcal{J}_{\text{prior}}, \tag{5.2}$$

where

$$\mathcal{J}_{\text{reprojection}} = \sum_{k=k_1}^{k_2} \sum_{j=1}^{J} \mathbf{e}_{\mathbf{y}_{k,j}}^T \mathbf{R}_{\mathbf{y}_{k,j}}^{-1} \mathbf{e}_{\mathbf{y}_{k,j}} \tag{5.3}$$

and

$$\mathcal{J}_{\text{prior}} = \mathbf{e}_{\hat{\mathbf{T}}_{k_1,0}}^T \mathbf{R}_{\hat{\mathbf{T}}_{k_1,0}}^{-1} \mathbf{e}_{\hat{\mathbf{T}}_{k_1,0}}. \tag{5.4}$$

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 $\hat{\mathbf{y}}_{k,j} = \mathbf{g}\left(\hat{\mathbf{T}}_{k,0}\hat{\mathbf{p}}_0^j\right)$, 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 $\mathrm{SE}(3)$ matrix logarithm as $\mathbf{e}_{\check{\mathbf{T}}_{k_1,0}} = \log\left(\check{\mathbf{T}}_{k_1,0}^{-1}\hat{\mathbf{T}}_{k_1,0}\right)^\vee \in \mathbb{R}^6$. 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. 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.

5.5 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 GPS and use them to determine the global direction of the sun $\mathbf{s}_0$, expressed as a 3D unit vector, from ephemeris data. We define the world frame $\underrightarrow{\mathcal{F}}_0$ to be a local ENU coordinate system with the initial GPS 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 $\underrightarrow{\mathcal{F}}_k$ 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 (5.2)):

$$\mathcal{J} = \mathcal{J}_{\text{reprojection}} + \mathcal{J}_{\text{prior}} + \mathcal{J}_{\text{sun}}, \tag{5.5}$$

where

$$\mathcal{J}_{\text{sun}} = \sum_{k=k_1}^{k_2} \mathbf{e}_{\mathbf{s}_k}^T \mathbf{R}_{\mathbf{s}_k}^{-1} \mathbf{e}_{\mathbf{s}_k}, \tag{5.6}$$

and $\mathcal{J}_{\text{reprojection}}$ and $\mathcal{J}_{\text{prior}}$ are defined in Equations (5.3) and (5.4), 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 define $\mathbf{f}(\cdot)$ to be a function that transforms a 3D unit vector in camera frame $\underrightarrow{\mathcal{F}}_k$ to a zenith-azimuth parametrization:

$$\begin{bmatrix}\theta \\ \phi\end{bmatrix} = \mathbf{f}(\mathbf{s}_k) = \begin{bmatrix}\text{acos}(-s_{k,y}) \\ \text{atan2}(s_{k,x}, s_{k,z})\end{bmatrix} \tag{5.7}$$

where $\mathbf{s}_k = \begin{bmatrix}s_{k,x} & s_{k,y} & s_{k,z}\end{bmatrix}^T$. We can then define the term $\mathbf{e}_{\mathbf{s}_k} = \mathbf{f}(\hat{\mathbf{s}}_k) - \mathbf{f}(\mathbf{s}_k)$ to be the error in the predicted sun direction, expressed in azimuth-zenith coordinates, and $\mathbf{R}_{\mathbf{s}_k}$ to be the covariance of these errors. While $\mathbf{R}_{\mathbf{s}_k}$ would generally be treated as an empirically determined static covariance, in our approach we use the per-observation covariance computed using Equation (5.29), which allows us to weight each observation individually according to a measure of its intrinsic quality. 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.

5.6 Bayesian Convolutional Neural Networks

Recall from Section 2.6.4 that dropout (Figure 5.2) is a stochastic regularization technique that was originally designed to improve the generalization performance of deep networks. Dropout works by randomly 'turning off' certain inputs of layers to simulate an ensemble of smaller networks during training. Gal and Ghahramani (2016a) identified a link between applying dropout at test-time given a network trained with dropout (through a technique called *Monte Carlo dropout*, or *MC dropout*) and approximate variational inference for a particular type of Bayesian Neural Network. In this work, we leverage this insight to build a Bayesian Convolutional Neural Network through MC dropout. To elucidate the intuition behind this technique, we first review variational inference and then outline its connection to the technique of dropout.

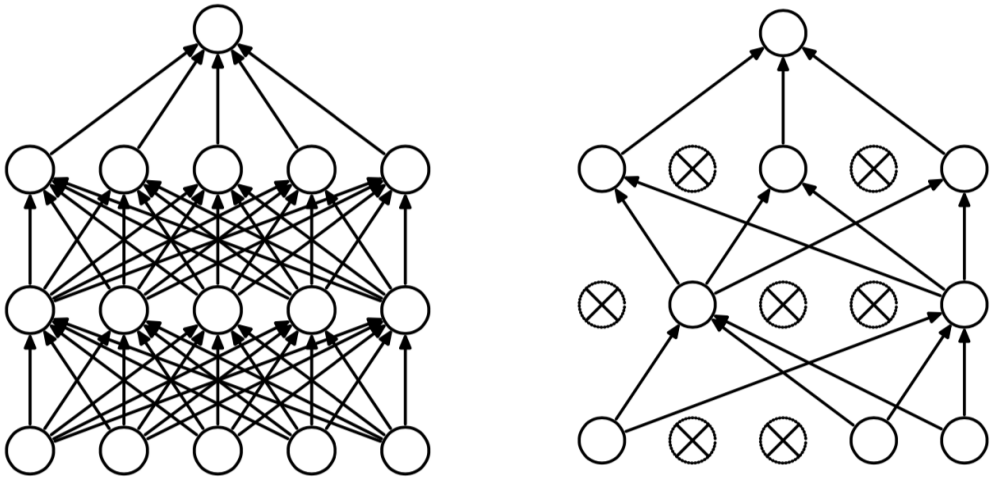
5.6.1 Bayesian Modelling and Variational Inference

Consider the goal of generating a function $f(\mathbf{x}; \boldsymbol{\pi})$ that maps inputs $\mathbf{x}$ to outputs $\mathbf{y}$ as a function of parameters $\boldsymbol{\pi}$. Recall from Chapter 4 that in Bayesian inference we define a *prior* on the space of parameters $p(\boldsymbol{\pi})$ and a *likelihood function* that defines how likely we are to observe $\mathbf{y}$ given an input $\mathbf{x}$ and parameters $\boldsymbol{\pi}$, $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\pi})$. For example, in regression problems, we often define the Gaussian likelihood

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\pi}) = \mathcal{N}(f(\mathbf{x}; \boldsymbol{\pi}), \tau^{-1}\mathbf{1}) \tag{5.8}$$

where τ is a model *precision*. Given a dataset $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\} = \{\{\mathbf{x}_1, ...\mathbf{x}_N\}, \{\mathbf{y}_1, ..., \mathbf{y}_N)\}\}$, the Bayesian formulation seeks a posterior over the parameters $\boldsymbol{\pi}$:

$$p(\boldsymbol{\pi}|\mathcal{D}) = \frac{p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\pi})p(\boldsymbol{\pi})}{p(\mathbf{Y}|\mathbf{X})} \tag{5.9}$$



(a) Standard fully-connected neural network.

(b) A neural network with dropout applied.

Figure 5.2: The technique of *dropout* stochastically removes the contribution of certain neurons to regularize learning. Figures from Srivastava et al. (2014).

where we may think of this posterior as evaluating the sentence 'the likelihood of the observed outputs given a particular set of parameters (weighted by a prior) compared to how likely they are in general'. This latter 'in general' quantity is the marginal likelihood (also known as the model *evidence*) over the space of parameters,

$$p(\mathbf{Y}|\mathbf{X}) = \int_{\boldsymbol{\pi}} p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\pi}) p(\boldsymbol{\pi}) d\boldsymbol{\pi}. \tag{5.10}$$

For a given test input $\mathbf{x}^*$, we can then *infer* a predictive distribution over the space of outputs by integrating over the posterior of model parameters:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) = \int_{\boldsymbol{\pi}} p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\pi}) p(\boldsymbol{\pi}|\mathcal{D}) d\boldsymbol{\pi}. \tag{5.11}$$

For most problems, evaluating the posterior (Equation (5.9)) analytically is not feasible. In such cases, we can turn to the technique of *variational inference* (Gal, 2016).

Variational Inference

In variational inference, we define a simpler variational density $q(\boldsymbol{\pi}; \boldsymbol{\theta})$ based on the auxiliary parameters $\boldsymbol{\theta}$. We then seek to minimize the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) between $q(\boldsymbol{\pi}; \boldsymbol{\theta})$ and the true posterior $p(\boldsymbol{\pi}|\mathcal{D})$:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \, \mathrm{KL}(q(\boldsymbol{\pi}; \boldsymbol{\theta}) || p(\boldsymbol{\pi}|\mathcal{D})) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \int q(\boldsymbol{\pi}; \boldsymbol{\theta}) \log \frac{q(\boldsymbol{\pi}; \boldsymbol{\theta})}{p(\boldsymbol{\pi}|\mathcal{D})} d\boldsymbol{\pi}, \tag{5.12}$$

which then allows us to compute the approximate predictive distribution as

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathcal{D}) \approx \int_{\boldsymbol{\pi}} p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\pi}) q(\boldsymbol{\pi}; \boldsymbol{\theta}^*) d\boldsymbol{\pi}. \tag{5.13}$$

One can show (see, for example, Gal (2016)) that minimizing KL divergence (Equation (5.12)) is the same as maximizing the evidence lower bound (ELBO)[1]

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} -\mathcal{L}_{\text{ELBO}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \int_{\boldsymbol{\pi}} \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\pi}) q(\boldsymbol{\pi}; \boldsymbol{\theta}) d\boldsymbol{\pi} - \text{KL}(q(\boldsymbol{\pi}; \boldsymbol{\theta}) \,||\, p(\boldsymbol{\pi})). \tag{5.14}$$

Intuitively, maximizing the first term above (referred to as the expected log likelihood) ensures that $q(\boldsymbol{\pi}; \boldsymbol{\theta})$ explains the dataset well, while minimizing the latter term regularizes the optimization so that $q(\boldsymbol{\pi}; \boldsymbol{\theta})$ does not stray too far away from a prior.

5.6.2 Monte Carlo Dropout as Approximate Variational Inference

A key insight of Gal (2016); Gal and Ghahramani (2016b) was that the training objective of a neural network with dropout resembles the ELBO optimization of Equation (5.14) with a careful choice of $q(\boldsymbol{\pi}; \boldsymbol{\theta})$. Namely, consider training a neural network with dropout and with another form of regularization called *weight decay* (which penalizes large weight magnitudes). The loss function for supervised training in this scenario can be written as

$$\mathcal{L}_{\text{dropout}}(\boldsymbol{\pi}) = \underbrace{\frac{1}{N} \sum_{i=1}^{N} \mathcal{E}(\mathbf{y}_i, \tilde{\mathbf{y}}_i)}_{\text{error function}} + \underbrace{\lambda \sum_{\ell=1}^{L} \left(\|\mathbf{W}_\ell\|_2^2 + \|\mathbf{b}_\ell\|_2^2 \right)}_{\text{weight decay}}, \tag{5.15}$$

where $\mathbf{y}_i$ is the target output, λ is a scalar weight decay hyper-parameter, and $\tilde{\mathbf{y}}_i = NN(\mathbf{x}_i)$ is a stochastic sample of the network output with dropout applied. Now if we define $q(\boldsymbol{\pi})$ to be a *Bernoulli variational distribution* over the network parameters as the re-parametrized density,

$$q(\boldsymbol{\pi}; \boldsymbol{\theta}) = \{\text{diag}(\{\epsilon_{i\ell}\}) \mathbf{W}_\ell, \mathbf{b}_\ell\}_{\ell=1}^{L} \tag{5.16}$$

$$\epsilon_{i\ell} \sim \text{Bernoulli}(p) \tag{5.17}$$

where the auxiliary variables become the network parameters themselves, $\boldsymbol{\theta} = \{\mathbf{W}_\ell, \mathbf{b}_\ell\}_{\ell=1}^{L}$. We can show an equivalence between optimizing the variational inference ELBO loss function,

$$\mathcal{L}_{\text{ELBO}}(\boldsymbol{\theta}) = -\int_{\boldsymbol{\pi}} \log p(\mathbf{Y}|\mathbf{X}, \boldsymbol{\pi}) q(\boldsymbol{\pi}; \boldsymbol{\theta}) d\boldsymbol{\pi} + \text{KL}(q(\boldsymbol{\pi}; \boldsymbol{\theta}) \,||\, p(\boldsymbol{\pi})) \tag{5.18}$$

$$= -\sum_{i=1}^{N} \int_{\boldsymbol{\pi}} \log p(\mathbf{y}_i|\mathbf{x}_i, \boldsymbol{\pi}) q(\boldsymbol{\pi}; \boldsymbol{\theta}) d\boldsymbol{\pi} + \text{KL}(q(\boldsymbol{\pi}; \boldsymbol{\theta}) \,||\, p(\boldsymbol{\pi})) \tag{5.19}$$

[1] KL divergence and the ELBO differ by a constant that is not dependent on the auxiliary variables.

and optimizing Equation (5.15) above as follows. First, the first term above can be approximated through Monte Carlo integration with a single sample[2] $\tilde{\boldsymbol{\pi}}$ of $q(\boldsymbol{\pi};\boldsymbol{\theta})$ such that,

$$= -\sum_{i=1}^{N} \log p(\mathbf{y}_i|\mathbf{x}_i, \tilde{\boldsymbol{\pi}}) + \mathrm{KL}(q(\boldsymbol{\pi};\boldsymbol{\theta}) \,||\, p(\boldsymbol{\pi})). \tag{5.20}$$

Second, it is possible to show that there exist a Gaussian prior $p(\boldsymbol{\pi})$ such that the condition (Eq. 3.12, Gal (2016))

$$\frac{\partial \mathrm{KL}(q(\boldsymbol{\pi};\boldsymbol{\theta}) \,||\, p(\boldsymbol{\pi}))}{\partial \boldsymbol{\theta}} \approx \frac{\partial \lambda \sum_{\ell=1}^{L} \left(\|\mathbf{W}_\ell\|_2^2 + \|\mathbf{b}_\ell\|_2^2 \right)}{\partial \boldsymbol{\theta}} \tag{5.21}$$

holds. Finally, if we interpret $-\log p(\mathbf{y}_i|\mathbf{x}_i, \tilde{\boldsymbol{\pi}})$ as $\mathcal{E}(\mathbf{y}_i, \tilde{\mathbf{y}}_i)$ then it follows that

$$\frac{\partial \mathcal{L}_{\text{ELBO}}}{\partial \boldsymbol{\theta}} \propto \frac{\partial \mathcal{L}_{\text{Dropout}}}{\partial \boldsymbol{\theta}} \tag{5.22}$$

where the proportional constant can be written down analytically (Gal, 2016). Thus the two objectives result in the same optimal parameters!

Computing Moments

The results above indicate that, once trained, a neural network with dropout gives us access to a predictive distribution over outputs $\mathbf{y}$. An unbiased estimator for its mean and covariance are given by applying dropout at test time and computing statistics over T samples:

$$\mathbb{E}\left[\mathbf{y}^*\right] = \frac{1}{T} \sum_{t=1}^{T} \tilde{\mathbf{y}}_t^*, \tag{5.23}$$

where $\tilde{\mathbf{y}}_t^* = NN(\mathbf{x}^*; \tilde{\boldsymbol{\pi}}_t)$ and

$$\mathrm{Var}(\mathbf{y}^*) = \tau^{-1}\mathbf{1} + \frac{1}{T} \sum_{t=1}^{T} \tilde{\mathbf{y}}_t^* (\tilde{\mathbf{y}}_t^*)^T - \mathbb{E}\left[\mathbf{y}^*\right] \mathbb{E}\left[\mathbf{y}^*\right]^T, \tag{5.24}$$

where the precision τ defines the likelihood

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\pi}) = \mathcal{N}(NN(\mathbf{x}; \boldsymbol{\pi}), \tau^{-1}\mathbf{1}) \tag{5.25}$$

and can be alternatively modelled as

$$\tau = \frac{l^2(1-p)}{2N\lambda}, \tag{5.26}$$

where l defines a characteristic length scale (Gal, 2016). This test-time stochastic sampling through dropout is referred to as Monte Carlo dropout.

[2] In practice, stochastic gradient descent will select a mini-batch of these samples to give an unbiased estimate.

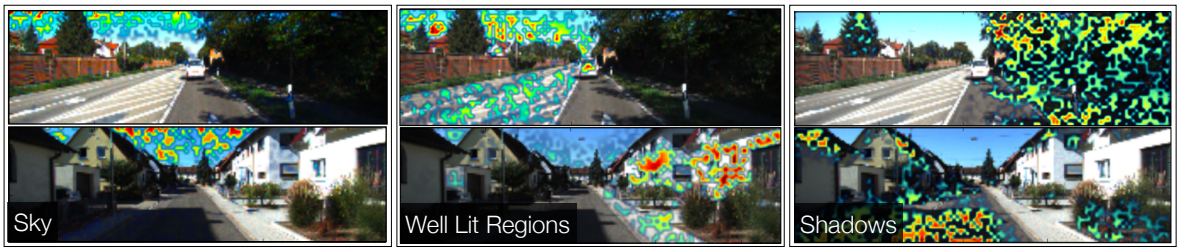


Figure 5.3: Three conv1 layer activation maps superimposed on two images from the KITTI odometry benchmark (Geiger et al., 2013) 00 and 04 for three selected filters. Each filter picks out salient parts of the image that aid in sun direction inference.

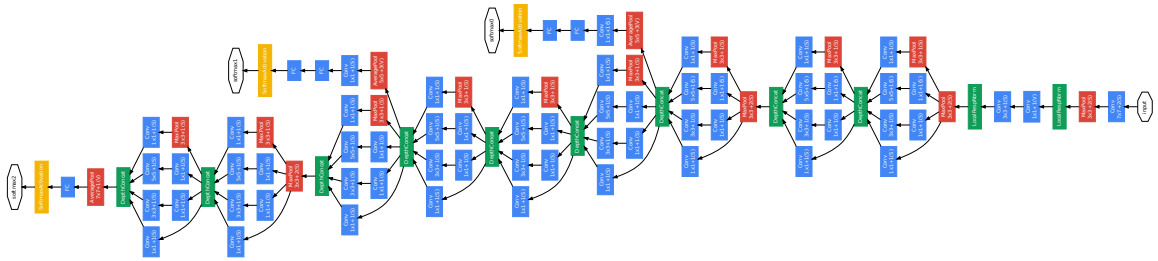


Figure 5.4: The GoogLeNet architecture used in this work. This deep convolutional network relies on *inception* modules that concatenate the output of several convolutions with a max-pooling operation. Figure from Szegedy et al. (2015).

5.6.3 Extension to Convolutional Neural Networks

Although the derivation is more tedious, Gal (2016); Gal and Ghahramani (2016a) show that the same approximate variational inference can be carried out for convolutional neural networks by applying dropout after every convolution (and before pooling) during training and using MC dropout during testing.

5.7 Indirect Sun Detection using a Bayesian Convolutional Neural Network

Leveraging the work of Gal and Ghahramani (2016a), we apply MC dropout to a Convolutional Neural Network (CNN) (approximating a BCNN, or Bayesian CNNN) to infer the direction of the sun and an associated uncertainty, and refer to our model as Sun-BCNN. We motivate the choice of a deep model through the empirical findings of Clement et al. (2017) and Ma et al. (2016), who demonstrated that a CNN-based sun detector can substantially outperform hand-crafted models such as that of Lalonde et al. (2011) 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). Unlike Ma et al. (2016), 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.

5.7.1 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:

$$\mathcal{L}\left(\hat{\mathbf{s}}_k\right) = 1 - \left(\hat{\mathbf{s}}_k \cdot \mathbf{s}_k\right). \tag{5.27}$$

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 Euclidian distance between $\mathbf{s}_k$ and $\hat{\mathbf{s}}_k$, which is equivalent to Equation (5.27) since both vectors have unit length:

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

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

5.7.2 Uncertainty Estimation

Using our notation for sun direction, we obtain the first two moments of the predictive distribution from our BCNN as (c.f. Equations (5.23) and (5.24)):

$$\mathbb{E}\left[\hat{\mathbf{s}}^*\right]_k = \hat{\bar{\mathbf{s}}}_k^* \approx \frac{1}{N}\sum_{n=1}^{N} \hat{\mathbf{s}}_k^*(\mathbf{x}_k^*, \boldsymbol{w}^n) \tag{5.28}$$

$$\begin{aligned}\mathrm{Var}(\hat{\mathbf{s}}_k^*) \approx \tau^{-1}\mathbf{1} &+ \frac{1}{N}\sum_{n=1}^{N} \hat{\mathbf{s}}_k^*(\mathbf{x}_k^*, \boldsymbol{w}^n)\, \hat{\mathbf{s}}_k^*(\mathbf{x}_k^*, \boldsymbol{w}^n)^T \\ &- \hat{\bar{\mathbf{s}}}_k^*\, \hat{\bar{\mathbf{s}}}_k^{*\,T},\end{aligned} \tag{5.29}$$

where $\mathbf{1}$ is the identity matrix, and $\boldsymbol{w}^n$ is a weight sample obtained indirectly by performing a forward pass with dropout. Following Gal and Ghahramani (2016a), we build our BCNN by adding dropout layers after every convolutional and fully connected layer in the network. We then retain these layers at test time to sample the network stochastically, following the technique of Monte Carlo Dropout, and obtain the relevant statistical quantities using Equations (5.28) and (5.29).

5.7.3 Implementation and Training

We implement 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 train 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 requires approximately 2.5 hours on an NVIDIA Titan X GPU. Interestingly, Figure 5.4 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. (2011).

Data Preparation & Transfer Learning

We resize images from their original size to $[224 \times 224]$ pixels to achieve the image size expected by GoogleLeNet. 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. (2016)) results in better test-time accuracy. We do not crop or rotate the images, nor do we augment the dataset in any other way.

Model Precision

We find an empirically optimal model precision τ by optimizing the Average Normalized Estimation Error Squared (ANEES) across the entire test set for each dataset. While this hyperparameter should in principle be tuned using a validation set, we omit this step to keep our training procedure consistent with that of Ma et al. (2016). We note that the BCNN uncertainty estimates are affected by two significant factors: 1) variational inference is known to underestimate 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 partition 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 varies 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.

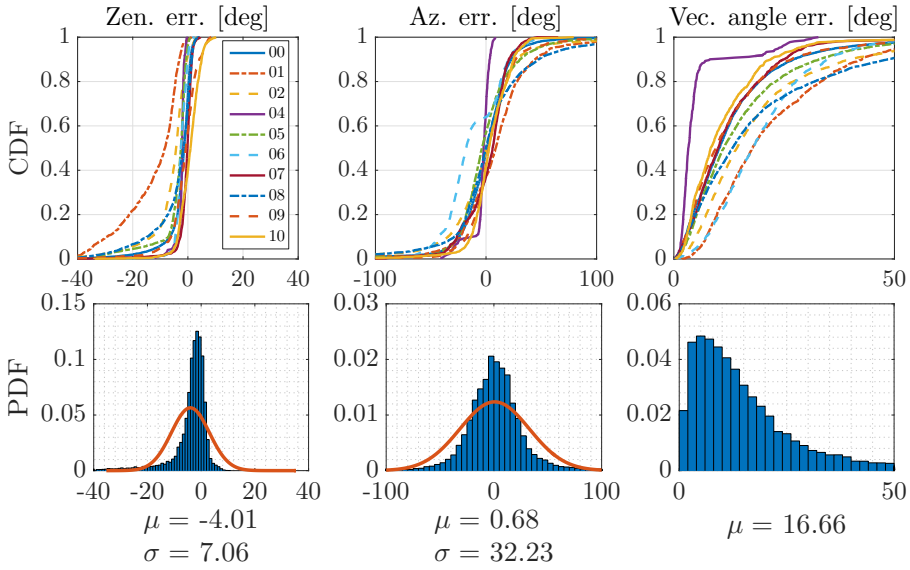


Figure 5.5: Distributions of azimuth error, zenith error, and angular distance for Sun-BCNN compared to ground truth over each test sequence in the KITTI dataset. *Top row*: Cumulative distributions of errors for each test sequence individually. *Bottom row:* Histograms and Gaussian fits of aggregated errors.

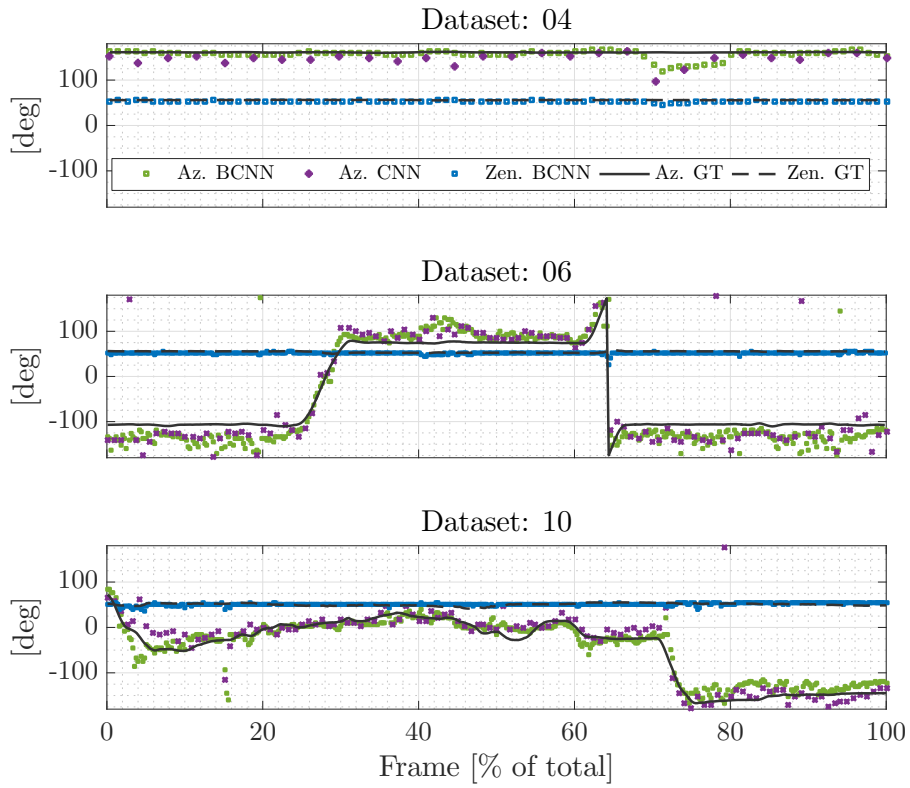


Figure 5.6: Azimuth (Sun-CNN and Sun-BCNN) and zenith (Sun-BCNN only) predictions over time for KITTI test sequences 04, 06 and 10. Sun-CNN is trained and tested on every tenth image, whereas Sun-BCNN is trained and tested on every image. In our VO experiments, we use the Sun-BCNN predictions of every tenth image to make a fair comparison.

Table 5.1: 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 $\tau^{-1} = 0.015$.

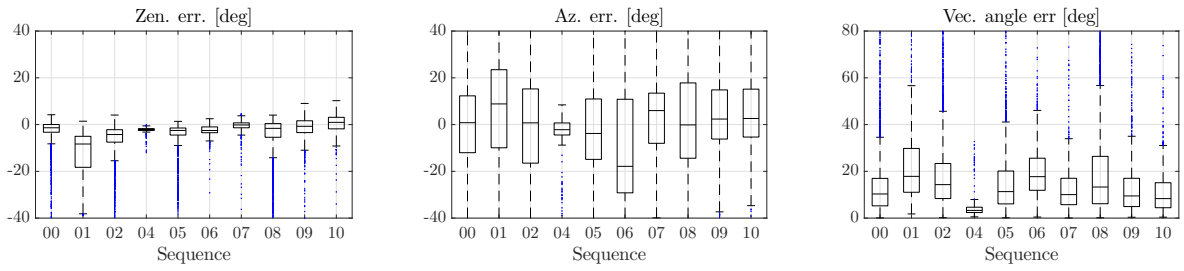
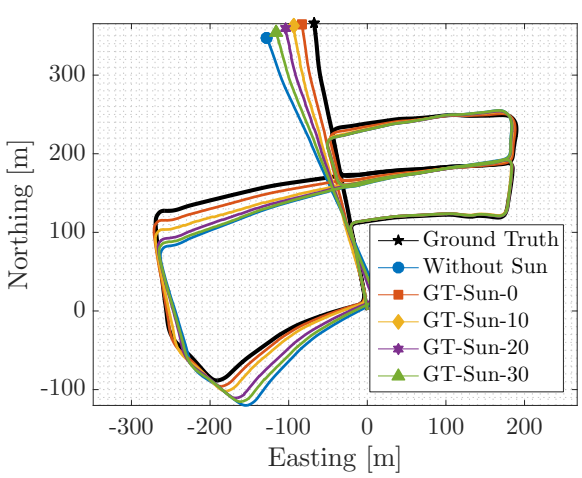
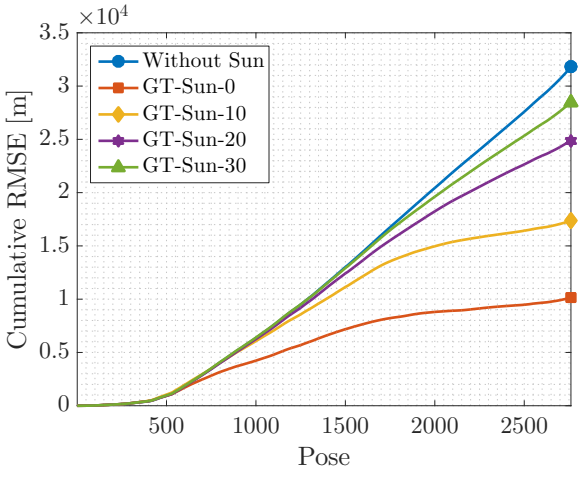


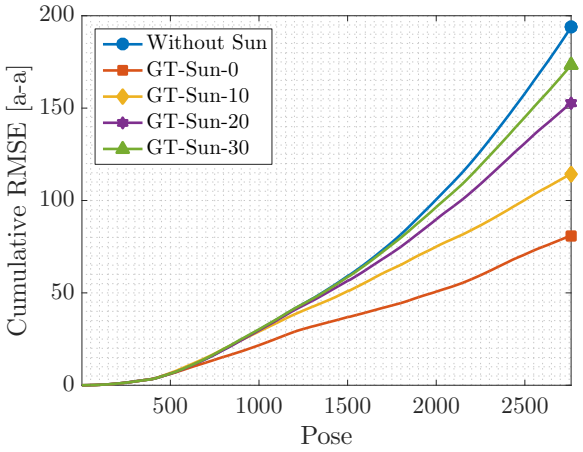
Figure 5.7: Box-and-whiskers plot of final test errors on all ten KITTI odometry sequences (c.f. Table 5.1).



(a) VO and ground truth trajectories



(b) Translational CRMSE (EN-plane)



(c) Rotational CRMSE

Figure 5.8: 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.

5.8 Urban Driving Experiments: The KITTI Odometry Benchmark

We investigated the performance of Sun-BCNN on the KITTI odometry benchmark training set (Geiger et al., 2013), which consists of 21.6 km of urban driving data[4]. Importantly, the dataset includes 6-DOF ground truth poses obtained from an accurate GPS/INS tracking system, as well as calibrated transformations between this sensor and the colour stereo pair we use for sun estimation and VO in our experiments. This allows us to create a training set of ground truth sun vectors for each image by querying the solar ephemeris model at each ground truth pose and rotating the resulting vector from the GPS/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 sequence. This procedure is consistent with that of Ma et al. (2016), against whose Sun-CNN we directly compare, and allows us to evaluate each sequence using the maximum amount of training data.

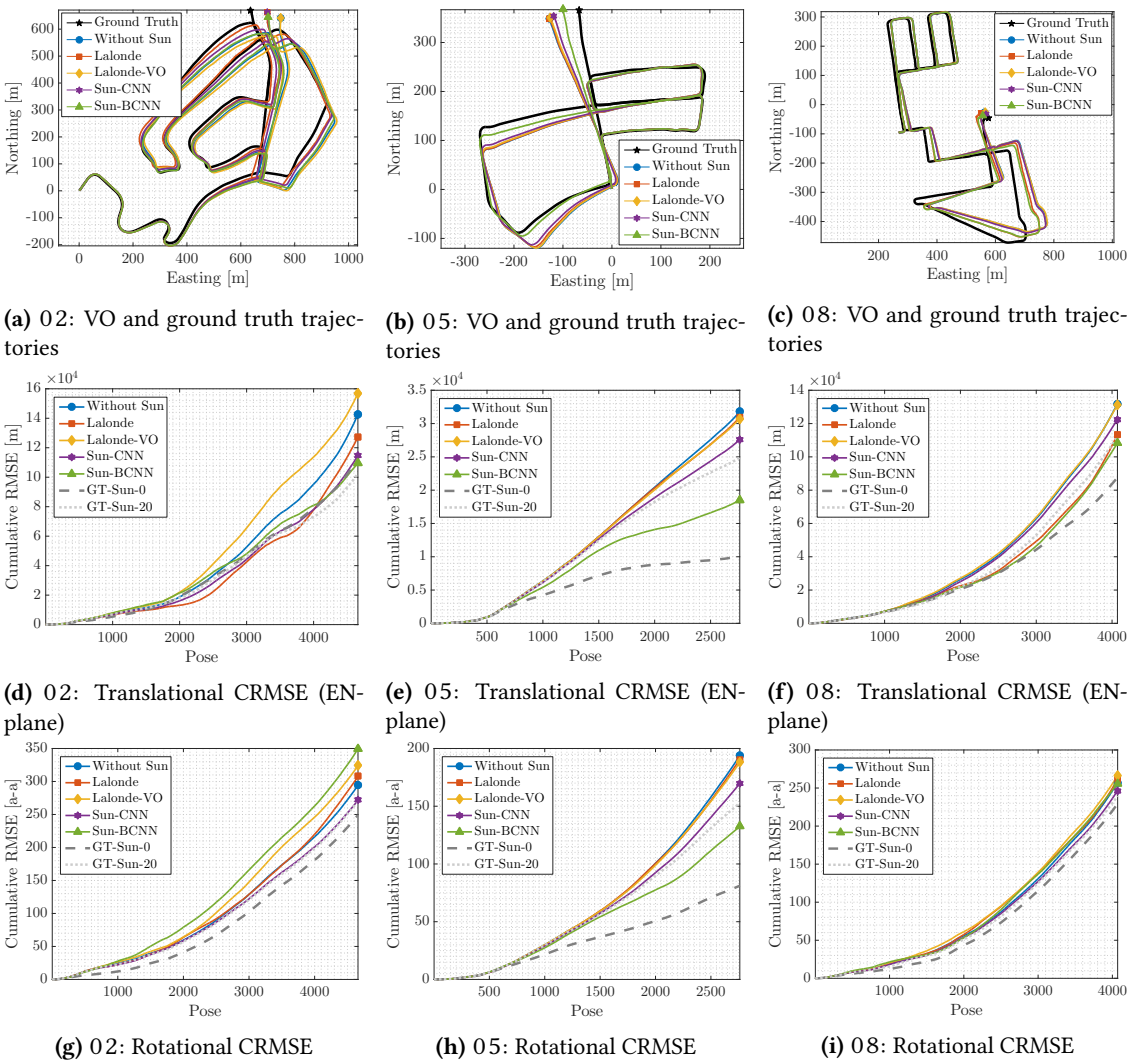
Figure 5.9: Sun BCNN predictions and associated ground truth sun directions on the KITTI sequence 05. *Top two rows*: Sun BCNN produces accurate predictions in a variety of azimuth values. *Bottom row*: Poor results occur rarely due to shadow ambiguities.

5.8.1 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 evaluating Equation (5.28) with $N = 25$ and then re-normalized the resulting vector to preserve unit length. To obtain the required covariance

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

[4] Because we rely on the first pose reported by the GPS/INS system, we used 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.



(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 5.10: 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., 2011), Lalonde-VO (Clement et al., 2017), and Sun-CNN (Ma et al., 2016) methods provide modest reductions in estimation error. The remaining sequences are less clear, but Sun-BCNN generally provides some benefit.

Table 5.2: 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 GPS/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.

on azimuth and zenith angles, we sampled the vector outputs, converted them to azimuth and zenith angles using Equation (5.7), and then applied Equation (5.29).

Remark (*Number of stochastic passes*). We base our choice of $N = 25$ on the results presented in Gal (2016), Figure 4.12. Namely, Gal shows that $N \approx 20$ provides a good balance of efficiency and accuracy (in terms of test-time error) for MC dropout with convolutional networks.

We investigate the impact of this parametrization (as opposed to working in azimuth and zenith coordinates directly) later in this chapter. As shown in Table 5.1, 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).

Figures 5.5 and 5.7 plot the error distributions for azimuth, zenith, and angular distance for all ten KITTI odometry sequences, while Figure 5.6 shows three characteristic plots of the azimuth and zenith predictions over time. 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 azimuth-zenith 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 a robust Huber loss on the sun measurement errors in our optimization problem.

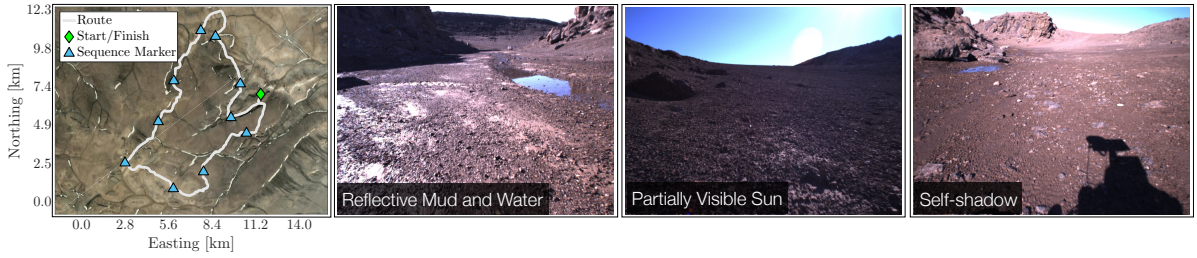


Figure 5.11: GPS track and sample images from the Devon Island traverse, with the start of each sequence highlighted. The Devon Island dataset is conducive to visual sun sensing due to the presence of strong environmental shadows, reflective surfaces such as mud and water, occasionally visible sun, and self-shadowing by the sensor platform. (Map data: Google, DigitalGlobe)

Table 5.1 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 stems 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.

5.8.2 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 compare them against the results obtained using simulated sun measurements with varying levels of noise, the method of Lalonde et al. (2011) and its VO-informed variant (Clement et al., 2017), and the Sun-CNN of Ma et al. (2016).

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 5.8 shows the results we obtained using simulated sun measurements on sequence 05, in which the basic VO suffers from substantial orientation drift.[5] Incorporating absolute orientation information from the simulated sun sensor allows the VO to correct these errors, but the magnitude of the correction decreases as sensor noise increases, consistent with the results of our simulation experiments. As shown in Table 5.2, which summarizes our VO results for all ten sequences, this is typical of sequences where orientation drift is the dominant source of error.

While the VO solutions for sequences such as 00 do not improve in terms of translational ARMSE, Table 5.2 shows that rotational ARMSE nevertheless improves 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 are dominated by non-rotational effects, and that the apparent benefit of the Lalonde method on translational ARMSE in sequence 00 is likely coincidental.

Vision-based Sun Sensing

Figure 5.9 illustrates the behaviour of Sun-BCNN on four characteristic images from test sequence 05 by overlaying the Sun-BCNN predictions and associated ground truth sun directions for each image. The two frames in the top row 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 mitigate the influence of these outlier measurements by imposing a robust Huber loss on the sun measurement errors in our optimizer.

Figure 5.10 shows the results we obtained for sequences 02, 05, and 08 using the Sun-CNN of

[5] In order to make a fair comparison to the Sun-CNN of Ma et al. (2016), 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.

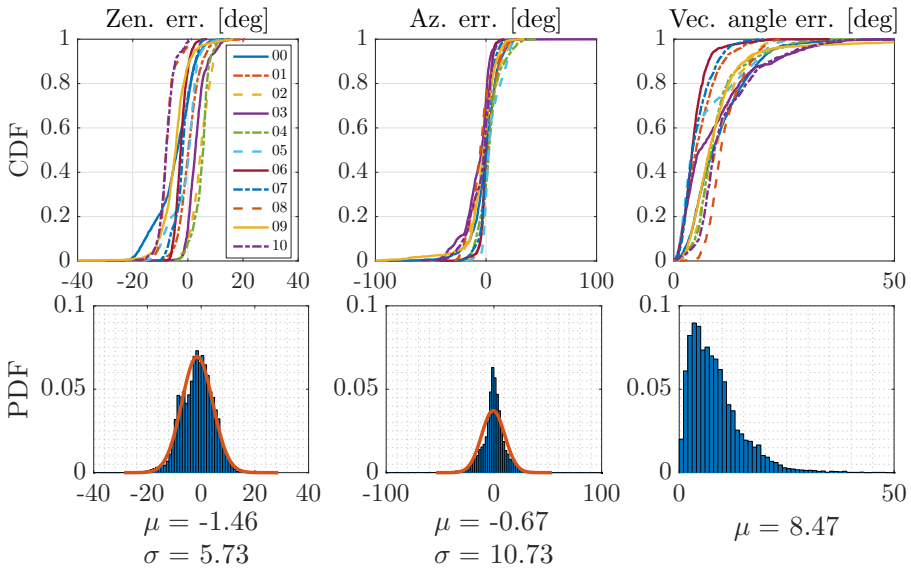


Figure 5.12: (Devon Island) Distributions of azimuth error, zenith error, and angular distance for Sun-BCNN compared to ground truth over each test sequence. *Top row*: Cumulative distributions of errors for each test sequence individually. *Bottom row:* Histograms and Gaussian fits of aggregated errors.

Ma et al. (2016), 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. (2011) in its original and VO-informed (Clement et al., 2017) 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 succeed 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 outperform the two Lalonde variants, consistent with the results of Ma et al. (2016) and Clement et al. (2017).

Table 5.2 shows results for all ten sequences using each method. With few exceptions, the VO results using Sun-BCNN achieve 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 do not benefit significantly from sun sensing since rotational drift is 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 and an improvement that comes without the need for additional sensors or a specially oriented camera.

5.9 Planetary Analogue Experiments: The Devon Island Rover Navigation Dataset

In addition to urban driving, we further investigate the usefulness of Sun-BCNN in the context of planetary exploration using the Devon Island Rover Navigation Dataset (Furgale et al., 2012), which

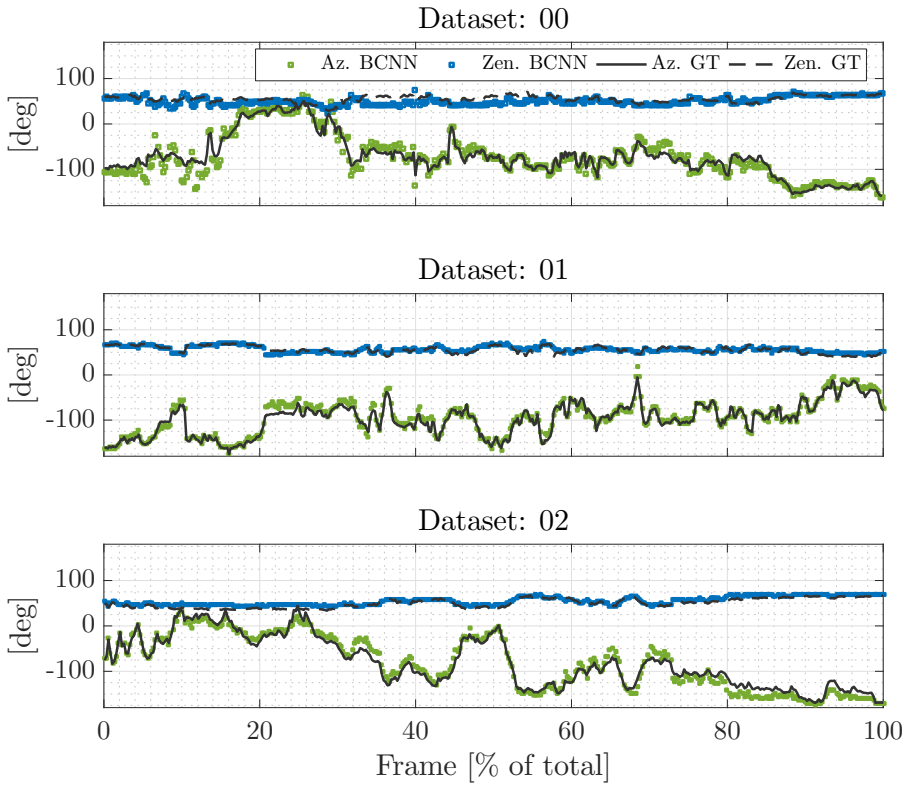


Figure 5.13: Azimuth (Sun-BCNN azimuth and zenith predictions over time for Devon Island test sequences 00, 01 and 12. Sun-BCNN is trained and tested on all frames (in our VO experiments, we use the Sun-BCNN predictions of every tenth image to make a fair comparison).

Table 5.3: Test Errors for Sun-BCNN on Devon Island 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**[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 $\tau^{-1} = 0.01$.

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 5.11). The rugged landscape of Devon Island (Figure 5.11) 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. For our purposes, we ignore the inclinometer and use the sun sensor measurements as 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 5.11, 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 partition 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 5.11). In aggregate, the sequences contain 13257 poses with associated sun sensor measurements. We apply a similar training and testing procedure as for the KITTI dataset, with the exception that we now withhold one sequence for validation and hyper-parameter tuning in addition to the sequence withheld for testing. This leaves nine sequences remaining to form the training sets for each test and validation pair.

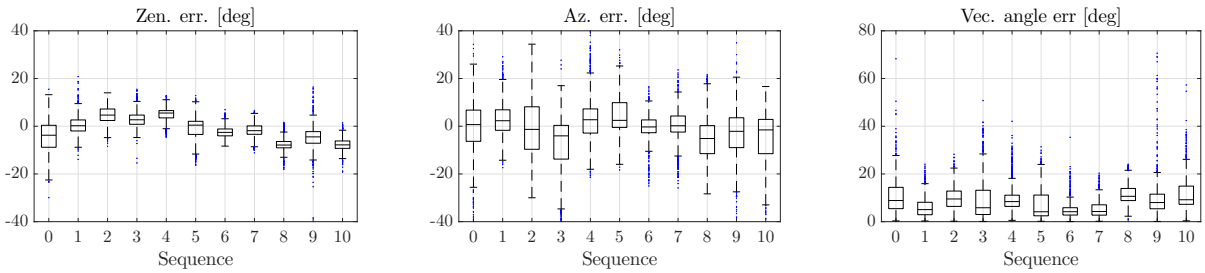


Figure 5.14: Box-and-whiskers plot of final test errors on Devon Island odometry sequences (c.f. Table 5.3).

5.9.1 Sun-BCNN Test Results

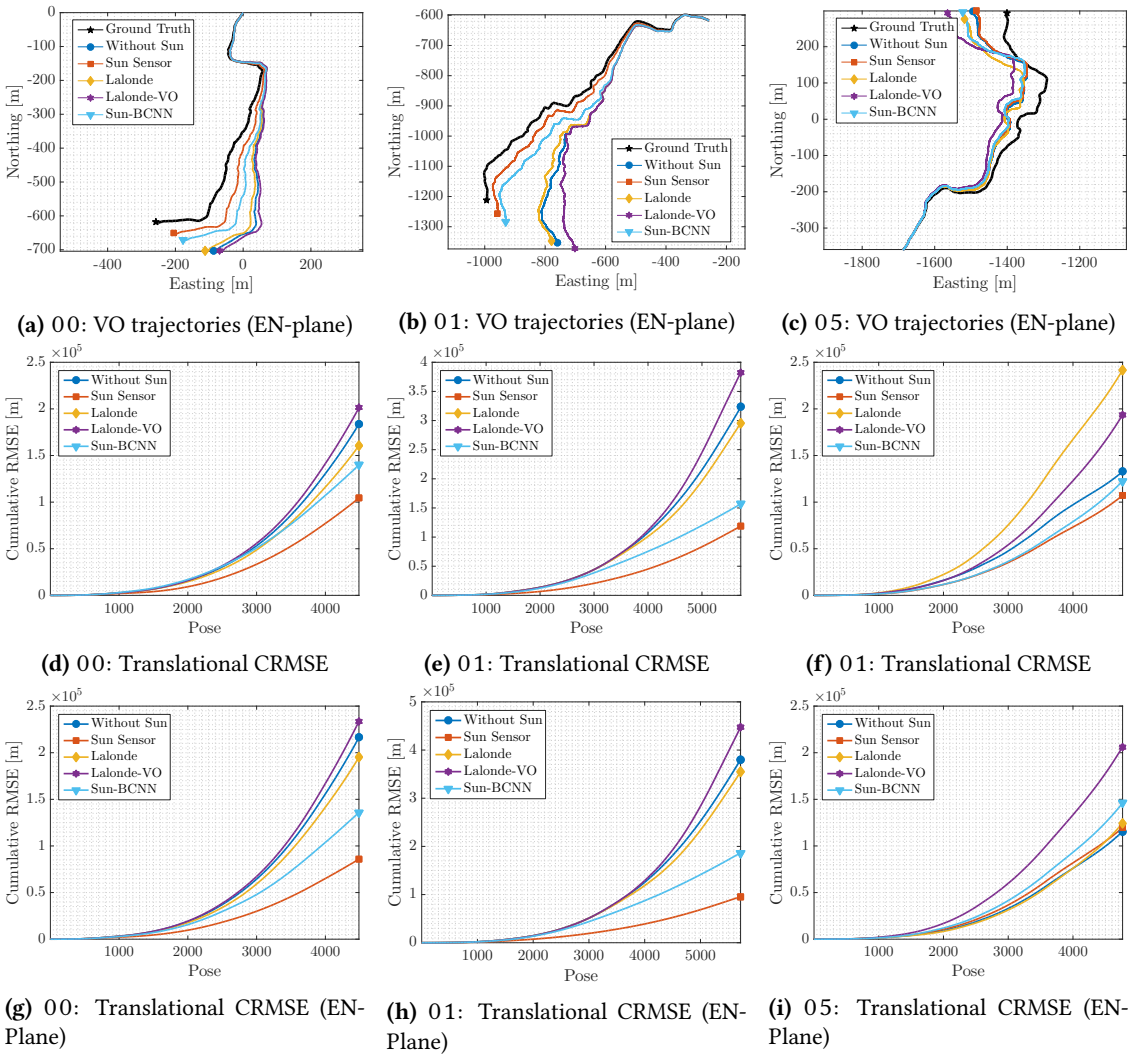
As in our experiments with the KITTI odometry benchmark, we obtained the mean estimated sun vector by evaluating Equation (5.28) with $N = 25$ and re-normalizing the resulting vector to preserve unit length. To obtain the required covariance on azimuth and zenith angles, we again sampled the vector outputs, converted them to azimuth and zenith angles using Equation (5.7), and then applied Equation (5.29). As shown in Table 5.3, 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).

Figures 5.12 and 5.14 plot the error distributions for azimuth, zenith, and angular distance for all 11 Devon Island odometry sequences, while Figure 5.13 shows three characteristic plots of the azimuth and zenith predictions over time. 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 (c.f. Figure 5.5), 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 5.3 summarizes the test errors and ANEES of each sequence numerically, while Figures 5.12 and 5.14 plot the error distributions for azimuth, zenith, and angular distance for each sequence. Figure 5.13 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 homoscedastic noise assumption is perhaps ill suited to this environment.

5.9.2 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 orientations for every image. We also do not report results using the Sun-CNN



(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 5.15: 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 (c.f. **??**).

of Ma et al. (2016) 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., 2011) and Lalonde-VO (Clement et al., 2017) methods.

Figure 5.15 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 5.4 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

Table 5.4: 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**

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.

5.10 Sensitivity Analysis

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.

5.10.1 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 5.4, 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., 2016), which consists of 1000 km of urban driving along a consistent route but in varying weather conditions and at varying times over the course of a year.

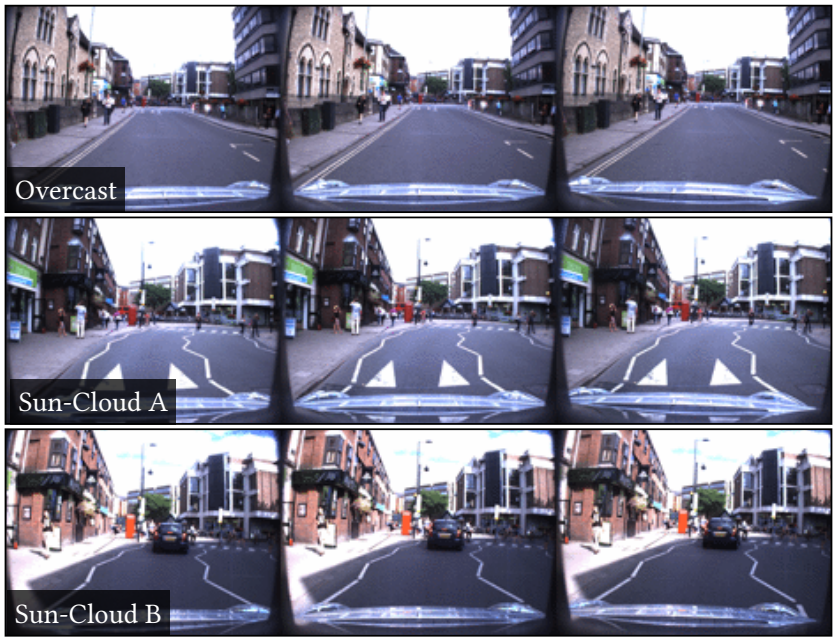


Figure 5.16: Sample images of approximately the same location taken from three different Oxford Robotcar sequences we used to investigate the effect of cloud cover on Sun-BCNN.

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 5.16 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 5.17 shows the results of these experiments with box and whisker plots for azimuth, zenith and vector angle errors while Table 5.5 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 different test conditions.

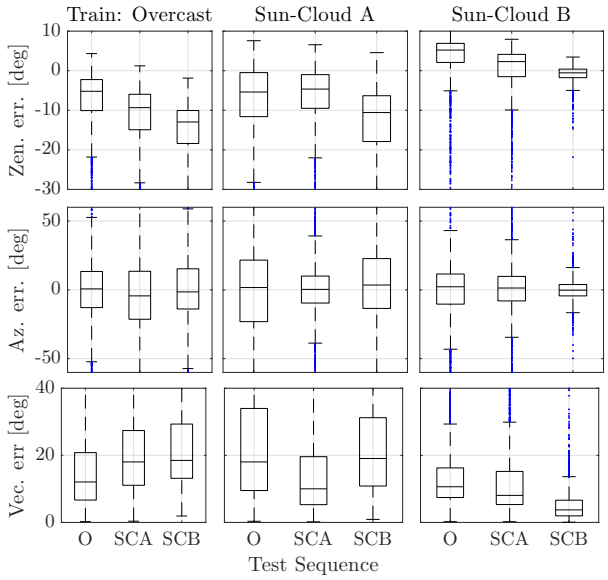


Figure 5.17: Box-and-whiskers plot for zenith, azimuth and vector angle errors for nine different combinations of train-test sequences taken from the Oxford Robotcar dataset. Each column corresponds to a different training sequence, and each plot contains three different test sequences. In the bottom legend, we use the labels O: *Overcast*, SCA: *Sun-Cloud A*, SCB: *Sun-Cloud B*.

5.10.2 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.

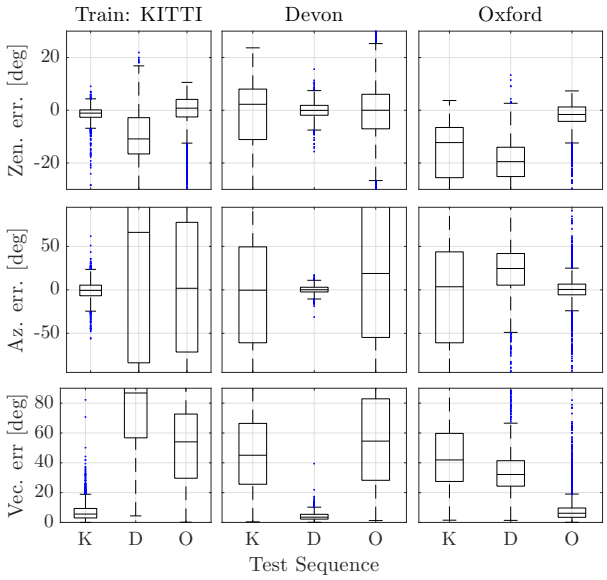


Figure 5.18: Box-and-whiskers plot for zenith, azimuth and vector angle errors for nine different combinations of train-test datasets. Each column corresponds to a different training sequence, and each plot contains three different test sequences. In the bottom legend, we use the labels K: KITTI, D: 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.

Results

Figure 5.18 shows the results of these experiments with box and whisker plots for azimuth, zenith and vector angle errors while Table 5.5 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.

5.10.3 Mean and Covariance Computation

In our formulation, Sun-BCNN outputs a sampling of unit-norm 3D vectors. Due to the unit-norm constraint, it is not immediately clear how to apply Equations (5.28) and (5.29) to calculate the mean and covariance of these samples. In this section we present and empirically evaluate two possible procedures for each computation using the previously discussed combined datasets for KITTI, Devon

Table 5.5: 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 5.6: 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 5.7: A comparison of prediction errors from different mean estimation methods.

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

Island, and Oxford.

Mean computation

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

1. In *Method I* (used in this work), we first evaluate Equation (5.28) 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 (5.7).

2. In *Method II*, we first convert each of the N unit vectors produced through stochastic passes through the BCNN to azimuth and zenith angles using Equation (5.7). We then evaluate Equation (5.28) on the angles themselves to obtain the mean in azimuth-zenith coordinates.

We evaluated both methods using the same combined datasets and partitioning scheme as in the transfer learning experiment previously presented.

Results Table 5.7 presents the azimuth, zenith 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 Computation

Procedure We further investigated two different covariance computation methods, which we also refer to as *Method I* and *Method II*.

1. In *Method I*, we first evaluate Equation (5.29) 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 (5.7).

Table 5.8: A comparison of ANEES values for different mean and covariance propagation methods.

Sequence	Covariance Type	Mean Type	ANEES
KITTI	Method I	Method I	0.95
		Method II	5.10
	Method II	Method I	1.40
		Method II	0.87
Devon	Method I	Method I	1.29
		Method II	10.05
	Method II	Method I	0.50
		Method II	0.85
Oxford	Method I	Method I	1.50
		Method II	2.14
	Method II	Method I	1.30
		Method II	0.89

2. In *Method II* (used in this work), we first convert each of the N unit vectors produced by stochastic passes through the BCNN to azimuth and zenith angles, and then evaluate Equation (5.29) on the angles themselves.

We once again re-used the transfer learning datasets with the same partitioning scheme, and 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 replace 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 the each model's corresponding test set using both mean computation methods.

Results Table 5.8 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.

5.11 Summary

In summary, with Sun-BCNN, we applied learned *pseudo-sensors* to the problem of illumination direction in outdoor environments. Sun-BCNN presented

1. the application of a Bayesian CNN to the problem of sun direction estimation, incorporating the resulting covariance estimates into a visual odometry pipeline;

2. an empirical demonstration that a Bayesian CNN with dropout layers after each convolutional and fully-connected layer can achieve state-of-the-art accuracy at test time;
3. a loss function that incorporated a 3D unit-length sun direction vector, appropriate for full 6-DOF pose estimation;
4. experimental results on over 30 km of visual navigation data in urban (Geiger et al., 2013) and planetary analogue (Furgale et al., 2012) environments;
5. an investigation into the sensitivity of the Bayesian CNN-based sun estimate to cloud cover, camera and environment changes, and measurement parameterization; and
6. open-source software[6].

[6] https://github.com/utiasSTARS/sun-bcnn-vo.

Chapter 6

Learned Pose Corrections

> Life can only be understood backwards; but it must be lived forwards.
>
> Søren Kierkegaard

6.1 Introduction

If a pseudo-sensor like Sun-BCNN can extract useful orientation information directly from images, is it possible to create a more general approach that can learn full $\mathrm{SE}(3)$ corrections to an existing egomotion estimate? In this chapter, we show that this is possible through what we call a *deep pose correction* (DPC) network.

Remark (*Associated Publications*). DPC is associated with the journal publication:

1. Peretroukhin, V. and Kelly, J. (2018). DPC-Net: Deep pose correction for visual localization. *IEEE Robotics and Automation Letters*, 3(3):2424–2431.

This chapter is largely a reproduction of that publication.

6.2 Motivation

As we saw noted in the previous chapter, convolutional neural networks (CNNs) are at the core of many state-of-the-art classification and segmentation algorithms in computer vision (LeCun et al., 2015). These CNN-based techniques achieve accuracies previously unattainable by classical methods. In mobile robotics and state estimation, however, it remains unclear to what extent these deep architectures can obviate the need for classical geometric modelling. Visual localization algorithms like VO can suffer from several systematic error sources that include estimator biases (Peretroukhin et al.,

2014), poor calibration, and environmental factors (e.g., a lack of scene texture). While machine learning approaches can be used to better model specific subsystems of a localization pipeline (e.g., the heteroscedastic feature track error covariance modelling of Peretroukhin et al. (2016, 2015a)), much recent work (Costante et al., 2016; Clark et al., 2017; Kendall et al., 2015; Melekhov et al., 2017; Oliveira et al., 2017) has been devoted to completely replacing the estimator with a CNN-based system.

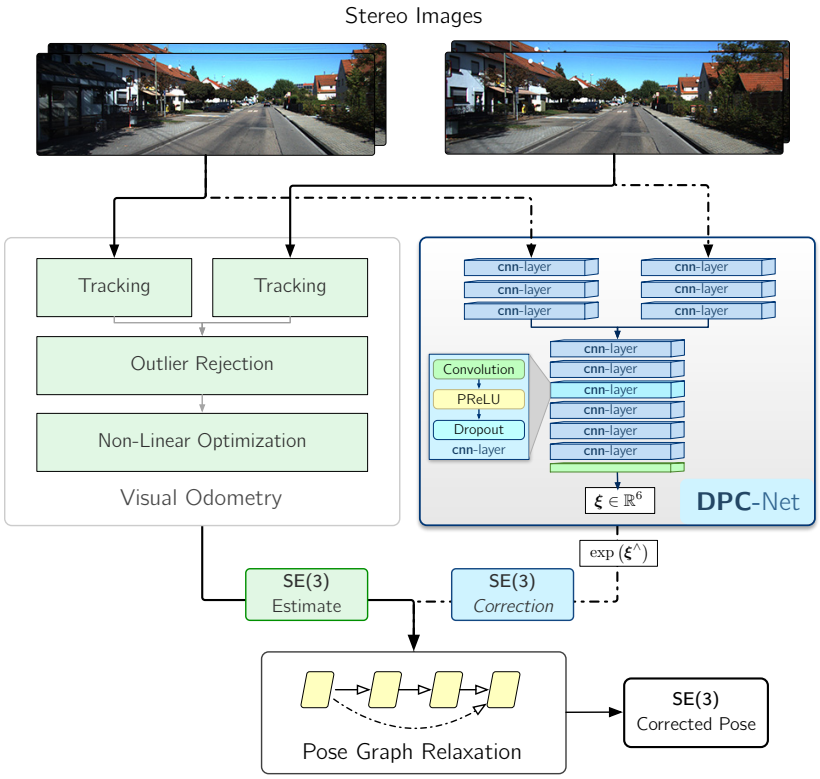


Figure 6.1: We propose a Deep Pose Correction network (DPC-Net) that learns $\mathrm{SE}(3)$ *corrections* to classical visual localizers.

We contend that this type of complete replacement places an unnecessary burden on the CNN. Not only must it learn projective geometry, but it must also understand the environment, and account for sensor calibration and noise. Instead, we take inspiration from Sun-BCNN that demonstrates that CNNs can infer difficult-to-model geometric quantities to improve an existing localization estimate. In a similar vein, we propose a system (Figure 6.1) that takes as its starting point an efficient, classical localization algorithm that computes high-rate pose estimates. To it, we add a Deep Pose Correction Network (DPC-Net) that learns low-rate, 'small' *corrections* from training data that we then fuse with the original estimates. DPC-Net does not require any modification to an existing localization pipeline, and can learn to correct multi-faceted errors from estimator bias, sensor mis-calibration or environmental effects.

Although in this chapter we focus on visual data, the DPC-Net architecture can be readily modified to learn $\mathrm{SE}(3)$ corrections for estimators that operate with other sensor modalities (e.g., lidar). For this general task, we derive a pose regression loss function and a closed-form analytic expression for its

Jacobian. Our loss permits a network to learn an unconstrained Lie algebra coordinate vector, but derives its Jacobian with respect to $\mathrm{SE}(3)$ geodesic distance.

6.3 Related Work

In the past decade, much of machine learning and its sub-disciplines has been revolutionized by carefully constructed deep neural networks (DNNs) (LeCun et al., 2015). For tasks like image segmentation, classification, and natural language processing, most prior state-of-the-art algorithms have been replaced by their DNN alternatives.

In mobile robotics, deep neural networks have ushered in a new paradigm of end-to-end training of visuomotor policies (Levine et al., 2016) and significantly impacted the related field of reinforcement learning (Duan et al., 2016). In state estimation, however, most successful applications of deep networks have aimed at replacing a specific sub-system of a localization and mapping pipeline (for example, object detection (Yang et al., 2016), place recognition (Sunderhauf et al., 2015), or bespoke discriminative observation functions (Haarnoja et al., 2016)).

Nevertheless, a number of recent approaches have presented convolutional neural network architectures that purport to obviate the need for classical visual localization algorithms. For example, Kendall et al. (Kendall et al., 2015; Kendall and Cipolla, 2017) presented extensive work on PoseNet, a CNN-based camera re-localization approach that regresses the 6-DOF pose of a camera within a previously explored environment. Building upon PoseNet, Melekhov (Melekhov et al., 2017) applied a similar CNN learning paradigm to relative camera motion. In related work, Costante et al. (Costante et al., 2016) presented a CNN-based VO technique that uses pre-processed dense optical flow images. By focusing on RGB-D odometry, Handa et al. Handa et al. (2016), detailed an approach to learning relative poses with *3D Spatial Transformer* modules and a dense photometric loss. Finally, Oliviera et al. (Oliveira et al., 2017) and Clark et al. (Clark et al., 2017) described techniques for more general sensor fusion and mapping. The former work outlined a DNN-based topometric localization pipeline with separate VO and place recognition modules while the latter presented VINet, a CNN paired with a recurrent neural network for visual-inertial sensor fusion and online calibration.

With this recent surge of work in end-to-end learning for visual localization, one may be tempted to think this is the only way forward. It is important to note, however (as noted in the introduction), that these deep CNN-based approaches do not yet report state-of-the-art localization accuracy, focusing instead on proof-of-concept validation.

Taking inspiration from Sun-BCNN (Chapter 5) that shows that CNNs can be used to inject global orientation information into a visual localization pipeline, and leveraging ideas from recent approaches to trajectory tracking in the field of controls (Li et al., 2017a; Punjani and Abbeel, 2015), we formulate a system that learns *pose corrections* to an existing estimator, instead of learning the entire localization problem *ab initio*.

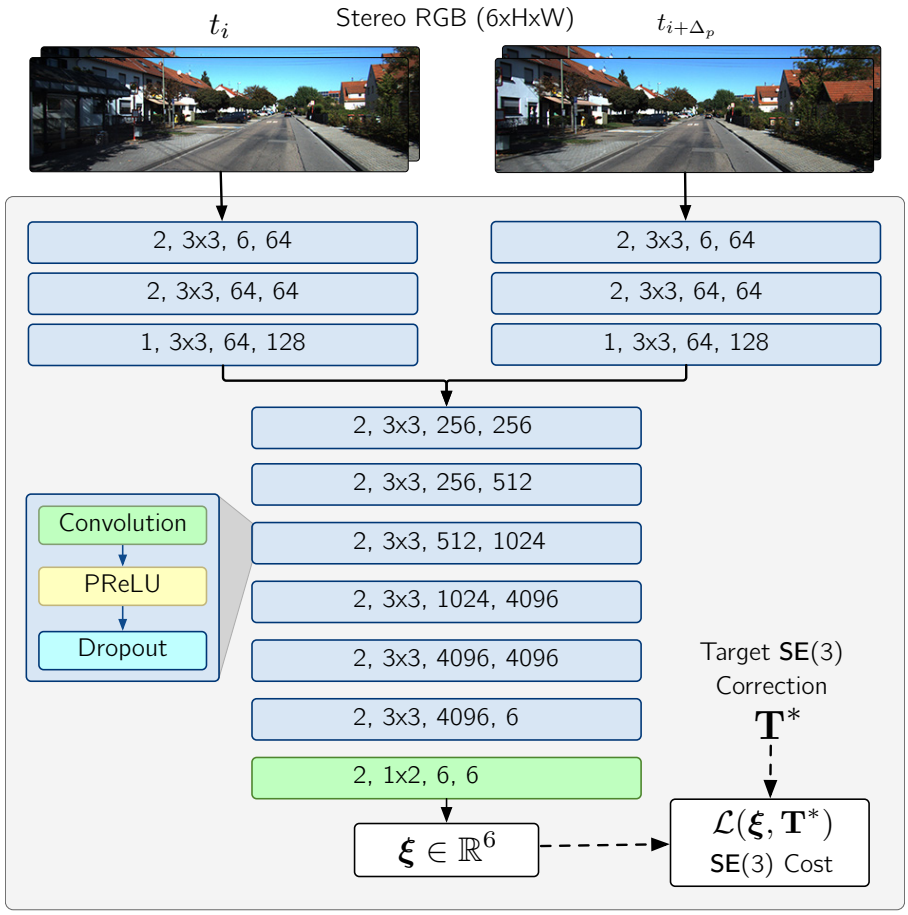


Figure 6.2: The Deep Pose Correction network with stereo RGB image data. The network learns a map from two stereo pairs to a vector of Lie algebra coordinates. Each darker blue block consists of a convolution, a PReLU non-linearity, and a dropout layer. We opt to not use MaxPooling in the network, following Handa et al. (2016). The labels correspond to the stride, kernel size, and number of input and output channels of for each convolution.

6.4 System Overview: Deep Pose Correction

We base our network structure on that of (Handa et al., 2016) but learn $\mathrm{SE}(3)$ *corrections* from stereo images and require no pre-training. Similar to (Handa et al., 2016), we use primarily 3x3 convolutions and avoid the use of max pooling to preserve spatial information. We achieve downsampling by setting the stride to 2 where appropriate (see Figure 6.2 for a full description of the network structure). We derive a novel loss function, unlike that used in Kendall and Cipolla (2017); Melekhov et al. (2017); Oliveira et al. (2017), based on $\mathrm{SE}(3)$ geodesic distance. Our loss naturally balances translation and rotation error without requiring a hand-tuned scalar hyper-parameter. Similar to Costante et al. (2016), we test our final system on the KITTI odometry benchmark, and evaluate how it copes with degraded visual data.

Given two coordinate frames $\underset{\sim}{\mathcal{F}}_i$, $\underset{\sim}{\mathcal{F}}_{i+\Delta_p}$ that represent a camera's pose at time t_i and $t_{i+\Delta_p}$ (where Δ_p is an integer hyper-parameter that allows DPC-Net to learn corrections across multiple temporally consecutive poses), we assume that our visual localizer gives us an estimate, $\hat{\mathbf{T}}_{i,i+\Delta_p}$, of the true transform $\mathbf{T}_{i,i+\Delta_p} \in \mathrm{SE}(3)$ between the two frames. We aim to learn a target correction,

$$\mathbf{T}_i^* = \mathbf{T}_{i,i+\Delta_p} \hat{\mathbf{T}}_{i,i+\Delta_p}^{-1}, \tag{6.1}$$

from two pairs of stereo images (collectively referred to as $\mathbf{I}_{t_i,t_{i+\Delta_p}}$) captured at t_i and $t_{i+\Delta_p}$. Note that the visual estimator does not necessarily compute $\hat{\mathbf{T}}_{i,i+\Delta_p}$ directly. $\hat{\mathbf{T}}_{i,i+\Delta_p}$ may be compounded from several estimates. Given a dataset, $\{\mathbf{T}_i^*, \mathbf{I}_{t_i,t_{i+\Delta_p}}\}_{i=1}^N$, we now turn to the problem of selecting an appropriate loss function for learning $\mathrm{SE}(3)$ corrections.

6.4.1 Loss Function: Correcting $\mathrm{SE}(3)$ Estimates

In this work, we choose to parametrize our correction prediction as $\mathbf{T} = \exp\left(\boldsymbol{\xi}^\wedge\right)$, where $\boldsymbol{\xi} \in \mathbb{R}^6$, a vector of Lie algebra coordinates, is the output of our network (similar to Handa et al. (2016)). We define a loss for $\boldsymbol{\xi}$ as

$$\mathcal{L}(\mathbf{T}, \mathbf{T}^*) = \mathcal{L}(\boldsymbol{\xi}, \mathbf{T}^*) = \frac{1}{2} g(\boldsymbol{\xi})^T \boldsymbol{\Sigma}^{-1} g(\boldsymbol{\xi}), \tag{6.2}$$

where

$$g(\boldsymbol{\xi}) \triangleq \log\left(\exp\left(\boldsymbol{\xi}^\wedge\right) \mathbf{T}^{*-1}\right)^\vee. \tag{6.3}$$

Here, $\boldsymbol{\Sigma}$ is the covariance of our estimator (expressed using unconstrained Lie algebra coordinates), and $(\cdot)^\wedge$, $(\cdot)^\vee$ are defined as in Chapter 2. Given two stereo image pairs, we use the output of DPC-Net, $\boldsymbol{\xi}$, to correct our estimator as follows:

$$\hat{\mathbf{T}}^{\text{corr}} = \exp\left(\boldsymbol{\xi}^\wedge\right) \hat{\mathbf{T}}, \tag{6.4}$$

where we have dropped subscripts for clarity.

Remark (*Loss Functions over* $\mathrm{SE}(3)$). An alternative possible approach to learning $\mathbf{T}^*$ is to break it into constituent parts and then compose a loss function as the weighted sum of translational and rotational error norms (as done in Kendall et al. (2015); Melekhov et al. (2017); Oliveira et al. (2017)). That is,

$$\mathcal{L}(\mathbf{T}, \mathbf{T}^*) = \mathcal{L}_{\text{rot}}(\mathbf{C}, \mathbf{C}^*) + \beta \mathcal{L}_{\text{trans}}(\mathbf{r}, \mathbf{r}^*). \tag{6.5}$$

This however does not account for the possible correlation between the two losses, and requires the careful tuning of a scalar weight β.

6.4.2 Loss Function: $\mathrm{SE}(3)$ Covariance

Importantly, since we are learning small estimator *corrections*, we can compute an empirical covariance over the training set as

$$\boldsymbol{\Sigma} = \frac{1}{N-1} \sum_{i=1}^{N} \left(\boldsymbol{\xi}_i^* - \overline{\boldsymbol{\xi}^*} \right) \left(\boldsymbol{\xi}_i^* - \overline{\boldsymbol{\xi}^*} \right)^T, \tag{6.6}$$

where

$$\boldsymbol{\xi}_i^* \triangleq \log \left(\mathbf{T}_i^* \right)^\vee, \quad \overline{\boldsymbol{\xi}^*} \triangleq \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\xi}_i^*. \tag{6.7}$$

The term $\boldsymbol{\Sigma}$ balances the rotational and translation loss terms based on their magnitudes in the training set, and accounts for potential correlations. We stress that if we were learning poses directly, the pose targets and their associated mean would be trajectory dependent and would render this type of covariance estimation meaningless. Further, we find that, in our experiments, $\boldsymbol{\Sigma}$ weights translational and rotational errors similarly to that presented in **?** based on the diagonal components, but contains relatively large off-diagonal terms.

6.4.3 Loss Function: $\mathrm{SE}(3)$ Jacobians

In order to use Equation (6.2) to train DPC-Net with back-propagation, we need to compute its Jacobian with respect to our network output, $\boldsymbol{\xi}$. Applying the chain rule, we begin with the expression

$$\frac{\partial \mathcal{L}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = g(\boldsymbol{\xi})^T \boldsymbol{\Sigma}^{-1} \frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}. \tag{6.8}$$

The term $\dfrac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$ is of importance. We can derive it in two ways. To start, note two important identities Barfoot (2017). First, recalling the identity introduced in Chapter 2,

$$\exp\left((\boldsymbol{\xi} + \delta\boldsymbol{\xi})^\wedge \right) \approx \exp\left((\boldsymbol{\mathcal{J}} \delta\boldsymbol{\xi})^\wedge \right) \exp\left(\boldsymbol{\xi}^\wedge \right), \tag{6.9}$$

where $\mathcal{J} \triangleq \mathcal{J}(\boldsymbol{\xi})$ is the left SE(3) Jacobian. Second, if $\mathbf{T}_1 \triangleq \exp\left(\boldsymbol{\xi}_1{}^\wedge\right)$ and $\mathbf{T}_2 \triangleq \exp\left(\boldsymbol{\xi}_2{}^\wedge\right)$, then

$$\begin{aligned}\log\left(\mathbf{T}_1\mathbf{T}_2\right)^\vee &= \log\left(\exp\left(\boldsymbol{\xi}_1{}^\wedge\right)\exp\left(\boldsymbol{\xi}_2{}^\wedge\right)\right)^\vee \\ &\approx \begin{cases} \mathcal{J}(\boldsymbol{\xi}_2)^{-1}\boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_1 \text{ small} \\ \boldsymbol{\xi}_1 + \mathcal{J}(-\boldsymbol{\xi}_1)^{-1}\boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_2 \text{ small.} \end{cases}\end{aligned} \tag{6.10}$$

To derive $\dfrac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$, we can linearize Equation (6.3) about $\boldsymbol{\xi}$, by considering a small change $\delta\boldsymbol{\xi}$ and applying Equation (6.9):

$$g(\boldsymbol{\xi}+\delta\boldsymbol{\xi}) = \log\left(\exp\left(\left(\boldsymbol{\xi}+\delta\boldsymbol{\xi}\right)^\wedge\right)\mathbf{T}^{*-1}\right)^\vee \tag{6.11}$$

$$\approx \log\left(\exp\left(\left(\mathcal{J}\delta\boldsymbol{\xi}\right)^\wedge\right)\exp\left(\boldsymbol{\xi}^\wedge\right)\mathbf{T}^{*-1}\right)^\vee. \tag{6.12}$$

Now, assuming that $\mathcal{J}\delta\boldsymbol{\xi}$ is 'small', and using Equation (6.3), Equation (6.10) gives:

$$g(\boldsymbol{\xi}+\delta\boldsymbol{\xi}) \approx \mathcal{J}(g(\boldsymbol{\xi}))^{-1}\mathcal{J}(\boldsymbol{\xi})\delta\boldsymbol{\xi} + g(\boldsymbol{\xi}). \tag{6.13}$$

Comparing this to the first order Taylor expansion: $g(\boldsymbol{\xi}+\delta\boldsymbol{\xi}) \approx g(\boldsymbol{\xi}) + \dfrac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\delta\boldsymbol{\xi}$, we see that

$$\frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \mathcal{J}(g(\boldsymbol{\xi}))^{-1}\mathcal{J}(\boldsymbol{\xi}). \tag{6.14}$$

This expression makes no assumptions about the 'magnitude' of our correction and works reliably for any target. To summarize, to apply back-propagation to Equation (6.2), we use Equation (6.8) and Equation (6.14).

Remark (*Alternative Derivation of* $\dfrac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$). If we assume that only $\boldsymbol{\xi}$ is 'small', we can apply Equation (6.10) directly to define

$$\frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \mathcal{J}(-\boldsymbol{\xi}^*)^{-1}, \tag{6.15}$$

with $\boldsymbol{\xi}^* \triangleq \log\left(\mathbf{T}^*\right)^\vee$. Although attractively compact, note that this expression for $\dfrac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$ assumes that $\boldsymbol{\xi}$ is small, and may be inaccurate for 'larger' $\mathbf{T}^*$ (since we will therefore require $\boldsymbol{\xi}$ to be commensurately 'large'). Note further that if $\boldsymbol{\xi}$ is small, then $\mathcal{J}(\boldsymbol{\xi}) \approx \mathbf{1}$ and $\exp\left(\boldsymbol{\xi}^\wedge\right) \approx \mathbf{1}$. Thus,

$$g(\boldsymbol{\xi}) \approx \log\left(\mathbf{T}^{*-1}\right)^\vee = -\boldsymbol{\xi}^*, \tag{6.16}$$

and Equation (6.14) becomes

$$\frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \mathcal{J}(-\boldsymbol{\xi}^*)^{-1}. \tag{6.17}$$

.

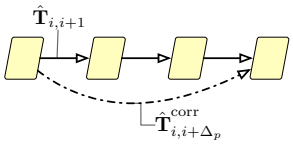


Figure 6.3: We apply pose graph relaxation to fuse high-rate visual localization estimates ($\hat{\mathbf{T}}_{i,i+1}$) with low-rate deep pose corrections ($\hat{\mathbf{T}}^{\text{corr}}_{i,i+\Delta_p}$).

6.4.4 Loss Function: Correcting $\mathrm{SO}(3)$ Estimates

Our framework can be easily modified to learn $\mathrm{SO}(3)$ corrections only. We can parametrize a similar objective for $\boldsymbol{\phi} \in \mathbb{R}^3$,

$$\mathcal{L}(\boldsymbol{\phi}, \mathbf{C}_*) = \frac{1}{2} f(\boldsymbol{\phi})^T \boldsymbol{\Sigma}^{-1} f(\boldsymbol{\phi}), \tag{6.18}$$

where

$$f(\boldsymbol{\phi}) \triangleq \log\left(\exp\left(\boldsymbol{\phi}^\wedge\right) \mathbf{C}_*{}^{-1}\right)^\vee. \tag{6.19}$$

Equations (6.9) and (6.10) have analogous $\mathrm{SO}(3)$ formulations:

$$\exp\left((\boldsymbol{\phi} + \delta\boldsymbol{\phi})^\wedge\right) \approx \exp\left((\mathbf{J}\delta\boldsymbol{\phi})^\wedge\right) \exp\left(\boldsymbol{\phi}^\wedge\right), \tag{6.20}$$

and

$$\begin{aligned} \log\left(\mathbf{C}_1\mathbf{C}_2\right)^\vee &= \log\left(\exp\left(\boldsymbol{\phi}_1{}^\wedge\right) \exp\left(\boldsymbol{\phi}_2{}^\wedge\right)\right)^\vee \\ &\approx \begin{cases} \mathbf{J}(\boldsymbol{\phi}_2)^{-1}\boldsymbol{\phi}_1 + \boldsymbol{\phi}_2 & \text{if } \boldsymbol{\phi}_1 \text{ small} \\ \boldsymbol{\phi}_1 + \mathbf{J}(-\boldsymbol{\phi}_1)^{-1}\boldsymbol{\phi}_2 & \text{if } \boldsymbol{\phi}_2 \text{ small,} \end{cases} \end{aligned} \tag{6.21}$$

where $\mathbf{J} \triangleq \mathbf{J}(\boldsymbol{\phi})$ is the left $\mathrm{SO}(3)$ Jacobian. Accordingly, the final loss Jacobians are identical in structure to Equation (6.8) and Equation (6.14), with the necessary $\mathrm{SO}(3)$ replacements.

6.4.5 Pose Graph Relaxation

In practice, we find that using camera poses several frames apart (i.e. $\Delta_p > 1$) often improves test accuracy and reduces overfitting. As a result, we turn to pose graph relaxation to fuse low-rate corrections with higher-rate visual pose estimates. For a particular window of $\Delta_p + 1$ poses (see Figure 6.3), we solve the non linear minimization problem

$$\{\mathbf{T}_{t_i,n}\}_{i=0}^{\Delta_p} = \underset{\{\mathbf{T}_{t_i,n}\}_{i=0}^{\Delta_p} \in \mathrm{SE}(3)}{\operatorname{argmin}} \quad \mathcal{O}_t, \tag{6.22}$$

where n refers to a common navigation frame, and where we define the total cost, $\mathcal{O}_t$, as a sum of visual estimation and correction components:

$$\mathcal{O}_t = \mathcal{O}_v + \mathcal{O}_c. \tag{6.23}$$

The former cost sums over each estimated transform,

$$\mathcal{O}_v \triangleq \sum_{i=0}^{\Delta_p - 1} \mathbf{e}_{t_i,t_{i+1}}^T \, \mathbf{\Sigma}_v^{-1} \mathbf{e}_{t_i,t_{i+1}}, \tag{6.24}$$

while the latter incorporates a single pose correction,

$$\mathcal{O}_c \triangleq \mathbf{e}_{t_0,t_{\Delta_p}}^T \, \mathbf{\Sigma}_c^{-1} \mathbf{e}_{t_0,t_{\Delta_p}}, \tag{6.25}$$

with the pose error defined as

$$\mathbf{e}_{1,2} = \log\left(\hat{\mathbf{T}}_{1,2}\mathbf{T}_2\mathbf{T}_1^{-1}\right)^{\vee}. \tag{6.26}$$

We refer the reader to Barfoot (2017) for a detailed treatment of pose-graph relaxation.

6.5 Experiments

To assess the power of our proposed deep corrective paradigm, we trained DPC-Net on visual data with localization estimates from the canonical indirect stereo visual odometry (S-VO) estimator we presented in Chapter 3. In addition to training the full $\mathrm{SE}(3)$ DPC-Net, we modified the loss function and input data to learn simpler $\mathrm{SO}(3)$ rotation corrections, and simpler still, yaw angle corrections. For reference, we compared S-VO with different DPC-Net corrections to a state-of-the-art dense estimator. Finally, we trained DPC-Net on visual data and localization estimates from radially-distorted and cropped images.

6.5.1 Training & Testing

For all experiments, we used the KITTI odometry benchmark training set Geiger et al. (2013). Specifically, our data consisted of the eight sequences 00,02 and 05-10 (we removed sequences 01, 03, 04 to ensure that all data originated from the 'residential' category for training and test consistency).

For testing and validation, we selected the first three sequences (00,02, and 05). For each test sequence, we selected the subsequent sequence for validation (i.e., for test sequence 00 we validated with 02, for 02 with 05, etc.) and used the remaining sequences for training. We note that by design, we train DPC-Net to predict corrections for a specific sensor and estimator pair. A pre-trained DPC-Net may further serve as a useful starting point to fine-tune new models for other sensors and estimators. In this work, however, we focus on the aforementioned KITTI sequences and leave a thorough investigation of generalization for future work.

To collect training samples, $\{\mathbf{T}_i^*, \mathbf{I}_{t_i,t_{i+\Delta_p}}\}_{i=0}^N$, we used a stereo visual odometry estimator and GPS-INS ground-truth from the KITTI odometry dataset[1]. We resized all images to $[400, 120]$ pixels, approximately preserving their original aspect ratio[2]. For non-distorted data, we use $\Delta_p \in [3, 4, 5]$

[1] We used RGB stereo images to train DPC-Net but grayscale images for the estimator.

[2] Because our network is fully convolutional, it can, in principle, operate on different image resolutions with no modifications, though we do not investigate this ability in this work.

for training, and test with $\Delta_p = 4$. For distorted data, we reduce this to $\Delta_p \in [2, 3, 4]$ and $\Delta_p = 3$, respectively, to compensate for the larger estimation errors.

Our training datasets contained between 35,000 and 52,000 training samples[3] depending on test sequence. We trained all models for 30 epochs using the Adam optimizer, and selected the best epoch based on the lowest validation loss.

Rotation

To train rotation-only corrections, we extracted the $\mathrm{SO}(3)$ component of $\mathbf{T}_i^*$ and trained our network using Equation (6.18). Further, owing to the fact that rotation information can be extracted from monocular images, we replaced the input stereo pairs in DPC-Net with monocular images from the left camera[4].

Yaw

To further simplify the corrections, we extracted a single-degree-of-freedom yaw rotation correction angle[5] from $\mathbf{T}_i^*$, and trained DPC-Net with monocular images and a mean squared loss.

6.5.2 Estimators

Sparse Visual Odometry

Our baseline estimator is based on the reprojection-error based maximum likelihood approach detailed in Chapter 3. As before, we rely on the open-source `viso2` package Geiger et al. (2011), to detect and track sparse stereo image key-points, $\mathbf{y}_{i,c_1}$ and $\mathbf{y}_{i,c_0}$, between stereo image pairs (assumed to be undistorted and rectified). We model reprojection errors (due to sensor noise and quantization) as zero-mean Gaussians with a known covariance, $\mathbf{R}$:

$$\mathbf{e}_{i,t} = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}_t \mathbf{f}^{-1}(\mathbf{y}_{i,c_0})) \tag{6.27}$$

$$\sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{6.28}$$

where $\mathbf{f}(\cdot)$ is the stereo camera projection function and we set $\mathbf{R} = \mathrm{diag}(\begin{smallmatrix}1 & 1 & 4\end{smallmatrix})$ px^2. To generate an initial guess and to reject outliers, we use three point Random Sample Consensus (RANSAC) based on stereo reprojection error. Finally, we solve for the maximum likelihood transform, $\mathbf{T}_t^*$, through a Gauss-Newton minimization:

$$\mathbf{T}_t^* = \underset{\mathbf{T}_t \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} \mathbf{e}_{i,t}^T \mathbf{R}^{-1} \mathbf{e}_{i,t}. \tag{6.29}$$

Figure 6.4: Illustration of our image radial distortion procedure. Left: rectified RGB image (frame 280 from KITTI odometry sequence 05). Middle: the same image with radial distortion applied. Right: distorted, cropped, and scaled image.

Sparse Visual Odometry with Radial Distortion

Similar to Costante et al. (2016), we modified our input images to test our network's ability to correct estimators that compute poses from degraded visual data. Unlike Costante et al. (2016), who darken and blur their images, we chose to simulate a poorly calibrated lens model by applying radial distortion to the (rectified) KITTI dataset using a plumb-bob distortion model. The model computes radially-distorted image coordinates, x_d, y_d, from the normalized coordinates x_n, y_n as

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \left(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6\right) \begin{bmatrix} x_n \\ y_n \end{bmatrix}, \tag{6.30}$$

where $r = \sqrt{x_n^2 + y_n^2}$. We set the distortion coefficients, κ_1, κ_2, and κ_3 to $-0.3, 0.2, 0.01$ respectively, to approximately match the KITTI radial distortion parameters. We solved Equation (6.30) iteratively and used bilinear interpolation to compute the distorted images for every stereo pair in a sequence. Finally, we cropped each distorted image to remove any whitespace. Figure 6.4 illustrates this process.

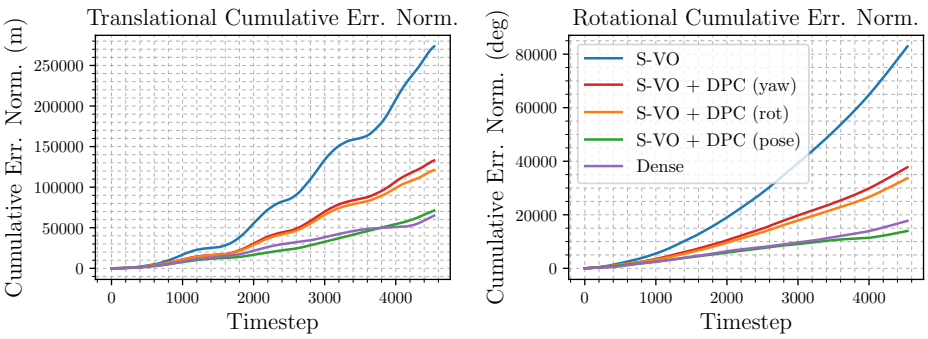
With this distorted dataset, we computed S-VO localization estimates and then trained DPC-Net to correct for the effects of the radial distortion and effective intrinsic parameter shift due to the cropping process.

Direct Keyframe-Based Visual Odometry

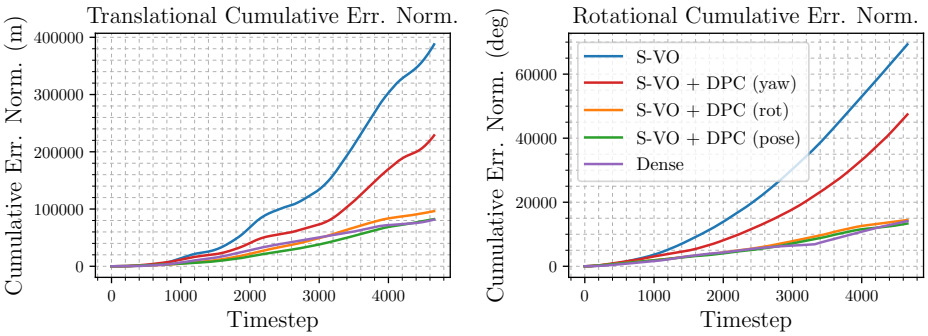
Finally, we present localization estimates from a computationally-intensive keyframe-based direct visual localization pipeline largely based on the framework presented in Engel et al. (2018). This pipeline computes relative camera poses by minimizing photometric error with respect to a keyframe image. To compute the photometric error, the pipeline relies on an inverse compositional approach to map the image coordinates of a tracking image to the image coordinates of the reference depth image. As the camera moves through an environment, a new keyframe depth image is computed and stored when the camera field-of-view differs sufficiently from the last keyframe.

We used this expensive direct estimator as our benchmark for a state-of-the-art visual localizer, and compared its accuracy to that of a much less computationally expensive sparse estimator paired with DPC-Net.

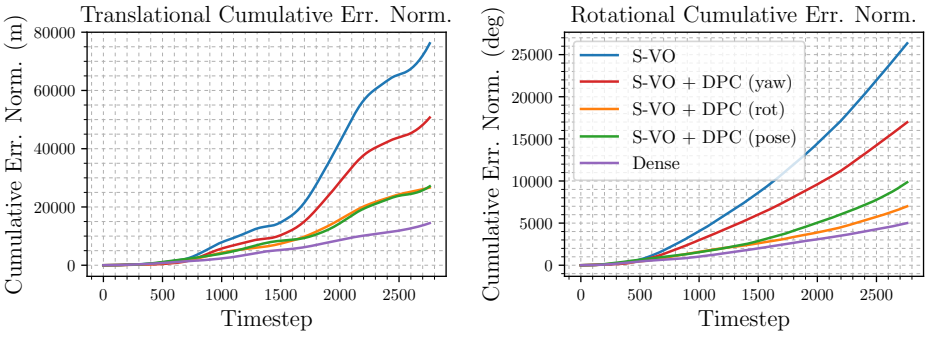
[3] If a sequence has M poses, we collect $M - \Delta_p$ training samples for each Δ_p.
[4] The *stereo* VO estimator remained unchanged.
[5] We define yaw in the camera frame as the rotation about the camera's vertical y axis.



(a) Sequence 00: Cumulative Errors.

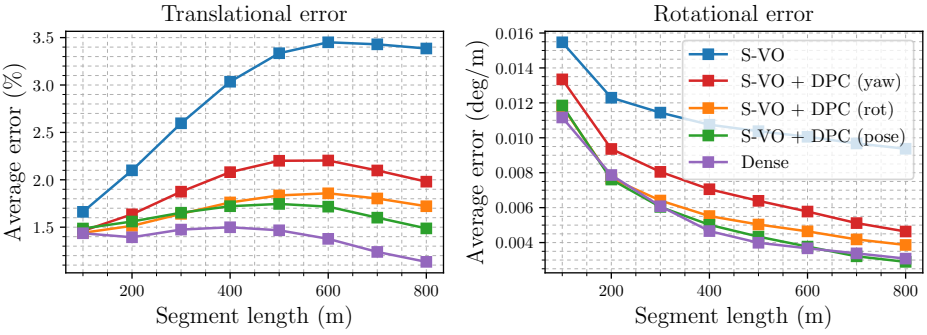


(b) Sequence 02: Cumulative Errors.

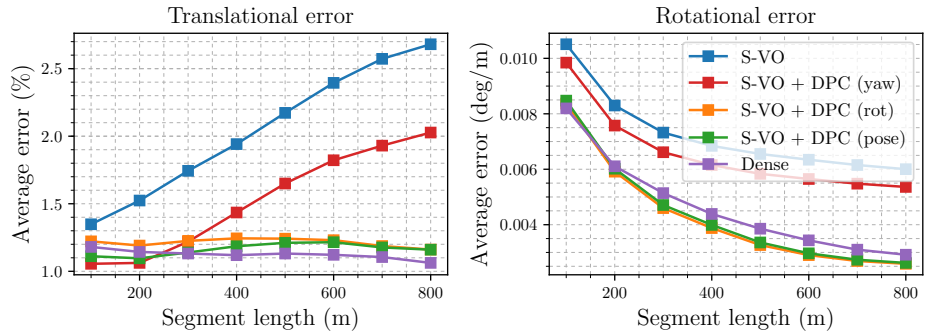


(c) Sequence 05: Cumulative Errors.

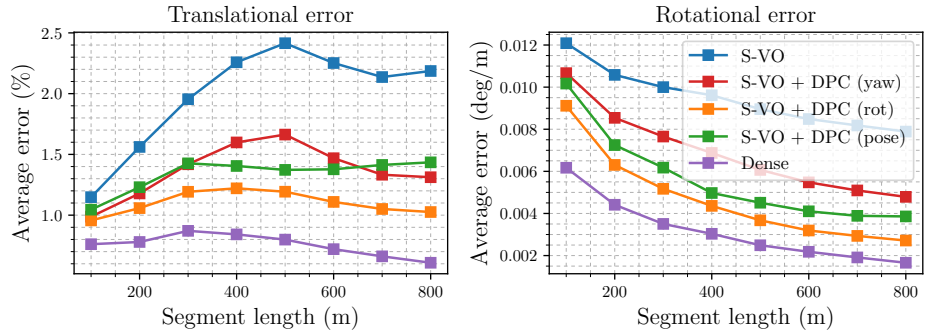
Figure 6.5: c-ATE for S-VO with and without DPC-Net.



(a) Sequence 00: Segment Errors.

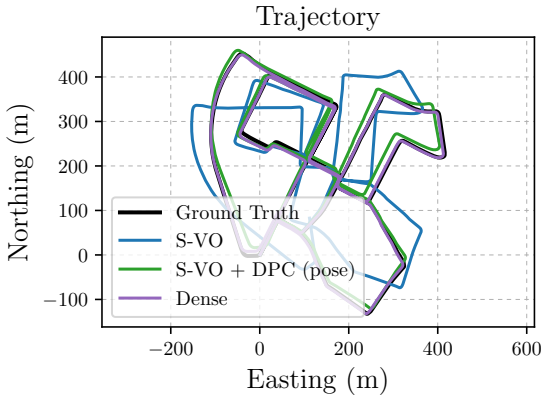


(b) Sequence 02: Segment Errors.

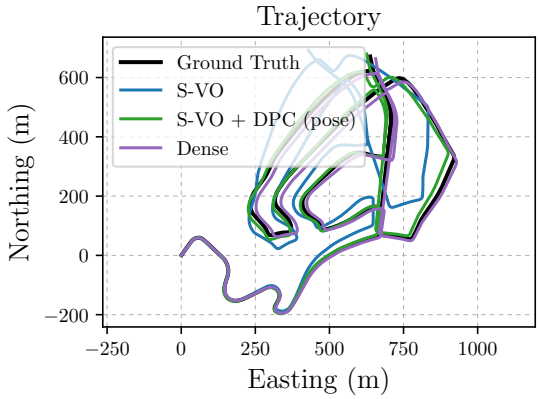


(c) Sequence 05: Segment Errors.

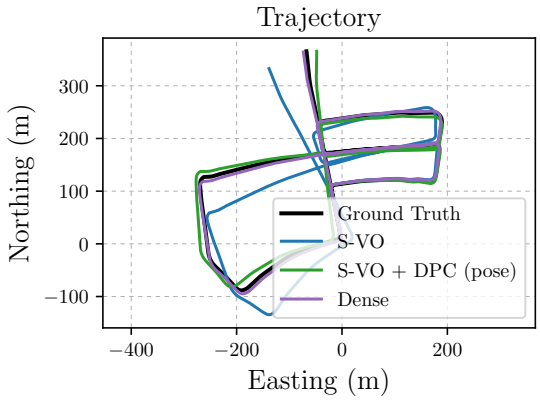
Figure 6.6: Mean segment errors for S-VO with and without DPC-Net.



(a) Sequence 00.



(b) Sequence 02.



(c) Sequence 05.

Figure 6.7: Top down projections for S-VO with and without DPC-Net.

6.5.3 Evaluation Metrics

To evaluate the performance of DPC-Net, we use three error metrics: mean absolute trajectory error, cumulative absolute trajectory error, and mean segment error. For clarity, we describe each of these three metrics explicitly and stress the importance of carefully selecting and defining error metrics when comparing *relative* localization estimates, as results can be subtly deceiving.

Mean Absolute Trajectory Error (m-ATE)

The mean absolute trajectory error averages the magnitude of the rotational or translational error[6] of estimated poses with respect to a ground truth trajectory defined within the same navigation frame. Concretely, $\mathbf{e}_{\text{m-ATE}}$ is defined as

$$\mathbf{e}_{\text{m-ATE}} \triangleq \frac{1}{N}\sum_{p=1}^{N}\left\|\log\left(\hat{\mathbf{T}}_{p,0}^{-1}\mathbf{T}_{p,0}\right)^{\vee}\right\|. \tag{6.31}$$

Although widely used, m-ATE can be deceiving because a single poor relative transform can significantly affect the final statistic.

Cumulative Absolute Trajectory Error (c-ATE)

Cumulative absolute trajectory error sums rotational or translational $\mathbf{e}_{\text{m-ATE}}$ up to a given point in a trajectory. It is defined as

$$\mathbf{e}_{\text{c-ATE}}(q) \triangleq \sum_{p=1}^{q}\left\|\log\left(\hat{\mathbf{T}}_{p,0}^{-1}\mathbf{T}_{p,0}\right)^{\vee}\right\|. \tag{6.32}$$

c-ATE can show clearer trends than m-ATE (because it is less affected by fortunate trajectory overlaps), but it still suffers from the same susceptibility to poor (but isolated) relative transforms.

Segment Error

Our final metric, segment error, averages the end-point error for all the possible segments of a given length within a trajectory, and then normalizes by the segment length. Since it considers multiple starting points within a trajectory, segment error is much less sensitive to isolated degradations. Concretely, $\mathbf{e}_{\text{seg}}(s)$ is defined as

$$\mathbf{e}_{\text{seg}}(s) \triangleq \frac{1}{sN_s}\sum_{p=1}^{N_s}\left\|\log\left(\hat{\mathbf{T}}_{p+s_p,p}^{-1}\mathbf{T}_{p+s_p,p}\right)^{\vee}\right\|, \tag{6.33}$$

where N_s and s_p (the number of segments of a given length, and the number of poses in each segment, respectively) are computed based on the selected segment length s. In this work, we follow the KITTI benchmark and report the mean segment error norms for all $s \in [100, 200, 300, ..., 800]$ (m).

[6] For brevity, the notation $\log(\cdot)^{\vee}$ returns rotational or translational components depending on context.

Table 6.1: m-ATE and Mean Segment Errors for VO results with and without DPC-Net.

			m-ATE		Mean Segment Errors	
Sequence (Length)	**Estimator**	**Corr. Type**	Translation (m)	Rotation (deg)	Translation (%)	Rotation (millideg / m)
00 (3.7 km)[1]	S-VO	—	60.22	18.25	2.88	11.18
	Dense	—	**12.41**	**2.45**	**1.28**	**5.42**
	S-VO + DPC-Net	Pose	15.68	3.07	1.62	5.59
		Rotation	26.67	7.41	1.70	6.14
		Yaw	29.27	8.32	1.94	7.47
02 (5.1 km)[2]	S-VO	—	83.17	14.87	2.05	7.25
	Dense	—	**16.33**	3.19	1.21	4.67
	S-VO + DPC-Net	Pose	17.69	**2.86**	**1.16**	4.36
		Rotation	20.66	3.10	1.21	**4.28**
		Yaw	49.07	10.17	1.53	6.56
05 (2.2 km)[3]	S-VO	—	27.59	9.54	1.99	9.47
	Dense	—	**5.83**	**2.05**	**0.69**	**3.20**
	S-VO + DPC-Net	Pose	9.82	3.57	1.34	5.62
		Rotation	9.67	2.53	1.10	4.68
		Yaw	18.37	6.15	1.37	6.90

[1] Training sequences 05, 06, 07, 08, 09, 10. Validation sequence 02.

[2] Training sequences 00, 06, 07, 08, 09, 10. Validation sequence 05.

[3] Training sequences 00, 02, 07, 08, 09, 10. Validation sequence 06.

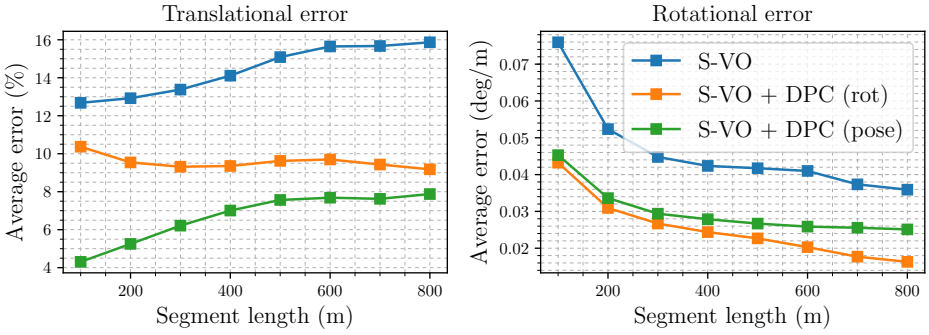
[4] All models trained for 30 epochs. The final model is selected based on the epoch with the lowest validation error.

6.6 Results & Discussion

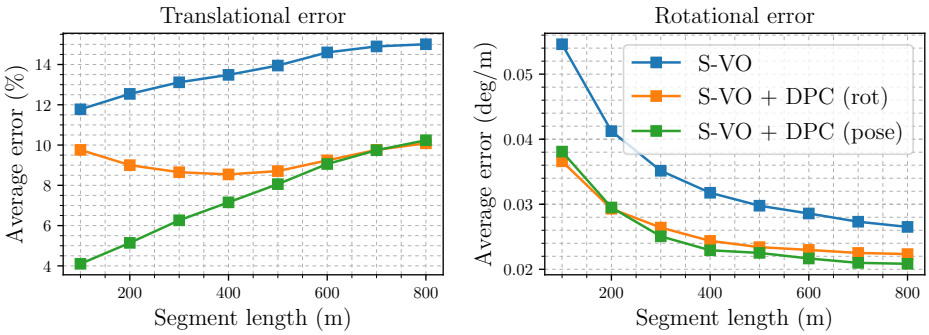
6.6.1 Correcting Sparse Visual Odometry

Figures 6.5 and 6.6 plot c-ATE and mean segment errors for test sequences 00, 02 and 05 for three different DPC-Net models paired with our S-VO pipeline. Table 6.1 summarize the results quantitatively, while Figure 6.7 plots the North-East projection of each trajectory. On average, DPC-Net trained with the full $\mathrm{SE}(3)$ loss reduced translational m-ATE by 72%, rotational m-ATE by 75%, translational mean segment errors by 40% and rotational mean segment errors by 44% (relative to the uncorrected estimator). Mean segment errors of the sparse estimator with DPC approached those observed from the dense estimator on sequence 00, and outperformed the dense estimator on 02. Sequence 05 produced two notable results: (1) although DPC-Net significantly reduced S-VO errors, the dense estimator still outperformed it in all statistics and (2) the full $\mathrm{SE}(3)$ corrections performed slightly worse than their $\mathrm{SO}(3)$ counterparts. We suspect the latter effect is a result of motion estimates with predominantly rotational errors which are easier to learn with an $\mathrm{SO}(3)$ loss.

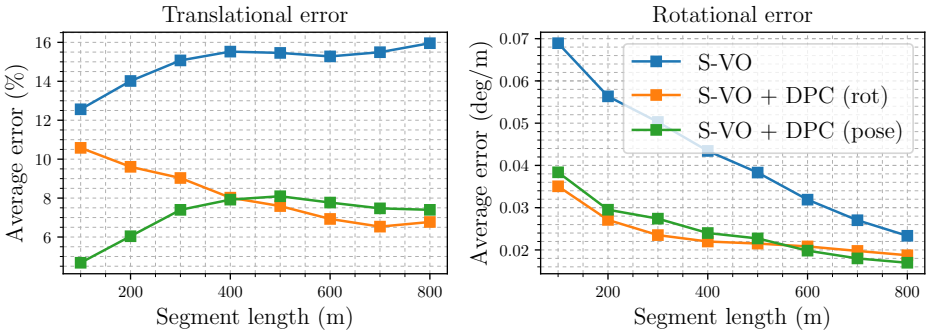
In general, coupling DPC-Net with a simple frame-to-frame sparse visual localizer yielded a final localization pipeline with accuracy similar to that of a dense pipeline while requiring significantly less visual data (recall that DPC-Net uses resized images).



(a) Sequence 00 (Distorted): Segment Errors.



(b) Sequence 02 (Distorted): Segment Errors.



(c) Sequence 05 (Distorted): Segment Errors.

Figure 6.8: c-ATE and segment errors for S-VO with radially distorted images with and without DPC-Net.

Table 6.2: m-ATE and Mean Segment Errors for VO results with and without DPC-Net for distorted images.

			m-ATE		Mean Segment Errors	
Sequence (Length)	**Estimator**	**Corr. Type**	Translation (m)	Rotation (deg)	Translation (%)	Rotation (millideg / m)
`00-distorted` (3.7 km)	S-VO	—	168.27	37.15	14.52	46.43
	S-VO + DPC	Pose	114.35	28.64	**6.73**	29.93
		Rotation	**84.54**	21.90	9.58	**25.28**
`02-distorted` (5.1 km)	S-VO	—	335.82	51.05	13.74	34.37
	S-VO + DPC	Pose	**196.90**	**23.66**	**7.49**	**25.20**
		Rotation	269.90	53.11	9.25	25.99
`05-distorted` (2.2 km)	S-VO	—	73.44	12.27	14.99	42.45
	S-VO + DPC	Pose	**47.50**	**10.54**	**7.11**	24.60
		Rotation	71.42	13.10	8.14	**23.56**

6.6.2 Distorted Images

Figure 6.8 plots mean segment errors for the radially distorted dataset. On average, DPC-Net trained with the full $\mathrm{SE}(3)$ loss reduced translational mean segment errors by 50% and rotational mean segment errors by 35% (relative to the uncorrected sparse estimator, see Table 6.2). The yaw-only DPC-Net corrections did not produce consistent improvements (we suspect due to the presence of large errors in the remaining degrees of freedom as a result of the distortion procedure). Nevertheless, DPC-Net trained with $\mathrm{SE}(3)$ and $\mathrm{SO}(3)$ losses was able to significantly mitigate the effect of a poorly calibrated camera model. We are actively working on modifications to the network that would allow the corrected results to approach those of the undistorted case.

6.7 Summary

In summary, we presented DPC-Net, an approach to learn six degree-of-freedom corrections for a canonical VO pipeline for a particular environment and camera. Our contributions included

1. the formulation of a novel deep corrective approach to egomotion estimation,
2. a novel cost function for deep $\mathrm{SE}(3)$ regression that naturally balances translation and rotation errors, and
3. an open-source implementation of DPC-Net in `PyTorch`[7].

[7] See https://github.com/utiasSTARS/dpc-net.

Chapter 7

Learned Probabilistic Rotations

> Anyone who has ever used any other parametrization of the rotation group will, within hours of taking up the quaternion parametrization, lament his or her misspent youth.
>
> Simon Altmann

7.1 Introduction

Finally, we take the lessons of Sun-BCNN (Chapter 5) and DPC-Net (Chapter 6) and present a method to learn probabilistic estimates of rotation. We show that this pseudo-sensor can extract relative rotation estimates which can be fused with full pose estimates from a canonical VO pipeline estimates through pose graph optimization. To do this, we develop a network structure we call *HydraNet* that can account for both **epistemic and aleatoric** sources of uncertainty and adapt it to the problem of estimating elements of $\mathrm{SO}(3)$.

Remark (*Associated Publications*). This work is associated with the publication

- Peretroukhin, V., Wagstaff, B., and Kelly, J. (2019). Deep probabilistic regression of elements of SO(3) using quaternion averaging and uncertainty injection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19) Workshop on Uncertainty and Robustness in Deep Visual Learning*, pages 83–86, Long Beach, California, USA.

In this chapter we elaborate on the formulation and experimental validation presented within that publication.

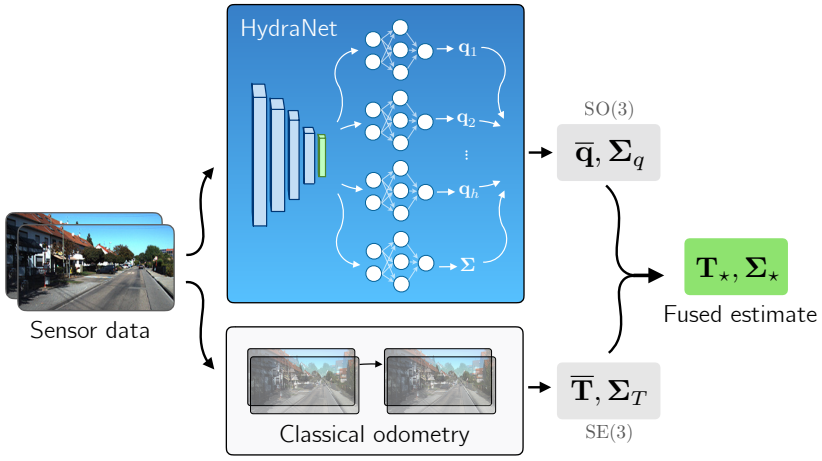


Figure 7.1: HydraNet produces an estimate of relative rotation (with a principled covariance matrix) that can be fused with existing egomotion pipelines through pose graph optimization.

7.2 Motivation

Accounting for position and orientation, or pose, is at the heart of computer vision. Many algorithms in image classification and feature tracking, for example, are explicitly concerned with output that is robust to camera orientation. Conversely, visual odometry, structure from motion, and SLAM use visual sensors to estimate and track the pose of a camera as it moves through some environment.

As discussed in the previous chapter, several recent authors (Clark et al., 2017; Melekhov et al., 2017; Kendall et al., 2015) has attempted to transfer the success of deep neural networks in many areas of computer vision to the task of camera pose estimation. These approaches, however, can produce arbitrarily poor pose estimates if sensor data differs from what is observed during training (i.e., it is 'out of training distribution') and their monolithic nature makes them difficult to debug. Further, as we have pointed out in previous chapters, despite much research effort, classical motion estimation algorithms, like indirect stereo visual odometry, still achieve state-of-the-art performance in nominal conditions. Nevertheless, the representational power of deep regression algorithms makes them an attractive option to complement classical motion estimation when these latter methods perform poorly (e.g., under diverse lighting conditions or low scene texture). By endowing deep regression models with a useful notion of uncertainty, we can account for out-of-training-distribution errors and fuse these models with classical methods using probabilistic factor graphs. In this work, we choose to focus on rotation regression, since many motion algorithms are sensitive to rotation errors (Peretroukhin et al., 2018; Olson et al., 2003), and good rotation initializations can be critical to robust optimization.

7.3 Related work

Much recent work in the literature has been devoted to replacing classical localization algorithms with deep network equivalents. Some approaches (Clark et al., 2017; Kendall et al., 2015; Kendall and Cipolla, 2017; Melekhov et al., 2017) learn poses directly, while others learn them indirectly as the

spatial transforms that result in minimal loss defined over some other domain (e.g., pixel or depth space) (Byravan and Fox, 2017; Handa et al., 2016).

Despite this surge of research in neural-network-based replacements, some authors have nevertheless used deep networks to augment classical state estimation algorithms. Deep networks have been trained as pose correctors whose corrections can be fused with existing estimates through pose graph relaxation (Peretroukhin and Kelly, 2018) (see Chapter 6), and as depth prediction networks that can be incorporated into a classical monocular pipelines to provide an initial estimate for metric scale (Yang et al., 2018). The pseudo-sensor we present in this chapter is perhaps closest in spirit to (Haarnoja et al., 2016) which fuses deep probabilistic observation functions with classical models using a Kalman Filter, but focuses on unconstrained targets and does not investigate uncertainty quantification on manifolds.

In the robotics community, there has been significant effort to leverage the tools of matrix Lie groups to handle poses and associated uncertainty (Solà et al., 2018; Barfoot and Furgale, 2014). In parallel, the computer vision community has developed a rich literature of rotation averaging (Hartley et al., 2013) which focuses on principled ways to combine elements of $\mathrm{SO}(3)$ based on different metrics defined over the group.

Finally, uncertainty in the context of deep learning has been investigated through the technique of MC Dropout (Chapter 5, Gal (2016), Kendall and Gal (2017)). Concurrently, *ensembles of networks* have been shown to be a scalable way to extract uncertainty for deep regression and classification (Lakshminarayanan et al., 2017), while multi-headed networks have been proposed in the context of ensemble learning (Lee et al., 2015) and for bootstrapped uncertainty in reinforcement learning (Osband et al., 2016). Finally, an an alternate binary approach (Richter and Roy, 2017) to dealing with uncertainty is to classify test samples as either *in training distribution* (i.e., cases where our model should have high accuracy) and *out of training distribution* (i.e., indeterminate cases where we may revert to an alternate prediction schema). This latter binary classification can be thought of as a thresholded epistemic uncertainty, and we believe, can be obviated through good uncertainty quantification.

7.4 Approach

We develop our method for probabilistic $\mathrm{SO}(3)$ regression in three steps. First, we motivate why learning elements of $\mathrm{SO}(3)$ is particularly germane to field of egomotion estimation. Second, we present a multi-headed network that can regress unconstrained targets and produce consistent uncertainty estimates. Toward this end, we present a one-dimensional regression experiment, validating prior works (Lakshminarayanan et al., 2017; Osband et al., 2016) that suggest a bootstrap-inspired approach provides better calibrated uncertainties than one based on stochastic sampling through MC dropout and can be straightforwardly extended to incorporate both aleatoric and epistemic uncertainty. Finally, we generalize these results to targets that belong to $\mathrm{SO}(3)$ by defining a rotation average using the quaternionic metric, and show how we can compute anisotropic uncertainty on four-dimensional unit quaternions.

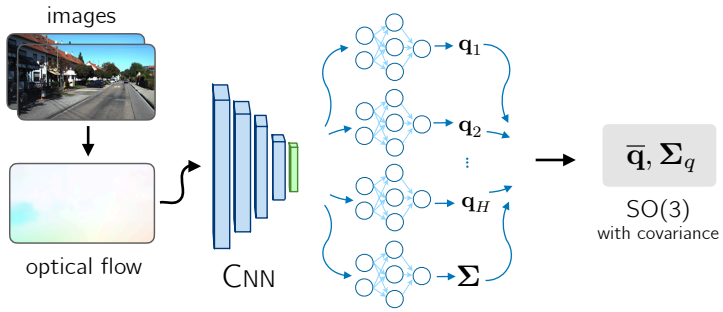


Figure 7.2: The HydraNet structure. Input data (in this case, pre-processed optical flow images) is passed through a main body and then through a number of heads. Outputs are combined to produce an average and an uncertainty.

7.4.1 Why Rotations?

We focus our attention on learning rotations for three primary reasons. First, rotations can be learned without reference to scale, using monocular images without the need for metric depth estimation. These images can come from cheap, light-weight imaging sensors that can be found on many ground and aerial vehicles. Furthermore, many depth-equipped sensors like stereo cameras and RGB-D cameras have limited depth range and produce poor depth estimates in large-scale outdoor environments. Second, many egomotion estimation techniques, like visual odometry or visual SLAM, are particularly sensitive to rotation estimates as small early errors have a large influence on final pose estimates (Olson et al., 2003). Finally, the constrained nature of rotations presents several difficulties for optimization algorithms. Indeed, if rotations are known, the general problem of pose graph relaxation becomes a linear least squares problem that can be solved with no initial guess for translations (Carlone et al., 2015b).

7.4.2 Probabilistic Regression and HydraNet

To begin, we will consider a (non-Bayesian) approach to uncertainty quantification. Consider the one dimensional regression task where, given an input $x \in \mathbb{R}$, with a target output $y_t \in \mathbb{R}$, we desire a probabilistic estimate

$$\overline{y}, \sigma^2, \tag{7.1}$$

such that σ^2 maximizes a likelihood model of our prediction $\overline{y}$ given that we know y_t.

HydraNet

One possible way to obtain $\overline{y}$ is to train a deep neural network, $g(x)$. To endow this network with uncertainty, we present a network structure we call HydraNet (see Figure 7.3). HydraNet is composed of a large, main 'body', $b(x; \boldsymbol{\pi}_b) = NN(x; \boldsymbol{\pi}_b)$ with $H+1$ heads, $h_i(x; \boldsymbol{\pi}_{h_i}) = NN(x; \boldsymbol{\pi}_{h_i})$, attached to the output of the body. Given an input x, we get $H+1$ outputs as:

$$\{y_1, ..., y_H, \sigma_a^2\} = \{h_1 \circ b(x), h_2 \circ b(x), ..., h_{H+1} \circ b(x)\} \tag{7.2}$$

where $\circ$ denotes function composition. To compute $\overline{y}$, we compute the arithmetic mean of the outputs,

$$\overline{y} = \frac{1}{H} \sum_{i=1}^{H} y_i(x). \tag{7.3}$$

The head structure, however, provides several key advantages toward the goal of estimating consistent uncertainty. Namely, it allows us to define the overall uncertainty in terms of two sources, *epistemic* (σ_e) and *aleatoric* (σ_a) (Kendall and Gal, 2017):

$$\sigma^2 = \underbrace{\sigma_{\mathrm{e}}^2}_{\text{epistemic}} + \underbrace{\sigma_{\mathrm{a}}^2}_{\text{aleatoric}}. \tag{7.4}$$

Remark (*Epistemic and Aleatoric Uncertainty*). The former, σ_e, is also sometimes referred to as model uncertainty; it is a measure of how close a particular test sample is to known training samples. The latter, σ_a, is inherent to the observation of the target itself. Even if the model can localize a test sample exactly in some salient input space, the aleatoric uncertainty will prevent exact regression due to physical processes like sensor noise.

To account for aleatoric uncertainty, we follow prior work (Haarnoja et al., 2016; Lakshminarayanan et al., 2017) and dedicate one head of the network to regressing a variance directly through a negative log likelihood loss under the assumption of Gaussian likelihood. That is, we define a supervised loss,

$$\boldsymbol{\pi}_{h_i}^*, \boldsymbol{\pi}_b^* = \underset{\boldsymbol{\pi}_i, \boldsymbol{\pi}_b}{\operatorname{argmin}} \mathcal{L}(y_i, \sigma_a^2, y_t) = \underset{\boldsymbol{\pi}}{\operatorname{argmin}} \frac{1}{2\sigma_a^2}(y - y_t)^2 + \log(\sigma_a^2), \tag{7.5}$$

where y_t is a target output.

To capture epistemic uncertainty, we train each head with random weight initializations and apply losses independently during training. During test time, we compute a sample covariance over the different outputs. That is, at test time, we compute:

$$\sigma_e^2 = \frac{1}{H-1}(y_i - \bar{y})^2 \tag{7.6}$$

This approach is inspired by the method of the statistical bootstrap (Osband et al., 2016), which predicts population statistics by computing statistics over subsets of a sample chosen with replacement. Unlike Osband et al. (2016), we do not train each head of the network with a bootstrapped sample, but instead rely on the random initializations of their parameters and the method of dropout to introduce sufficient stochasticity into their outputs. Further, unlike Lakshminarayanan et al. (2017), we do not require numerous trained models that can incur high computational cost for complex regression tasks.

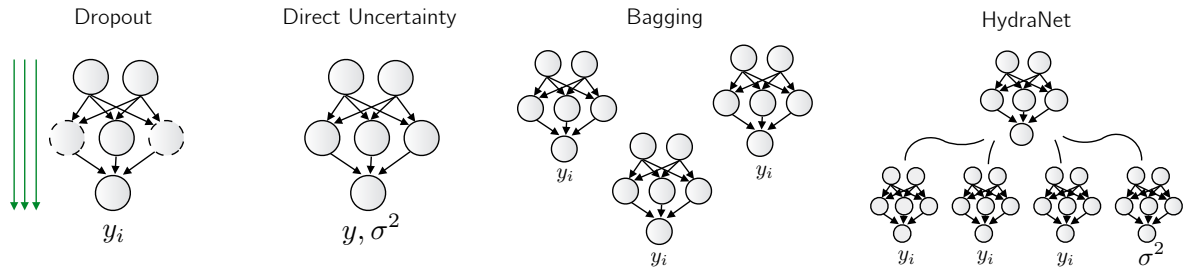


Figure 7.3: Different scalable approaches to neural network uncertainty.

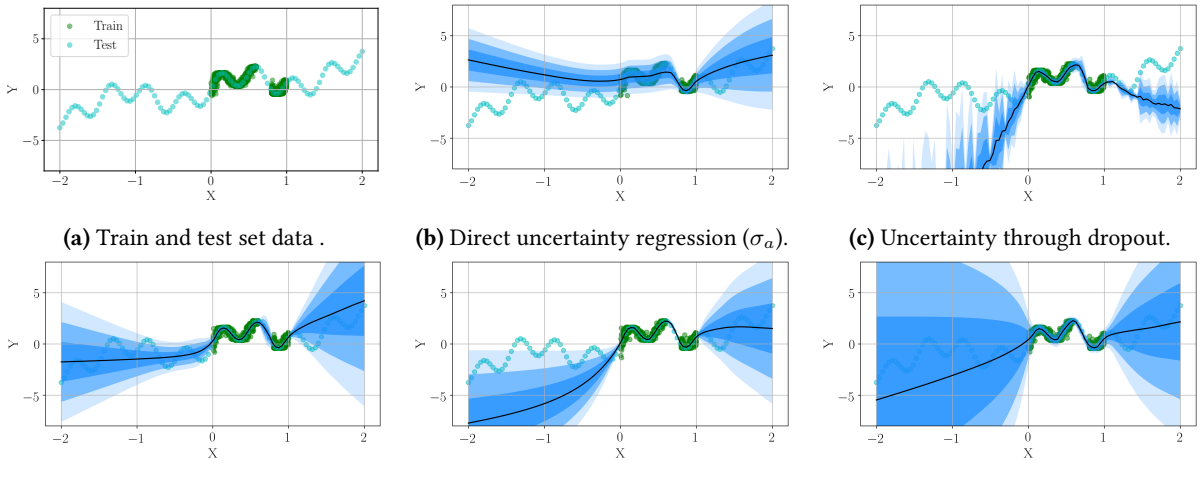
One-dimensional experiment

To build intuition for the advantages of HydraNet over other methods of extracting uncertainty (e.g., uncertainty through MC dropout (Gal and Ghahramani, 2016b)), we constructed an experiment similar to that presented in (Osband et al., 2016). We compared HydraNet to four other approaches: (1) direct aleatoric variance regression where the network outputs a second variance parameter that is constrained to be positive, (2) uncertainty through dropout at test time (Gal and Ghahramani, 2016b), (3) bootstrap aggregation (or bagging) of multiple independent models, and (4) HydraNet with no aleatoric uncertainty output.

For each method, we trained a four-layer fully-connected network to regress the output of a one-dimensional function:

$$y_i = x_i + \sin\left(4(x_i + \omega)\right) + \sin\left(13(x_i + \omega)\right) + \omega, \tag{7.7}$$

where $w \sim \mathcal{N}(\mu = 0, \sigma^2 = 3^2)$. Our training set consisted of 1000 samples randomly drawn from $x \in [0.0, 0.6] \bigcup [0.8, 1.0]$, while the test set consisted of 100 samples uniformly drawn from $x \in [-2, 2]$. The function and the train/test samples are shown in Figure 7.4a.



(a) Train and test set data .

(b) Direct uncertainty regression (σ_a).

(c) Uncertainty through dropout.

(d) Bootstrap aggregation.

(e) HydraNet (no aleatoric uncertainty).

(f) HydraNet.

Figure 7.4: A comparison of different ways to extract uncertainty from deep networks. Each shade of blue represents one standard deviation σ produced by the model.

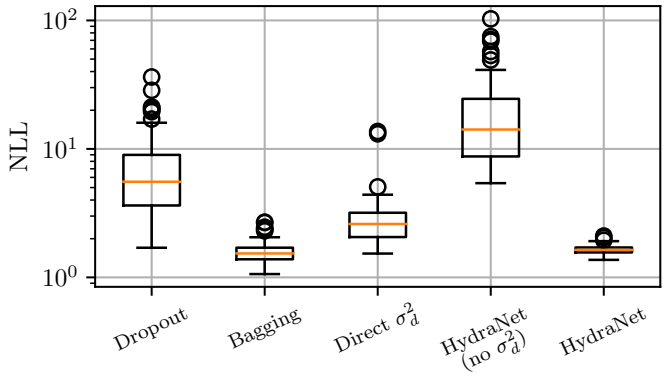


Figure 7.5: Negative log likelihood statistics of 100 repetitions of five neural-network-based uncertainty estimators. HydraNet performs similarly to bagging.

The direct aleatoric uncertainty regression and HydraNet methods were trained using a negative log likelihood loss under the assumption of Gaussian likelihood, while the other methods were trained to minimize mean squared error. We repeated training 100 times, and recorded the test-time negative log likelihood for each method at each repetition. We summarize the results in Figure 7.5. Figure 7.4 presents representative samples from the 100 repetitions for each method. Typically, direct uncertainty regression and dropout are overconfident in the out-of-distribution regions. We replicated the findings of (Osband et al., 2016) who find that uncertainty with dropout does not vary smoothly and can collapse outside of the training distribution. HydraNet combined with direct aleatoric uncertainty learning, however, produced similar excellent likelihoods to bootstrap aggregation without requiring multiple models.

7.4.3 Deep Probabilistic $\mathrm{SO}(3)$ Regression

In order to extend the ideas of HydraNet to the matrix Lie group SO(3), we consider different ways to regress and combine several estimates of rotation.s Given a network, $g(\cdot)$, and an input $\mathcal{I}$, we consider how to extend the ideas of HydraNet to process several outputs, $g_i(\mathcal{I})$, and combine them into an estimate of a 'mean' rotation, $\overline{\mathbf{R}}$, and an associated 3×3 covariance matrix, $\mathbf{\Sigma}$. To produce estimates of rotation for a given HydraNet head, we consider two options. First if $g(\mathcal{I}) \in \mathbb{R}^3$, then we can use the matrix exponential to produce a rotation matrix,

$$\mathbf{R} = \mathrm{Exp}\left(g(\mathcal{I})\right). \tag{7.8}$$

Since the capitalized exponential map $\mathrm{Exp}\left(\cdot\right)$ is surjective (Barfoot, 2017; Solà et al., 2018), this approach can parametrize any valid rotation matrix. Alternatively, if $g(\mathcal{I}) \in \mathbb{R}^4$, we can normalize it to produce a unit quaternion that resides on S^3,

$$\mathbf{q} = \frac{g(\mathcal{I})}{\|g(\mathcal{I})\|}. \tag{7.9}$$

As we noted in Chapter 2, unit quaternions are a double cover of $\mathrm{SO}(3)$, and can represent any rotation. We choose to use this latter parametrization because of its simple analytic mean expression that we describe below.

Rotation Averaging

To produce a mean of several $\mathrm{SO}(3)$ elements (i.e., to evaluate Equation (7.3) for rotations), we turn to the field of rotation averaging (Hartley et al., 2013). Given several estimates of a rotation, we define the mean as the rotation which minimizes some squared metric defined over the group[1],

$$\overline{\mathbf{R}} = \underset{\mathbf{R}\in\mathrm{SO}(3)}{\operatorname{argmin}} \sum_{i=1}^{n} d(\mathbf{R}_i, \mathbf{R})^2. \tag{7.10}$$

There are three common choices for a bijective metric (Hartley et al., 2013; Carlone et al., 2015b) on $\mathrm{SO}(3)$. The angular, chordal and quaternionic:

$$d_{\mathrm{ang}}(\mathbf{R}_a, \mathbf{R}_b) = \left\| \mathrm{Log}\left(\mathbf{R}_a \mathbf{R}_b^T\right) \right\|_2, \tag{7.11}$$

$$d_{\mathrm{chord}}(\mathbf{R}_a, \mathbf{R}_b) = \left\| \mathbf{R}_a - \mathbf{R}_b \right\|_{\mathrm{F}}, \tag{7.12}$$

$$d_{\mathrm{quat}}(\mathbf{q}_a, \mathbf{q}_b) = \min\left(\left\| \mathbf{q}_a - \mathbf{q}_b \right\|_2, \left\| \mathbf{q}_a + \mathbf{q}_b \right\|_2 \right), \tag{7.13}$$

where $\mathrm{Log}\,(\cdot)$, represents the capitalized matrix logarithm (Solà et al., 2018), and $\|\cdot\|_F$ the Frobenius norm. In the context of Equation (7.10), using the angular metric leads to the *Karcher mean*, which requires an iterative solver and has no known analytic expression. Applying the chordal metric leads to an analytic expression for the average but requires the use of Singular Value Decomposition. Using the quaternionic metric, however, leads to a simple, analytic expression for the rotation average as the normalized arithmetic mean of a set of unit quaternions (Hartley et al., 2013),

$$\overline{\mathbf{q}} = \underset{\mathbf{R}(\mathbf{q})\in\mathrm{SO}(3)}{\operatorname{argmin}} \sum_{i=1}^{H} d_{\mathrm{quat}}(\mathbf{q}_i, \mathbf{q})^2 = \frac{\sum_{i=1}^{H} \mathbf{q}_i}{\left\| \sum_{i=1}^{H} \mathbf{q}_i \right\|}. \tag{7.14}$$

This expression is simple to evaluate numerically, and if necessary, can be easily differentiated with respect to its constituent parts. For these reasons, we opt to construct our $\mathrm{SO}(3)$ HydraNet using unit quaternion outputs, and evaluate the rotation average using the quaternionic metric.

SO(3) Uncertainty

There are several ways to approach uncertainty on $\mathrm{SO}(3)$. One method (Carlone et al., 2015a) is to define a probability density directly on the group via the isotropic von Mises-Fisher density. This approach has two downsides: (1) it is isotropic and cannot account for dominant degrees of freedom (e.g.,

[1] Although this is a natural formulation for the rotation mean, it is possible to define other means in terms of absolute errors - see (Hartley et al., 2013).

vehicle yaw during driving), and (2) estimating the concentration parameter requires approximations or iterative solvers.

Instead, we opt to parametrize uncertainty over $\mathrm{SO}(3)$ by injecting uncertainty onto the manifold (Forster et al., 2015; Barfoot and Furgale, 2014; Barfoot, 2017) from a local tangent space about some mean element, $\overline{\mathbf{q}}$,

$$\mathbf{q} = \mathrm{Exp}\left(\boldsymbol{\epsilon}\right) \otimes \overline{\mathbf{q}}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}), \tag{7.15}$$

where $\otimes$ represents quaternion multiplication. In this formulation, $\boldsymbol{\Sigma}$ provides a 3×3 covariance matrix that can express uncertainty in different directions. Further, given a mean rotation, $\overline{\mathbf{q}}$, and samples, $\mathbf{q}_i$, we use the logarithmic map to compute a sample covariance matrix,

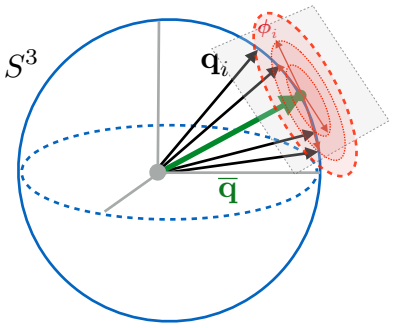


Figure 7.6: We can define uncertainty in the left tangent space of a mean element.

$$\boldsymbol{\Sigma}_e = \frac{1}{H-1} \sum_{i=1}^{H} \boldsymbol{\phi}_i \boldsymbol{\phi}_i^T, \quad \boldsymbol{\phi}_i = \mathrm{Log}\left(\mathbf{q}_i \otimes \overline{\mathbf{q}}^{-1}\right). \tag{7.16}$$

7.4.4 Loss Function

As with one-dimensional HydraNet, we train a direct regression of covariance through a parametrization of positive semi-definite matrices using a Cholesky decomposition[2] (Hu and Kantor, 2015; Haarnoja et al., 2016)). Given the network outputs of a unit quaternion $\mathbf{q}$, and a positive semi-definite matrix Σ, we define a loss function as the negative log likelihood of a given rotation under Equation (7.15) (see (Forster et al., 2015)) for a given target rotation, $\mathbf{q}_t$, as

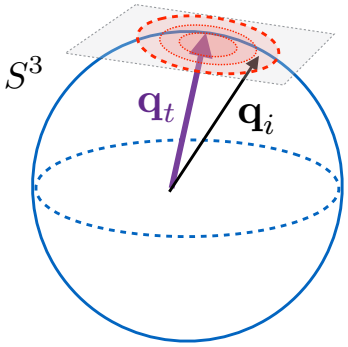


Figure 7.7: We define our negative log likelihood loss in the left tangent space of the target unit quaternion.

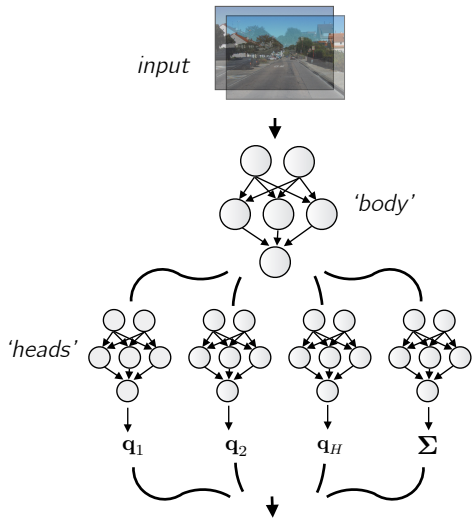
$$\mathcal{L}_{\mathrm{NLL}}(\mathbf{q}, \mathbf{q}_t, \boldsymbol{\Sigma}_a) = \frac{1}{2} \boldsymbol{\phi}^T \boldsymbol{\Sigma}_a^{-1} \boldsymbol{\phi} + \frac{1}{2} \log\det\left(\boldsymbol{\Sigma}_a\right), \tag{7.17}$$

where $\boldsymbol{\phi} = \mathrm{Log}\left(\mathbf{q} \otimes {\mathbf{q}_t}^{-1}\right)$. Combining the sample covariance, with the learned covariance, we extend Equation (7.4) to

$$\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_e + \boldsymbol{\Sigma}_a. \tag{7.18}$$

This covariance estimate is designed to grow for out-of-training-distribution errors (and account for *domain shift* (Lakshminarayanan et al., 2017)) while still accounting for uncertainty within the training set. We note that unlike Bayesian methods, we do not interpret each head as a *sample* from a posterior

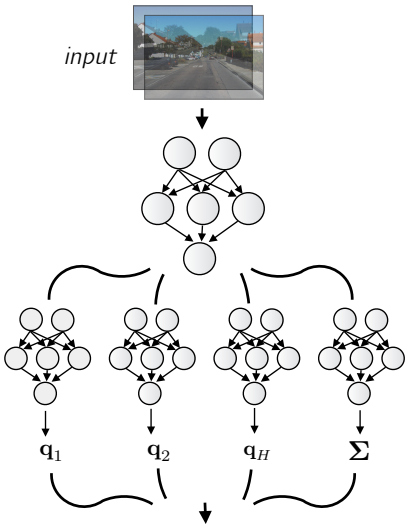
[2]Note that in all the experiments presented in this paper, we omit the off-diagonal components of this covariance and only learn a diagonal matrix with non-negative components.



1. Evaluate loss for H heads: $\mathcal{L} = \sum_{h=1}^{H} \mathcal{L}_{\text{NLL}}(\mathbf{q}_h, \mathbf{q}_t, \boldsymbol{\Sigma})$

$$\mathcal{L}_{\text{NLL}}(\mathbf{q}, \mathbf{q}_t, \boldsymbol{\Sigma}_a) = \frac{1}{2}\boldsymbol{\phi}^{\mathsf{T}}\boldsymbol{\Sigma}_a^{-1}\boldsymbol{\phi} + \frac{1}{2}\log\det(\boldsymbol{\Sigma}_a),$$
$$\boldsymbol{\phi} = \text{Log}\left(\mathbf{q} \otimes \mathbf{q}_t^{-1}\right)$$

(a) Training procedure (Algorithm 4).



1. Compute mean rotation $\bar{\mathbf{q}} = \dfrac{\sum_{h=1}^{H} \mathbf{q}_h}{\left\|\sum_{h=1}^{H} \mathbf{q}_h\right\|}$

2. Compute residuals $\boldsymbol{\phi}_h = \text{Log}\left(\mathbf{q}_h \otimes \bar{\mathbf{q}}^{-1}\right)$

3. Compute covariance as $\boldsymbol{\Sigma} = \dfrac{1}{H-1}\sum_{h=1}^{H} \boldsymbol{\phi}_h \boldsymbol{\phi}_h^{\mathsf{T}} + \boldsymbol{\Sigma}_a$

(b) Testing procedure (Algorithm 5).

Figure 7.8: The training and testing procedures for HydraNet for unit quaternion targets.

distribution[3]. Indeed, we note that in our 1D experiments, the heads have very small variance within the training distribution. The multi-headed structure and rotating averaging serves simply as a way to model epistemic uncertainty when the model encounters inputs that differ from those seen during training. We summarize our training and test procedures in Figure 7.8 as well as Algorithm 4 and Algorithm 5 respectively.

Remark (*Combining epistemic and aleatoric uncertainty*). The simple addition of the two covariance matrices produces a valid covariance matrix (and follows prior work Kendall and Gal (2017) on combining aleatoric and epistemic sources). We leave an investigation of other possible way to combine these sources to future work.

7.5 Experiments

7.5.1 Uncertainty Evaluation: Synthetic Data

Before we embarked on training with real data, we analyzed our proposed HydraNet structure on a synthetic world. Our goal was to produce probabilistic estimates of camera orientation based on noisy

[3]Notably, this means we do not scale our direct uncertainty when averaging as $\frac{1}{H}\boldsymbol{\Sigma}_a$.

Algorithm 4 Supervised training for $\mathrm{SO}(3)$ regression

Require: Training data $\mathcal{T}$, training targets $\mathbf{q}_t$, untrained model $g_\theta(\cdot)$ with parameters θ and $H+1$ heads

Ensure: Probabilistic regression model $g_\theta(\cdot)$

function TRAINHYDRANET($\mathcal{T}$)
  for each mini-batch $\mathcal{T}_i$ **do**
    Output $\mathbf{\Sigma}_a$ ▷ *1st head, Chol. decom.*
    for heads $2...(H+1)$ in g **do**
      Output $\mathbf{q}_h$ ▷ *Equation* (7.9)
      Evaluate NLL loss ▷ *Equation* (7.17)
    end forend
    Backprop, update θ
  end forend
  return $g(\cdot)$
end function

Algorithm 5 Testing of $\mathrm{SO}(3)$ regression

Require: Test sample $\mathcal{I}_j$, trained model $g_\theta(\cdot)$

Ensure: Test prediction $\mathbf{q}$, covariance $\mathbf{\Sigma}_t \succcurlyeq 0$

function TESTHYDRANET($\mathcal{I}_j$, $g_\theta(\cdot)$)
  Output $\mathbf{\Sigma}_a$ ▷ *1st head, Chol. decom.*
  for heads $2...(H+1)$ in g **do**
    Output $\mathbf{q}_h$ ▷ *Equation* (7.9)
  end forend
  Compute $\overline{\mathbf{q}}$ ▷ *Equation* (7.14)
  Compute $\mathbf{\Sigma}_e$ ▷ *Equation* (7.16)
  return $\overline{\mathbf{q}}$, $\mathbf{\Sigma}_e + \mathbf{\Sigma}_a$
end function

pixel coordinates of a set of fixed point landmarks. To accomplish this, we simulated a monocular camera observing a planar grid of evenly spaced (see Figure 7.9a) landmarks from a hemisphere surrounding the grid. We aligned the monocular camera's optical axis with the centre of the hemisphere so that all landmarks were visible in every camera pose. At each pose, we computed noisy pixel locations of the projection of every landmark, and stacked these 2D locations as an input vector. We generated 15000 training samples with poses that were randomly sampled from the hemisphere in the polar angle range of $[-60, 60]$ degrees. For testing, we sampled 500 poses in the range of $[-80, 80]$ degrees, purposely widening the range to include orientations that were not part of training.

To regress the camera orientation, we constructed a five layer residual network and attached 26 heads (25 + 1 for direct uncertainty learning) to regress a probabilistic estimate of $\mathbf{q}_{c,w}$, the orientation of the camera with respect to the world frame.

Figure 7.9b plots rotational errors $\boldsymbol{\phi} = \text{Log}\left(\mathbf{q} \otimes \mathbf{q}_t^{-1}\right)$ along with 3 sigma bounds based on both the total covariance, $\boldsymbol{\Sigma}_t$, and the direct covariance $\boldsymbol{\Sigma}_a$. The final regression estimates have consistent uncertainty, composed of a static aleatoric uncertainty and an epistemic uncertainty (Equation (7.16)) that grows when the test samples come from unfamiliar input data.

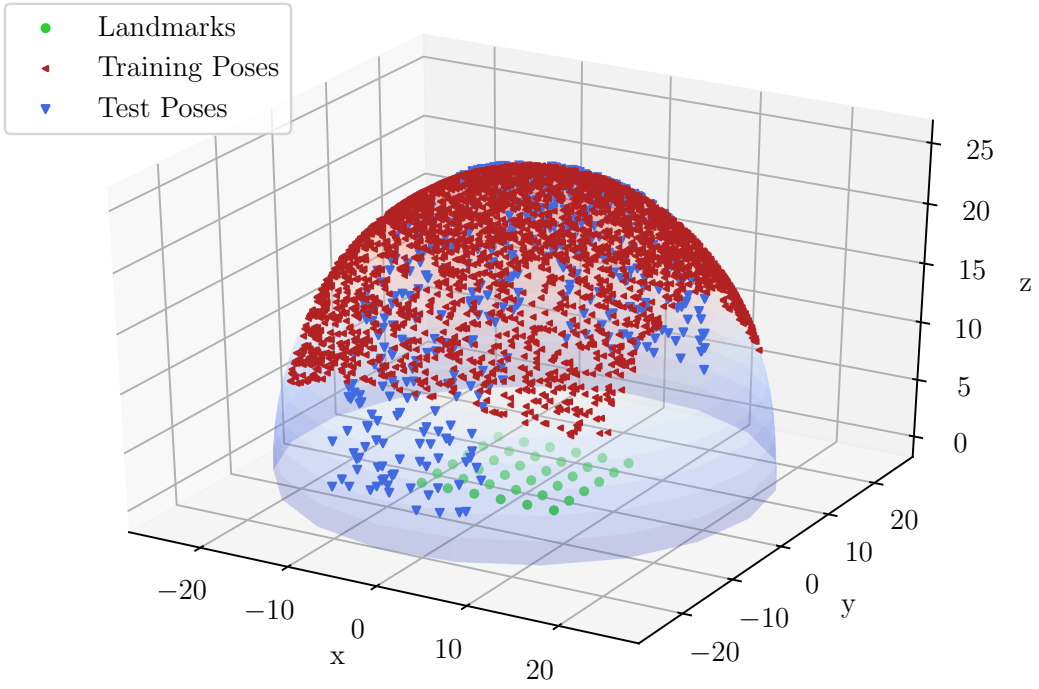
7.5.2 Absolute Orientation: 7-Scenes

Next, we used HydraNet to regress absolute orientations from RGB images from the 7-Scenes dataset (Glocker et al., 2013). Our goal was to achieve similar errors to other regression techniques (Kendall and Cipolla, 2017) but augment them with consistent covariance estimates. For this experiment, we used `resnet34` (He et al., 2016) (pre-trained on the ImageNet dataset) for the body of HydraNet and attached 25 HydraNet heads, each consisting of two fully connected layers. We cropped and resized all RGB images to match the expected ImageNet size and omitted the depth channel.

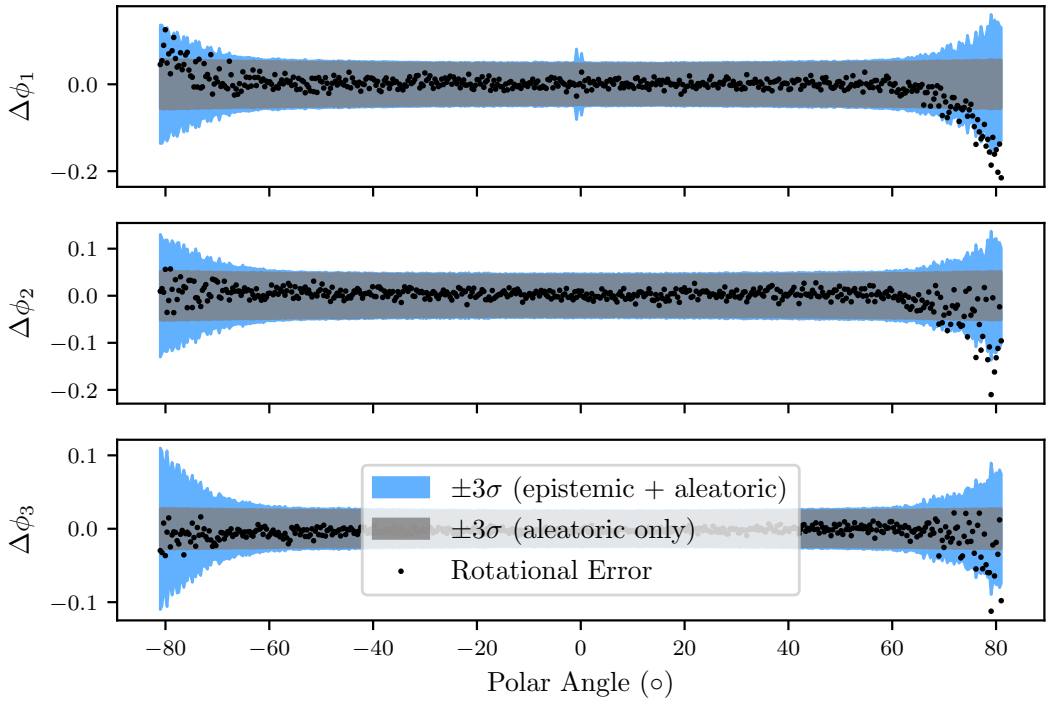
Table 7.1 presents the mean angular errors and negative log likelihoods achieved by our method. The HydraNet-based network produces similar angular errors to other regression methods (Kendall and Cipolla, 2017) but with additional benefit of consistent three-degree-of-freedom uncertainty. Note that we spent little time optimizing the network itself, and note that state-of-the art errors can be achieved using more sophisticated pixel-based losses (Brachmann and Rother, 2018). However, the general HydraNet structure and loss can be used whenever a probabilistic rotation output is required. Further, our results show that our covariance formulation can be used for 'large' rotation elements, where techniques (e.g., (Peretroukhin and Kelly, 2018)) that assume 'small' corrections may fail.

7.5.3 Relative Rotation: KITTI Visual Odometry

Finally, to show the benefit of fusing deep probabilistic estimates with classical estimators, we trained a network to estimate relative frame-to-frame rotations on the KITTI visual odometry (VO) benchmark. To regress relative rotations, we use the HydraNet-based network described in Figure 7.2. For each pair of poses, we process two RGB images (taken from the left RGB camera) into a two channel dense optical flow image using a fast classical algorithm (Farnebäck, 2003). Compared to using raw images, we found that using the optical flow pre-processing greatly improved training robustness and rotation



(a) Synthetic world used to illustrate our method. A monocular camera observes a 6×6 grid of point landmarks from poses sampled on a semi-sphere. The test set includes poses that are outside the training distribution.



(b) Rotation estimation errors for a deep network trained using our HydraNet approach on synthetic data (noisy pixel locations of 36 landmarks). We note that outside of the training distribution, our epistemic uncertainty ($\boldsymbol{\Sigma}_e$) grows, as expected.

Figure 7.9: Synthetic experiments of probabilistic rotation regression with HydraNet.

Table 7.1: HydraNet regression results for the 7scenes dataset compared to results reported in (Kendall and Cipolla, 2017). We report mean angular errors and the negative log likelihood (lower is better).

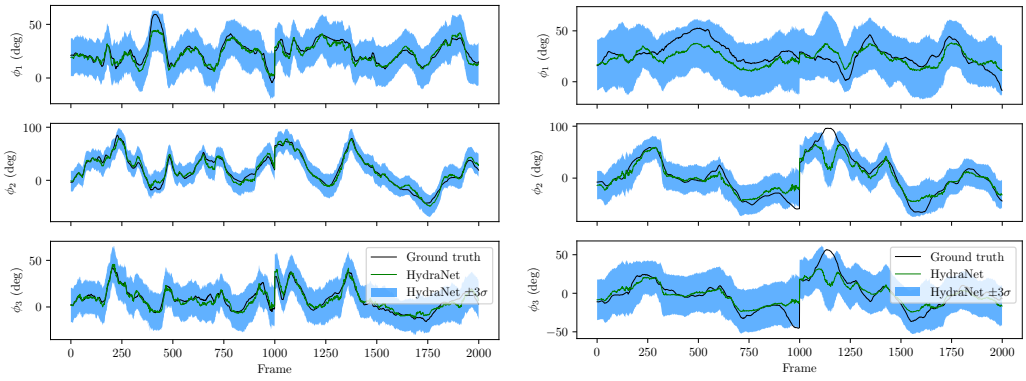
	Error (deg)		NLL	
Scene	HydraNet	PoseNet	HydraNet	PoseNet
Chess	6.3	4.5	-6.0	—
Fire	14.9	11.3	-3.6	—
Heads	14.3	13.0	-3.9	—
Office	8.6	5.6	-5.4	—
Pumpkin	9.0	4.8	-5.0	—
Kitchen	8.8	5.4	-5.0	—
Stairs	11.8	12.4	-4.7	—

Table 7.2: Results of fusing HydraNet relative rotation regression with classical stereo visual odometry.

		m-ATE		Mean Segment Errors	
Sequence (Length)	Estimator	Translation (m)	Rotation (°)	Translation (%)	Rotation (°/100m)
00 (3.7 km)	DeepVO (Wang et al., 2017b)	—	—	—	—
	SfMLearner (Zhou et al., 2017)	—	—	65.27	6.23
	UnDeepVO (Li et al., 2017b)	—	—	4.14	1.92
	`viso2-s`	27.91	6.25	1.96	0.81
	`viso2-s` + HydraNet	9.86	2.83	1.34	0.63
	Keyframe Direct VO	12.41	2.45	1.28	0.54
02 (5.1 km)	DeepVO	—	—	—	—
	SfMLearner	—	—	57.59	4.09
	UnDeepVO	—	—	5.58	2.44
	`viso2-s`	64.67	8.45	1.47	0.56
	`viso2-s` + HydraNet	50.19	6.51	1.47	0.63
	Keyframe Direct VO	16.33	3.19	1.21	0.47
05 (2.2 km)	DeepVO	—	—	2.62	3.61
	SfMLearner	—	—	16.76	4.06
	UnDeepVO	—	—	3.40	1.50
	`viso2-s`	23.72	8.10	1.79	0.79
	`viso2-s` + HydraNet	9.85	3.23	1.38	0.60
	Keyframe Direct VO	5.83	2.05	0.69	0.32

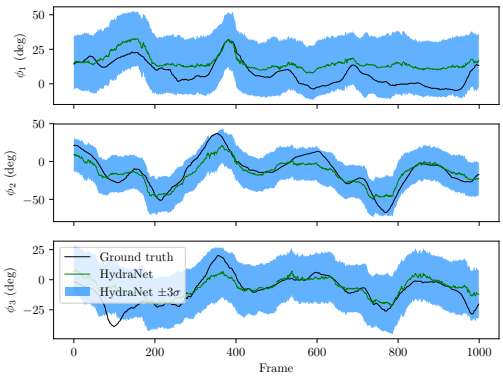
Table 7.3: HydraNet regression results for the KITTI odometry dataset. We report mean angular errors and the negative log likelihood (lower is better).

Sequence	Mean Angular Error (°)	NLL
00	0.199	-16.84
02	0.138	-18.44
05	0.109	-19.31

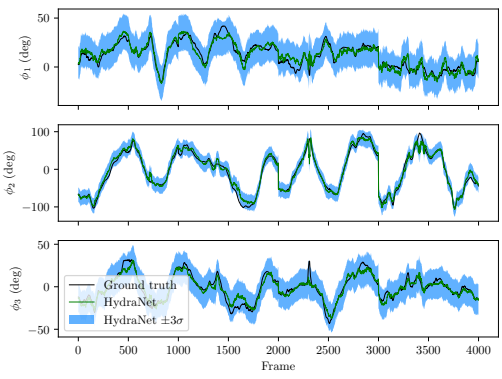


(a) Chess

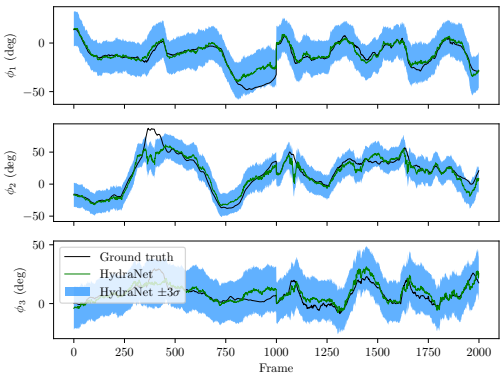
(b) Fire



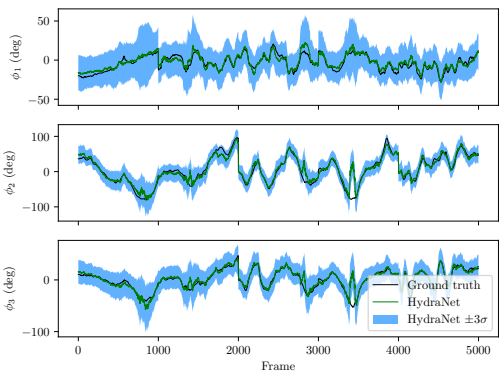
(c) Heads



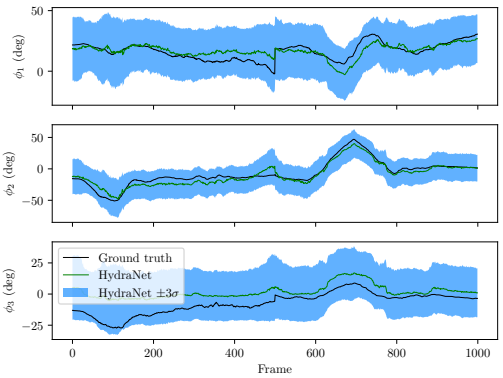
(d) Office



(e) Pumpkin

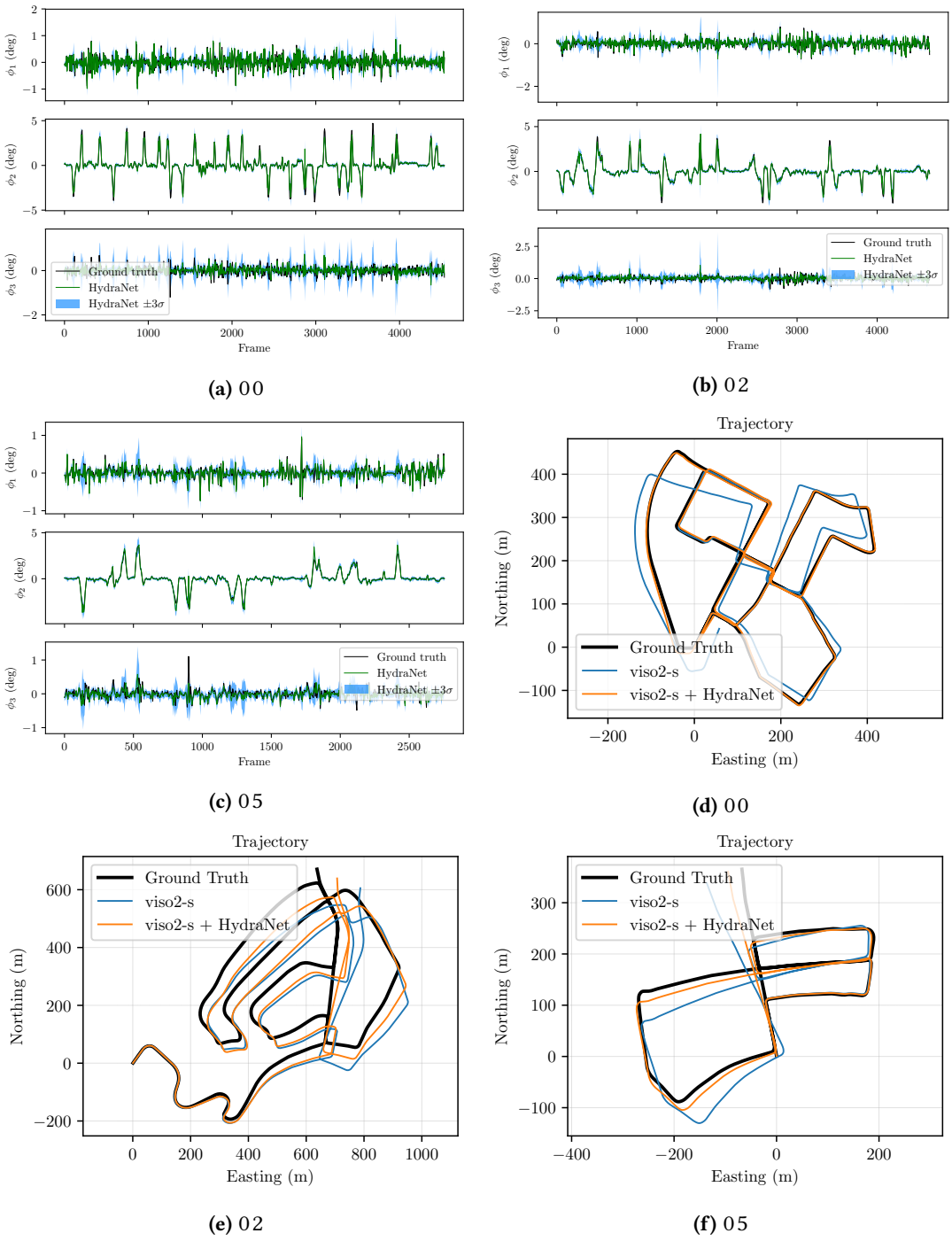


(f) Kitchen



(g) Stairs

Figure 7.10: Probabilistic regression plots for all seven datasets from the 7-Scenes dataset.



(a) 00 **(b)** 02

(c) 05 **(d)** 00

(e) 02 **(f)** 05

Figure 7.11: KITTI frame-to-frame rotation probabilistic regression for sequences 00, 02 and 05. Top-down trajectory plots show localization improvements after fusion with a classical stereo visual odometry pipeline.

accuracy. Since we use two-channel flow images, the body of the network is not pre-trained and instead contains an eight layer convolutional network (see supplemental materials for additional details). We maintained the same head structure as the 7-Scenes experiment. Table 7.3 and Figure 7.12 detail the mean test error and negative log likelihood for KITTI odometry sequences 00, 02 and 05 (chosen for

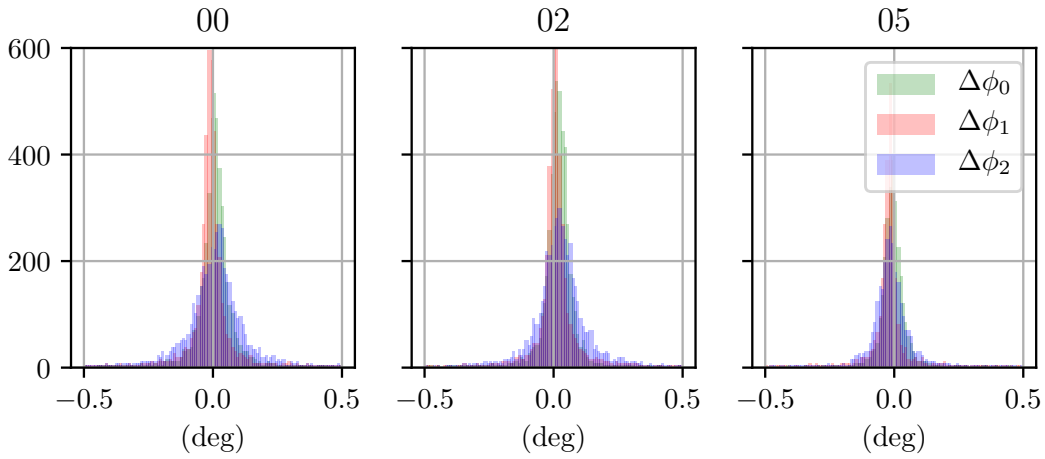


Figure 7.12: Error histograms for test KITTI sequences 00, 02, and 05 on three rotational axes.

their complexity and length). For each sequence, we trained the model on the remaining sequences in the benchmark. We found our model produced mean errors of approximately 0.1 degrees on all three test sequences. The covariance produced by HydraNet was consistent, spiking during yawing motions when the largest errors occurred (see Figure 7.11). Despite its consistency, the network covariance was dominated by $\boldsymbol{\Sigma}_a$. We suspect that unlike the synthetic data, $\boldsymbol{\Sigma}_e$ remained small throughout the tests sets due to a more constrained input space (RGB or flow images, compared to pixel locations), but leave a thorough investigation to future work.

Canonical Indirect Stereo Visual Odometry

For the classical visual odometry estimator, we use a similar pipeline to that used in the prior two chapters based on the open-source `viso2` package (Geiger et al., 2011) to detect and track sparse stereo image key-points. In brief, our pipeline modelled stereo re-projection errors, $\mathbf{e}_{l,t_i}$, as zero-mean Gaussians with a known static covariance, $\mathbf{R}$. To generate an initial guess and to reject outliers, we used three point Random Sample Consensus (RANSAC) based on stereo re-projection error. Finally, we solved for the maximum likelihood transform, $\mathbf{T}_t^*$, through a Gauss-Newton minimization of

$$\mathbf{T}_t^* = \underset{\mathbf{T}_t \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} \mathbf{e}_i^T \mathbf{R}^{-1} \mathbf{e}_i. \tag{7.19}$$

After convergence, we approximate the frame-to-frame transformation uncertainty as (Barfoot, 2017):

$$\boldsymbol{\Sigma}_{\text{vo}} \approx \left(\sum_{l=1}^{N_t} \mathbf{J}_{\mathbf{e}_i}^T \mathbf{R}^{-1} \mathbf{J}_{\mathbf{e}_i} \right)^{-1}, \tag{7.20}$$

where $\mathbf{J}_{\mathbf{e}_i}$ refers to the Jacobian of each reprojection error.

Fusion via Graph Relaxation

To fuse the output of our HydraNet pseudo-sensor with our canonical VO pipeline, we used pose graph relaxation. We describe our method briefly and refer the reader to (Barfoot, 2017) for a more detailed treatment. For every two poses, we defined a loss function based on a contribution from the estimator and from the network, weighed by their respective covariances:

$$\mathbf{T}_{1,w}^{*}, \mathbf{T}_{2,w}^{*} = \underset{\mathbf{T}_{1,w}, \mathbf{T}_{2,w} \in \mathrm{SE}(3)}{\operatorname{argmin}} \mathcal{L}(\hat{\mathbf{T}}_{2,1}, \hat{\mathbf{C}}_{2,1}) \tag{7.21}$$

$$= \delta\boldsymbol{\xi}_{1,2}^{T} \boldsymbol{\Sigma}_{\mathrm{vo}}^{-1} \delta\boldsymbol{\xi}_{1,2} + \delta\boldsymbol{\phi}_{1,2}^{T} \boldsymbol{\Sigma}_{\mathrm{hn}}^{-1} \delta\boldsymbol{\phi}_{1,2} \tag{7.22}$$

where $\delta\boldsymbol{\xi}_{1,2} = \operatorname{Log}\left(\left(\mathbf{T}_{2,w}\mathbf{T}_{1,w}^{-1}\right)\hat{\mathbf{T}}_{2,1}^{-1}\right)$ and $\delta\boldsymbol{\phi}_{1,2} = \operatorname{Log}\left(\left(\mathbf{C}_{2,w}\mathbf{C}_{1,w}^{T}\right)\hat{\mathbf{C}}_{2,1}^{T}\right)$. The estimates $\hat{\mathbf{T}}_{2,1}$, $\boldsymbol{\Sigma}_{\mathrm{vo}}$ and $\hat{\mathbf{C}}_{2,1}$, $\boldsymbol{\Sigma}_{\mathrm{hn}}$ are provided by our classical estimator and the HydraNet network respectively.

Table 7.2 summarizes the results when we perform this fusion - and Figure 7.11 shows the final effect on the trajectory for sequence 00. We found that fusing deep rotation regression with classical methods results in motion estimates that significantly out-perform other methods that rely on deep regression alone. However, we note that even with consistent estimates, a small bias can affect the final fused estimates (e.g., sequence 05) and removing bias is an important avenue for future work. Further, the KITTI dataset contains few deleterious effects that negatively affect classical algorithms, and therefore we expect that this fusion would produce even more pronounced improvements on more varied visual data.

Remark (*Improving Uncertainty Estimates*). There are several salient extensions to the work presented here.

1. In order to ensure that the output of HydraNet and the output of the canonical VO pipeline are fused optimally, we can apply the method of *covariance intersection* (CI) (Julier and Uhlmann, 2007). CI is a technique to fuse measurements when the correlation between them is unknown and provides provably consistent estimates.

2. To improve epistemic uncertainty, we can investigate the application of a *gradient blockade* between the heads and body of HydraNet. A gradient blockade would ensure that each head learns independently. At present, the $H + 1$th head (which outputs *aleatoric* covariance) indirectly connects the gradients of the remaining heads by weighting the likelihood loss (Brachmann and Rother, 2019) for each.

3. To further improve epistemic uncertainty, it is possible (as in Osband et al. (2016)) to train each head with a subset of training examples (mimicking the method of the statistical bootstrap, instead of relying solely on random initializations).

7.6 Summary

In this chapter, we described a method (our final *pseudo-sensor*) to regress probabilistic estimates of rotation using a deep multi-headed network structure. We used the quaternionic metric on $\mathrm{SO}(3)$ to define a rotation average, and extracted anisotropic covariances by modelling uncertainty through noise injection on the manifold.

Our novel contributions were

1. a deep network structure we call *HydraNet* that builds on prior work (Lakshminarayanan et al., 2017; Osband et al., 2016) to produce meaningful uncertainties over unconstrained targets,
2. a loss formulation and mathematical framework that extends HydraNet to means and covariances of the rotation group $\mathrm{SO}(3)$,
3. and open source code for $\mathrm{SO}(3)$ regression[4].

[4] https://github.com/utiasSTARS/so3_learning

Chapter 8

Conclusion

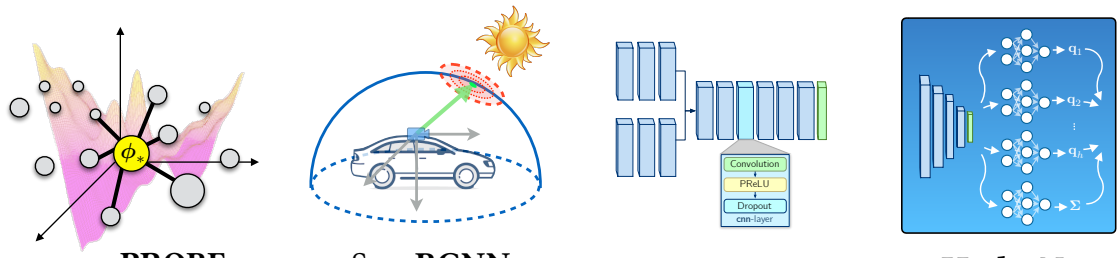
> What we call the beginning is often the end.
> And to make an end is to make a beginning.
>
> T.S. Eliot

This dissertation has dealt with the development of a general framework for improving the performance of model-based visual odometry pipelines through learned probabilistic pseudo-sensors that extract difficult-to-model latent information. We presented four examples of such *pseudo-sensors*. We close with a final summary of novel contributions associated with each chapter, a discussion of potential future work, and some concluding remarks.

8.1 Summary of Contributions

8.1.1 Predictive Robust Estimation

We began with a pseudo-sensor that used a heteroscedastic noise model to enable predictively robust estimation. PROBE contributed



(a) PROBE (Chapter 4).

(b) Sun-BCNN (Chapter 5).

(c) DPC-Net (Chapter 6).

(d) HydraNet (Chapter 7).

Figure 8.1: Four pseudo-sensors presented in this dissertation.

1. a probabilistic model for indirect stereo visual odometry, leading to a predictive robust algorithm for inference on that model,
2. two different approaches to constructing the robust algorithm: one baseds on k-nearest neighbours (Appendix A), and one based on Generalized Kernel (GK) estimation (Chapter 4),
3. a procedure for training our model using pairs of stereo images with known relative transforms, and
4. an iterative, expectation-maximization approach to train our GK model when the relative ground truth egomotion was unavailable.

8.1.2 Sun BCNN

With Sun-BCNN, we applied learned *pseudo-sensors* to the problem of illumination direction in outdoor environments. In sum, the novel contributions were:

1. the application of a Bayesian CNN to the problem of sun direction estimation, incorporating the resulting covariance estimates into a visual odometry pipeline;
2. an empirical demonstration that a Bayesian CNN with dropout layers after each convolutional and fully-connected layer can achieve state-of-the-art accuracy at test time;
3. a loss function that incorporated a 3D unit-length sun direction vector, appropriate for full 6-DOF pose estimation;
4. experimental results on over 30 km of visual navigation data in urban (Geiger et al., 2013) and planetary analogue (Furgale et al., 2012) environments;
5. an investigation into the sensitivity of the Bayesian CNN-based sun estimate to cloud cover, camera and environment changes, and measurement parameterization; and
6. open-source software[1].

8.1.3 Deep Pose Corrections

Next, we generalized the results of Sun-BCNN to learn full six degree-of-freedom corrections for a particular egomotion pipeline and a given environment with DPC-Net. Our contributions included

1. the formulation of a novel deep corrective approach to egomotion estimation,
2. a novel cost function for deep $\mathrm{SE}(3)$ regression that naturally balances translation and rotation errors, and
3. an open-source implementation of DPC-Net in `PyTorch`[2].

[1] https://github.com/utiasSTARS/sun-bcnn-vo.
[2] See https://github.com/utiasSTARS/dpc-net.

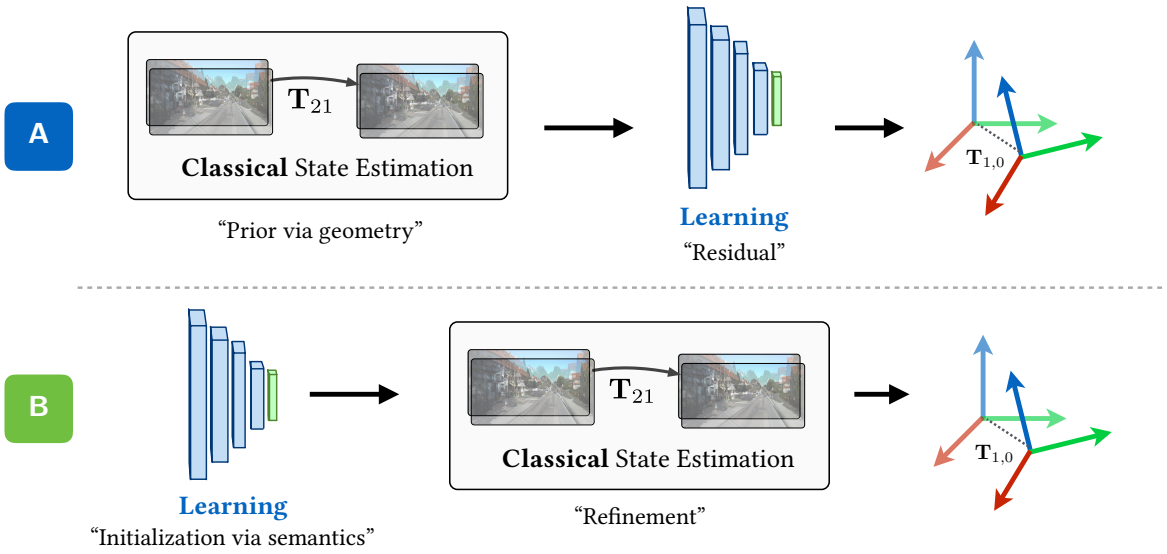


Figure 8.2: Two different ways to incorporate learning with classical pipelines. The first is by using pipelines as a *prior* which can then be corrected by learned approaches, while the second is use learning as an initialization which can then be *refined* by classical techniques.

8.1.4 Deep Probabilistic Inference of $\mathrm{SO}(3)$ with HydraNet

Finally, we applied the lessons of DPC-Net and Sun-BCNN to learning only rotation estimates through a network structure that incorporated both aleatoric and epistemic uncertainty which can be fused with classical pipelines through pose graph optimization. With this work, we contributed

1. a deep network structure we called *HydraNet* that builds on prior work to produce meaningful uncertainties (including both aleatoric and epistemic components) over unconstrained targets,
2. a loss formulation and mathematical framework that extends HydraNet to means and covariances of the rotation group $\mathrm{SO}(3)$,
3. and open source code for $\mathrm{SO}(3)$ regression[3].

8.2 Future Work

There are many potential avenues for future work. For instance, although the fusion of pipelines with pseudo-sensors can significantly improve localization performance in a given environment, there are few guarantees that the final estimates are accurate and consistent.

One promising thread of future work addresses this deficiency by developing more tightly-coupled perception systems that can be 'certified' to produce globally-optimal solutions while still being robust to adverse environmental effects. To do this, we propose a second way to fuse learned models with classical optimization techniques. Instead of treating the learning component as a way to reduce error

[3] https://github.com/utiasSTARS/so3_learning.

of an extant classical technique (like Chapters 5 to 7), we propose that we use modern deep learning approaches as initializations, or priors, that are more invariant to effects like large view-point changes and have the ability to generalize to large-scale environments (Figure 8.2). This is inline with the way our earlier work, PROBE (Chapter 4), uses non-parametric learning. Namely, we believe there are many potential fruitful applications where deep networks are used as input to parametrized loss functions that are then solved through convex optimization (e.g., Amos and Kolter (2017)). In this way, we can leverage recently-developed theory on convex relaxations (e.g., that presented in Rosen et al. (2019)) to ensure that the final localization and mapping results are 'optimal', in the sense that they are the global minima of a maximum-likelihood-based loss, and 'robust', in the sense that they provide consistent estimates in the presence of adverse conditions.

Remark (*Dissertation as Dialectic*). The German Philosopher Georg Hegel had a particular dialectic method that involved a triad: a thesis, antithesis and synthesis. Somewhat curiously, this *thesis* has been an attempt at a *synthesis* of the thesis posed by classical visual egomotion, and the antithesis posed by data-driven end-to-end learning techniques. My hope is that the synthesis proposed by the paradigm of learned *pseudo-sensors* will prove to be a fruitful one within the field of state-estimation, and for the broader robotics community.

8.3 Coda: In Search of Elegance

> Machinery that gives us abundance has left us in want. Our knowledge has made us cynical. Our cleverness, hard and unkind. We think too much, and feel too little. More than machinery, we need humanity.
>
> Charlie Chaplin, *The Great Dictator*

Looking back on my academic journey, I see a path from a fascination with the possible applications of autonomous systems to a fascination with autonomy *in-and-of-itself*. As a budding researcher, I saw robust, accurate perception as a means towards an end. An end which entailed truly autonomous systems 'perceiving' and 'interpreting' their surroundings with the goal of exploring distant planets and navigating busy urban streets. Now, however, I see 'perception' as an end in itself with a plethora of fascinating mathematical, philosophical and ethical challenges that can be tackled in light of, but not subservient to, the potential goals of some grander autonomous system. Throughout this transition, I have become more interested not only in the flesh and blood of perception systems, but also in the *spirit* of them. If perception is one of the bridges we must build to reach the land of autonomy, I am concerned not only with the structural integrity that lets us cross it today, but also with an elegance and rigour that lets it serve as a model for posterity.

With this in mind, I want to address a concerning shift that has occurred in the research community throughout my academic career. Many researchers who work on algorithms that enable autonomy (not only in perception, but also in planning and controls) have given up on the dream of modelling the world with the tools of Euclid, Newton and Euler in favour of methods that rely on exemplary data to 'train' arbitrarily complex predictive black-box models. In my estimation, this shift has brought with it a certain sense of resignation to the overwhelming complexity of the world. We are often content to use vague notions of complexity as reason to avoid building analytic models. Instead, we turn to crude, inscrutable surrogates of our own brains to model what we do not want to. This, I believe, is a tempting mistake. Although these solutions may serve as useful tools to temporarily bridge gaps in our understanding of the world, we will inevitably deplete the low-hanging empirical fruits that they can bear, and we will be left with a deep sense of dissatisfaction that only elegance and simplicity can fill.

I am certainly not the first or the last person who has taken issue with data-driven methods. Noam Chomsky gives the following critique of purely statistical approaches to science[4]. Consider the study of bee colonies. In order to to extract interpretable models of their behaviour (e.g., there is a queen bee, there are worker bees, etc.), one has to observe these colonies meticulously over generations. So why not avoid that entire endeavour and use a data-driven approach? We could set up a camera to observe a bee colony and collect data over several years. By tracking each bee, we could use the tools of modern machine learning to construct and train a large parametric model of each of their positions. Once complete, we could then query this model with a new image from our camera and recover, with extreme precision, the predicted location of each bee. This may allow us to improve honey production, but what have we learned? Is this an elegant model of bee behaviour? Have we not just transformed the problem of understanding the bees into one of understanding this surrogate model? Now consider doing the same with celestial objects–Kepler be damned!

Some may argue that the entire goal of science is predicting the future states of nature, so elegance is irrelevant. I vehemently disagree. I would rather have an interpretable model that is wrong during specific situations (where I can verify that certain assumptions are violated), rather than an obfuscated model which has vague limits to its predictive power.

If history is any judge, the models that stand the test of time are ones that are born out of our meticulous labour and enlightened insight to extract salient principles out of the complexity of the world. The hope that this labour can be replaced with black-box surrogates that indirectly learn these same principles is troubling and, in my view, unnecessary. No matter how much anthropomorphic language we use to describe these surrogates (endowing them with 'understanding', 'attention', and 'forgetfulness'), they will always be limited by our own ability to collect sufficient data, and by our ability to craft them in such a way as to consume significant amounts of training exemplars without 'overfitting' to them. What's more, if these models have any interaction with the world, they will also affect the world, and we are committing ourselves to an endless game of cat-and-mouse. Although it may seem that our time is best spent crafting ever-more-clever surrogates, we will soon reach a point

[4] http://norvig.com/chomsky.html

where we would be better off using the time and resources towards studying a particular problem more directly.

I do not want to cast aspersions flippantly. The transition to data-driven approaches in computer vision happened for good reason and with much hesitation. The elegance of analytic models has historically only been exceeded by their inability to model the often inelegant 'real world'. At the turn of the twenty-first century, roboticists were joking that the dirty secret of much of computer vision is that it doesn't work. Recent efforts into combining the connectionist ideas of the 20th century with the computational power accessible in the 21st have undoubtedly created systems that do attain impressive empirical results, and there is a constant stream of new theoretical insights into the types of structures and optimization methods that work well in a given domain.

However, as the world becomes more connected and complex with every passing day, I think it is of utmost importance that autonomy researchers are not tempted to focus solely on empirical results at the cost of elegant solutions. It is now well-accepted that data-driven methods are not the panacea (like it might have seemed for a brief moment a few years ago) to all problems in autonomy. However, this passive agreement may not be enough. Instead, we need to actively suppress the urge to try and solve a problem first through general 'learning' methods that are becoming more and more easy to implement and less and less easy to understand. We do not need to relinquish the dream of understanding the world and relegate ourselves to simply predicting it by any means possible. We can instead strive to simplify it and interpret it. If after significant effort we fail at that goal, and only then, should we turn to data-driven learned models to fill in the gaps in our understanding.

Appendices

Appendix A

PROBE: Isotropic Covariance Models through k-Nearest Neighbours

A.1 Introduction

This appendix presents our initial exploratory work on isotropic covariance modelling for image features through k-nearest-neighbours. Unlike the work we presented in Chapter 4, here we opt to learn a scalar weight, β_i, that represents the *quality* of image features. We use β_i to scale the covariance of each feature during non-linear optimization. We learn this quality (β_i) *indirectly* by computing an egomotion estimate with a small subset of visual features and then storing the resulting position error in a prediction space. We repeat this for a fixed amount of iterations. During testing, we map features into this prediction space and compute a β_i based on the K nearest errors in this prediction space. Our framework is flexible enough that we do not require ground truth at every image and we can, potentially, learn the model based on a single loop closure error.

Remark (*Associated Publication*). This initial work is associated with the following publication:

- Peretroukhin, V., Clement, L., Giamou, M., and Kelly, J. (2015a). PROBE: Predictive robust estimation for visual-inertial navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'15)*, pages 3668–3675, Hamburg, Germany.

A.2 Theory

To solve for the relative egomotion, $\mathbf{T}_t$, between two camera frames, $\underrightarrow{\mathcal{F}}_{c_0}$ and $\underrightarrow{\mathcal{F}}_{c_1}$, we follow the technique described in Section 3.1.3 to convert stereo observations into point-clouds and then solve for the maximum likelihood $\mathrm{SE}(3)$ transformation. We associate with each match $\{\mathbf{y}_{i,c_0}, \mathbf{y}_{i,c_1}\}$ a vector of *predictors*, $\boldsymbol{\phi}_{i,t}$. We compute the covariance as a function of these predictors, so that $\mathbf{R}_{i,c_0} =$

$\mathbf{R}_{i,c_1} = \mathbf{R}_{i,t} = \mathbf{R}(\boldsymbol{\phi}_{i,t})$, and we use the same covariance function for features in both frames[1],

$$\mathbf{y}_{i,c_0} \sim \mathcal{N}\left(\bar{\mathbf{y}}_{i,c_0}, \mathbf{R}_{i,t}\right) = \mathcal{N}\left(\bar{\mathbf{y}}_{i,c_0}, \mathbf{R}(\boldsymbol{\phi}_{i,t})\right) \tag{A.1}$$

$$\mathbf{y}_{i,c_1}, \sim \mathcal{N}\left(\bar{\mathbf{y}}_{i,c_1}, \mathbf{R}_{i,t}\right) = \mathcal{N}\left(\bar{\mathbf{y}}_{i,c_1}, \mathbf{R}(\boldsymbol{\phi}_{i,t})\right). \tag{A.2}$$

This then builds the following weighted least squares objective,

$$\mathbf{T}_t^* = \underset{\mathbf{T}\in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} \mathbf{e}_i(\mathbf{T}_t)^T \boldsymbol{\Sigma}_{i,t}^{-1} \mathbf{e}_i(\mathbf{T}_t). \tag{A.3}$$

where $\boldsymbol{\Sigma}_{i,t}$ is now given by,

$$\boldsymbol{\Sigma}_{i,t} = \mathbf{D}\mathbf{G}_{i,c_1}\mathbf{R}(\boldsymbol{\phi}_{i,t})\mathbf{G}_{i,c_1}^T\mathbf{D}^T + \mathbf{D}\mathbf{T}_t\mathbf{G}_{i,c_0}\mathbf{R}(\boldsymbol{\phi}_{i,t})\mathbf{G}_{i,c_0}^T\mathbf{T}_t^T\mathbf{D}^T \tag{A.4}$$

We build a model for $\mathbf{R}(\boldsymbol{\phi}_{i,t})$ as,

$$\mathbf{R}(\boldsymbol{\phi}_{i,c}) = \beta(\boldsymbol{\phi}_{i,t})\bar{\mathbf{R}}, \tag{A.5}$$

with

$$\beta(\boldsymbol{\phi}_{i,c}) = \left(\frac{1}{\epsilon_{\text{avg}}K}\sum_{k=1}^{K}\epsilon_k\right)^{\gamma},\ \epsilon_k \in \text{K-NN}(\boldsymbol{\phi}_{i,t}), \tag{A.6}$$

where $\{\epsilon_k\}_{k=1}^K$ are K egomotion errors that are 'nearest' to $\boldsymbol{\phi}_{i,c}$, ϵ_{avg} is an average error, $\bar{\mathbf{R}}$ is a baseline *nominal* covariance, and $\gamma > 1$ is a hyper-parameter designed to exaggerate the effect of small changes in position error.

A.3 Training

Training proceeds by traversing the training path, selecting a subset of visual features at each step, and using them to compute an incremental position estimate. By comparing the estimated position to the ground truth position, we compute the translational Root Mean Squared Error (RMSE), ϵ, and store it at each feature's position in the prediction space. The full algorithm is summarized in Algorithm 6.

A.4 Testing

To use the PROBE model in a test environment, we compute the location of each observed visual feature in our prediction space, and then compute its relative weight β_i as a function of its K nearest neighbours in the training set. For efficiency, the K nearest neighbours are found using a k-d tree. The final scaling factor β_i is a function of the mean of the α values corresponding to the K nearest neighbours, normalized by ϵ_{avg}, the mean α value of the entire training set.

The value of K can be determined through cross-validation, and in practice depends on the size of the training set and the environment. The computation of β_i is designed to map small differences

[1]We conjecture that this is reasonable in a VO setup, where images change minimally between consecutive frames.

Algorithm 6 Train PROBE based on a dataset ($\mathcal{D}$) of pairs of input sensor data ($\mathcal{I}_s$) and ground truth egomotion ($\mathbf{T}_s$).

function BUILDPROBEMODEL($\mathcal{D}$)
  for $l \leftarrow [1, ..., N_{iter}]$ **do**
    for all $\mathcal{I}_s$, $\mathbf{T}_s$ in $\mathcal{D}$ **do**
      $\mathcal{F} \leftarrow$ EXTRACTFEATURES($\mathcal{I}_s$)
      $\{f_1, \ldots, f_J\} \leftarrow$ SAMPLE($\mathcal{F}$)
      $\hat{\mathbf{T}} \leftarrow$ COMPUTETRANSFORM($\{f_1, \ldots, f_J\}$)
      $\epsilon \leftarrow$ ERROR($\hat{\mathbf{T}}, \mathbf{T}_s$)
      $\{\boldsymbol{\phi}_{s,1}, \ldots, \boldsymbol{\phi}_{s,J}\} \leftarrow$ PREDICTOR($\{f_1, \ldots, f_J\}$)
      Insert $\{\boldsymbol{\phi}_{s,1}, \ldots, \boldsymbol{\phi}_{s,J}\}$ into $\mathcal{M}$ and store ϵ at all J locations
    end for
  end for
  return $\mathcal{M}$
end function

Algorithm 7 Compute scalar covariance factors, β_i, for a set of stereo feature tracks (and IMU data), $\mathcal{F}$, given a PROBE model $\mathcal{M}$.

function USEPROBE($\mathcal{M}$, $\mathcal{F}$, γ)
  $\epsilon_{\text{avg}} \leftarrow$ AVERAGEERROR($\mathcal{M}$)
  for all f_i in $\mathcal{F}$ **do**
    $\boldsymbol{\phi}_i \leftarrow$ PREDICTOR(f_i)
    $\epsilon_1, ..., \epsilon_K \leftarrow$ FINDKNN($\boldsymbol{\phi}_i, K, \mathcal{M}$)
    $\beta_i \leftarrow \left(\frac{1}{\epsilon_{\text{avg}} K} \sum_{k=1}^{K} \epsilon_k \right)^{\gamma}$
  end for
  return $\boldsymbol{\beta} = \{\beta_i\}$
end function

Figure A.1: Three types of environments in the KITTI dataset, as well as 2 types of environments at the University of Toronto. We use one trial from each category to train and then evaluate separate trials in the same category.

Figure A.2: Our four-wheeled skid-steered Clearpath Husky rover equipped with Skybotix VI-Sensor and Ashtech DGPS antenna used to collect the outdoor UTIAS dataset.

in learned α values to scalar weights that span several orders of magnitude. An appropriate value of γ can be found by searching through a set range of candidate values and choosing the value that minimizes the average RMSE (ARMSE) on the training set.

A.5 Experiments

We trained and evaluated this version of PROBE in two sets of experiments. The first set of experiments made use of 4.5 km of data from the City, Residential, and Road categories of the KITTI dataset (Geiger et al., 2013). In the second set of experiments, we collected indoor and outdoor datasets at the University of Toronto Institute for Aerospace Studies (UTIAS) using a Skybotix VI-Sensor mounted on an Adept MobileRobots Pioneer 3-AT rover and a Clearpath Husky rover, respectively (Figure A.2). In both cases, the camera recorded stereo images at 10 Hz while the IMU operated at 200 Hz. The outdoor dataset consisted of a 264 m training run followed by a 302 m evaluation run, with ground truth provided by RTK-corrected GPS. The indoor dataset consisted of a 32 m training run and a 33 m evaluation run through a room with varying lighting and shadows. For the indoor dataset, no ground truth was available, so we trained PROBE using only the knowledge that the training path should form a closed loop.

We compare PROBE to what we call a *nominal* visual-inertial navigation pipeline (or VINS, a

Table A.1: Comparison of translational Average Root Mean Square Error (ARMSE) and Final Translational Error on the KITTI dataset.

			Nominal RANSAC (99% outlier rejection)		Aggressive RANSAC (99.99% outlier rejection)		PROBE	
Trial	Type	Path Length	ARMSE	Final Error	ARMSE	Final Error	ARMSE	Final Error
26_drive_0051	City [1]	251.1 m	4.84 m	12.6 m	3.30 m	8.62 m	3.48 m	8.07 m
26_drive_0104	City [1]	245.1 m	0.977 m	4.43 m	0.850 m	3.46 m	1.19 m	3.61 m
29_drive_0071	City [1]	234.0 m	5.44 m	30.3 m	5.44 m	30.4 m	3.03 m	12.8 m
26_drive_0117	City [1]	322.5 m	2.29 m	9.07 m	2.29 m	9.07 m	2.76 m	9.08 m
30_drive_0027	Residential [1, †]	667.8 m	4.22 m	12.2 m	4.30 m	10.6 m	3.64 m	4.57 m
26_drive_0022	Residential [2]	515.3 m	2.21 m	3.99 m	2.66 m	6.09 m	3.06 m	4.99 m
26_drive_0023	Residential [2]	410.8 m	1.64 m	8.20 m	1.77 m	8.27 m	1.71 m	8.13 m
26_drive_0027	Road [3]	339.9 m	1.63 m	8.75 m	1.63 m	8.65 m	1.40 m	7.57 m
26_drive_0028	Road [3]	777.5 m	4.31 m	16.9 m	3.72 m	13.1 m	3.92 m	13.2 m
30_drive_0016	Road [3]	405.0 m	4.56 m	19.5 m	3.33 m	14.6 m	2.76 m	13.9 m
UTIAS Outdoor	Snowy parking lot	302.0 m	7.24 m	10.1 m	7.02 m	10.6 m	6.85 m	6.09 m
UTIAS Indoor	Lab interior	32.83 m	—	0.854 m	—	0.738 m	—	0.617 m

[1] Trained using sequence 09_26_drive_0005. [2] Trained using sequence 09_26_drive_0046. [3] Trained using sequence 09_26_drive_0015.
[†] This residential trial was evaluated with a model trained on a sequence from the city category because of several moving vehicles that were better represented in that training dataset.

pipeline based on the point-cloud-error-based VO described in Chapter 3, but with an initial estimate of rotation given by integrating angular velocities from an IMU), as well as a VINS with an aggressive RANSAC routine. In the nominal pipeline, we use RANSAC with enough iterations to be 99% confident that we select only inliers when as many as 50% of the features are outliers. In the aggressive case, we increase the confidence level to 99.99%. When PROBE is used, we apply a pre-processing step that makes use of the rotational estimate from the IMU to reject any egregious feature matches by thresholding the cosine distance between pairs of matched feature vectors. We assume small translations between frames and typically set the threshold to reject feature vectors that are separated by more than five degrees.

For feature extraction, matching, and sparse optical flow calculations, we use the open source vision library `libviso2` (Geiger et al., 2011). For all prediction space calculations, we use features in the left image of the stereo pair.

To evaluate PROBE, we run a nominal VINS pipeline on a given test trial, tune the RANSAC threshold to achieve reasonable translation error ($< 5\%$ final drift), then repeat the trial with the aggressive RANSAC procedure. Finally, we run this procedure again, this time disabling RANSAC completely and applying our trained PROBE model (with pre-processing) to each observed feature. Table A.1 compares the performance of each trained PROBE model to that of the nominal VINS pipeline and aggressive-RANSAC VINS.

Appendix B

Visual Odometry Implementation Details

This appendix presents further implementation details for a standard VO pipeline. You can find a `python` implementation of this in the open-source framework `pyslam`[1] developed by myself and Lee Clement.

B.1 Overview

In brief, the VO pipeline consists of the following steps.

1. Define an initial camera pose, $\mathbf{T}_{cw}$, that relates the first camera frame, $\underrightarrow{\mathcal{F}}_{c_0}$, to a world frame $\underrightarrow{\mathcal{F}}_{w}$.

2. Given an ideal calibrated stereo camera, stereo match and track a set of N_t landmarks, $\{\mathbf{y}_{i,c_0}, \mathbf{y}_{i,c_1}\}_{i=1}^{N_t}$, between the initial stereo camera pose, $\underrightarrow{\mathcal{F}}_{c_0}$, and a subsequent frame $\underrightarrow{\mathcal{F}}_{c_1}$. This step is non-trivial in general, but for our work, we rely on the framework `viso2` to provide these putative correspondences.

3. Perform RANSAC to reject outlier tracks. Compute candidate transforms $\mathbf{T}_{c_1 c_0}$ using the method of Umeyama (1991) (further elaborated in Barfoot (2017)) to reject outlying tracks.

4. Select initial guess of the relative transform, $\mathbf{T}_{c_1 c_0}^{(0)} = \mathbf{T}_t^{(0)}$. This can either be set to $\mathbf{1}$ or given by the model with the highest amount of inliers from the previous step. Starting at this initial operating point, use iterated reweighted least squares (IRLS) to solve the M-estimation problem,

$$\mathbf{T}_{c_1 c_0}^{*} = \mathbf{T}_t^{*} = \underset{\mathbf{T} \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N} \rho\left(\sqrt{\mathbf{e}_i{}^T \, \boldsymbol{\Sigma}_i^{-1} \mathbf{e}_i}\right) = \underset{\mathbf{T} \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N} \rho(\epsilon_i), \tag{B.1}$$

[1] http://github.com/utiasSTARS/pyslam

where $\rho(\cdot)$ can be selected based on empirical performance, $\mathbf{e}_i$ is the reprojection error, $\mathbf{e}_i(\mathbf{T}_t) = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}_t\mathbf{f}^{-1}(\mathbf{y}_{i,c_0}))$, and $\mathbf{f}$ is the stereo camera model.

5. Update the current camera pose as:

$$\mathbf{T}_{cw} \leftarrow \mathbf{T}_t^*\mathbf{T}_{cw}. \tag{B.2}$$

B.2 Solution with Robust Loss

In order to solve Equation (B.1), we use the method of IRLS, and solve the transformed problem,

$$\mathbf{T}^* = \underset{\mathbf{T}\in\mathrm{SE}(3)}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{N} \mathbf{e}_i{}^T \mathbf{M}_i\mathbf{e}_i, \tag{B.3}$$

where $\mathbf{M}_i(\mathbf{T}) \triangleq \frac{1}{\epsilon_i}\frac{\partial\rho}{\partial\epsilon_i}\boldsymbol{\Sigma}_i^{-1}$. To minimize this, we linearize about an operating point $\mathbf{T}_t^{(n)}$ and solve for an optimal update $\delta\boldsymbol{\xi}^*$ by minimizing the linear least squares problem:

$$\delta\boldsymbol{\xi}^* = \underset{\delta\boldsymbol{\xi}\in\mathbb{R}^6}{\operatorname{argmin}} \mathcal{L}(\delta\boldsymbol{\xi}) = \frac{1}{2}\sum_{i=1}^{N_t} \left(\mathbf{e}_i - \mathbf{J}_i\delta\boldsymbol{\xi}\right)^T \mathbf{M}_i \left(\mathbf{e}_i - \mathbf{J}_i\delta\boldsymbol{\xi}\right) \tag{B.4}$$

where we note that all subscripted elements are functions of the operating point (i.e., $\mathbf{e}_i = \mathbf{e}_i(\mathbf{T}_t^{(n)})$, $\mathbf{J}_i = \mathbf{J}_i(\mathbf{T}_t^{(n)})$ and $\mathbf{M}_i = \mathbf{M}_i(\mathbf{T}_t^{(n)})$). This is solved by the normal equations,

$$\delta\boldsymbol{\xi}^* = \left(\sum_{i=1}^{N_t} \mathbf{J}_i^T\mathbf{M}_i\mathbf{J}_i\right)^{-1} \left(\sum_{i=1}^{N_t} \mathbf{J}_i^T\mathbf{M}_i\mathbf{e}_i\right). \tag{B.5}$$

Given $\delta\boldsymbol{\xi}^*$, we update the operating point as

$$\mathbf{T}_t^{(n+1)} = \mathrm{Exp}\left(\delta\boldsymbol{\xi}^*\right)\mathbf{T}_t^{(n)}, \tag{B.6}$$

and iterate until convergence (the exact convergence criterion is typically not an important factor in relative egomotion estimation).

B.3 Deriving the Necessary Jacobians

To evaluate Equation (B.5), we require the matrix $\mathbf{J}_i$. We can derive $\mathbf{J}_i$ for a reprojection error, $\mathbf{e}_i(\mathbf{T}_t) = \mathbf{y}_{i,c_1} - \mathbf{f}(\mathbf{T}_t\mathbf{f}^{-1}(\mathbf{y}_{i,c_0}))$, by and making the following first order approximations:

$$\mathbf{T}_t \approx (\mathbf{1} + \delta\boldsymbol{\xi}^\wedge)\mathbf{T}_t^{(n)} \tag{B.7}$$

$$\mathbf{f}(\boldsymbol{p}_i) \approx \mathbf{f}(\boldsymbol{p}_i^{(n)}) + \mathbf{J}_p(\boldsymbol{p}_i^{(n)})\delta\boldsymbol{p}, \tag{B.8}$$

where $\mathbf{J}_p(\boldsymbol{p}_i^{(n)}) = \left.\frac{\partial \mathbf{f}}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}_i^{(n)}}$ and $\boldsymbol{p}_i^{(n)} = \mathbf{T}_t^{(n)}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0})$. Using these definitions we can relate small perturbations of $\boldsymbol{p}_i^{(n)}$ to perturbations in our state $\mathbf{T}_t^{(n)}$,

$$\begin{aligned}
\boldsymbol{p}_i^{(n)} + \delta\boldsymbol{p} &\approx (\mathbf{1} + \delta\boldsymbol{\xi}^\wedge)\mathbf{T}_t^{(n)}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0}) && \text{(B.9)}\\
&= \underbrace{\mathbf{T}_t^{(n)}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0})}_{\boldsymbol{p}_i^{(n)}} + \delta\boldsymbol{\xi}^\wedge \underbrace{\mathbf{T}_t^{(n)}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0})}_{\boldsymbol{p}_i^{(n)}} && \text{(B.10)}\\
&= \boldsymbol{p}_i^{(n)} + \delta\boldsymbol{\xi}^\wedge\boldsymbol{p}_i^{(n)} && \text{(B.11)}\\
&= \boldsymbol{p}_i^{(n)} + (\boldsymbol{p}_i^{(n)})^\odot\delta\boldsymbol{\xi} && \text{(B.12)}
\end{aligned}$$

which allows us to write $\delta\boldsymbol{p} = (\boldsymbol{p}_i^{(n)})^\odot\delta\boldsymbol{\xi}$. Here we have used the notation from Barfoot (2017), where the $(\cdot)^\odot$ operator is defined on homogeneous points as follows:

$$\boldsymbol{p}^\odot = \begin{bmatrix}\mathbf{p}\\ 1\end{bmatrix}^\odot = \begin{bmatrix}\mathbf{1} & -\mathbf{p}^\wedge\\ \mathbf{0}^T & \mathbf{0}^T\end{bmatrix}. \tag{B.13}$$

This leads to the convenient expression, $\boldsymbol{\xi}^\wedge\boldsymbol{p} = \boldsymbol{p}^\odot\boldsymbol{\xi}$, that we use above. Combining everything, we have:

$$\begin{aligned}
\mathbf{e}_i(\delta\boldsymbol{\xi}) &\approx \underbrace{\mathbf{y}_{i,c_1} - \mathbf{f}(\boldsymbol{p}_i^{(n)})}_{\mathbf{e}_i(\mathbf{T}_t^{(n)})} - \mathbf{J}_p(\boldsymbol{p}_i^{(n)})\delta\boldsymbol{p} && \text{(B.14)}\\
&= \mathbf{e}_i(\mathbf{T}_t^{(n)}) - \underbrace{\mathbf{J}_p(\boldsymbol{p}_i^{(n)})(\boldsymbol{p}_i^{(n)})^\odot}_{\mathbf{J}_i}\delta\boldsymbol{\xi}, && \text{(B.15)}
\end{aligned}$$

which concludes our derivation. In summary, to evaluate Equation (B.5), we use the expression,

$$\mathbf{J}_i = \mathbf{J}_p\left(\mathbf{T}_t^{(n)}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0})\right)\left(\mathbf{T}_t^{(n)}\mathbf{f}^{-1}(\mathbf{y}_{i,c_0})\right)^\odot. \tag{B.16}$$

Finally, for stereo camera model defined with the origin in the left camera frame,

$$\mathbf{y}_{i,c} = \begin{bmatrix}u_l\\ v_l\\ d\end{bmatrix} = \mathbf{f}(\boldsymbol{p}_{i,c}) = \mathbf{f}(\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}) == \mathbf{M}\frac{1}{z}\boldsymbol{p}_{i,c}, \tag{B.17}$$

where

$$\mathbf{M} = \begin{bmatrix}f & 0 & c_u & 0\\ 0 & f & c_v & 0\\ 0 & 0 & 0 & fb\end{bmatrix}, \tag{B.18}$$

the Jacobian, $\mathbf{J}_p = \dfrac{\partial \mathbf{f}}{\partial \boldsymbol{p}}$ is given by

$$\mathbf{J}_p = \frac{\partial \mathbf{f}}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{f}{z} & 0 & -f\frac{x}{z^2} & 0 \\ 0 & \frac{f}{z} & -f\frac{y}{z^2} & 0 \\ 0 & 0 & -f\frac{b}{z^2} & 0 \end{bmatrix}. \tag{B.19}$$

Bibliography

Agarwal, S., Mierle, K., et al. (2016). Ceres solver.

Albertz, J. (2007). A look back. *Photogrammetric Engineering & Remote Sensing*, 73(5):504–506.

Alcantarilla, P. F. and Woodford, O. J. (2016). Noise models in feature-based stereo visual odometry.

Altmann, S. L. (1989). Hamilton, rodrigues, and the quaternion scandal. *Math. Mag.*, 62(5):291–308.

Amos, B. and Kolter, J. Z. (2017). OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 136–145. PMLR.

Barfoot, T. D. (2017). *State Estimation for Robotics*. Cambridge University Press.

Barfoot, T. D. and Furgale, P. T. (2014). Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Trans. Rob.*, 30(3):679–693.

Brachmann, E. and Rother, C. (2018). Learning less is more-6d camera localization via 3d surface regression. In *Proc. CVPR*, volume 8.

Brachmann, E. and Rother, C. (2019). Neural- Guided RANSAC: Learning where to sample model hypotheses. In *ICCV*.

Bruss, A. R. and Horn, B. K. (1983). Passive Navigation. *Computer Vision, Graphics, and Image Processing*, 21(1):3–20.

Byravan, A. and Fox, D. (2017). SE3-nets: Learning rigid body motion using deep neural networks. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 173–180.

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 Trans. Rob.*, 32(6):1309–1332.

Carlone, L., Rosen, D. M., Calafiore, G., Leonard, J. J., and Dellaert, F. (2015a). Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 125–132.

Carlone, L., Tron, R., Daniilidis, K., and Dellaert, F. (2015b). Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4597–4604.

Censi, A. (2007). An accurate closed-form estimate of icp's covariance. In *Proceedings 2007 IEEE international conference on robotics and automation*, pages 3167–3172. IEEE.

Cheng, Y., Maimone, M. W., and Matthies, L. (2006). Visual odometry on the mars exploration rovers - a tool to ensure accurate driving and science imaging. *IEEE Robot. Automat. Mag.*, 13(2):54–62.

Clark, R., Wang, S., Wen, H., Markham, A., and Trigoni, N. (2017). Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem.

Clement, L., Peretroukhin, V., and Kelly, J. (2017). Improving the accuracy of stereo visual odometry using visual illumination estimation. In Kulic, D., Nakamura, Y., Khatib, O., and Venture, G., editors, *2016 International Symposium on Experimental Robotics*, volume 1 of *Springer Proceedings in Advanced Robotics*, pages 409–419. Springer International Publishing, Berlin Heidelberg. Invited to Journal Special Issue.

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.

Crete, F., Dolmiere, T., Ladret, P., and Nicolas, M. (2007). The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *Human vision and electronic imaging XII*, volume 6492, page 64920I. International Society for Optics and Photonics.

Cvišić, I. and Petrović, I. (2015). Stereo odometry based on careful feature selection and tracking. In *Proc. European Conf. on Mobile Robots (ECMR)*, pages 1–6.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Comput. Vision and Pattern Recognition, (CVPR)*, pages 248–255.

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2016). Deep image homography estimation.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *Proc. Int. Conf. on Machine Learning*, ICML'16, pages 1329–1338.

Eisenman, A. R., Liebe, C. C., and Perez, R. (2002). Sun sensing on the mars exploration rovers. In *Aerosp. Conf. Proc.*, volume 5, pages 5–2249–5–2262 vol.5. IEEE.

Engel, J., Koltun, V., and Cremers, D. (2018). Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer.

Fischler, M. and Bolles, R. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395.

Fitzgibbon, A. W., Robertson, D. P., Criminisi, A., Ramalingam, S., and Blake, A. (2007). Learning priors for calibrating families of stereo cameras. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, pages 1–8.

Florez, S. A. R. (2010). *Contributions by vision systems to multi-sensor object localization and tracking for intelligent vehicles*. PhD thesis.

Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2015). IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation.

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). SVO: Fast semi-direct monocular visual odometry. In *Proc. IEEE Int. Conf. Robot. Automat.(ICRA)*, pages 15–22. IEEE.

Furgale, P. and Barfoot, T. D. (2010). Visual teach and repeat for long-range rover autonomy. *J. Field Robot.*, 27(5):534–560.

Furgale, P., Carle, P., Enright, J., and Barfoot, T. D. (2012). The devon island rover navigation dataset. *Int. J. Rob. Res.*, 31(6):707–713.

Furgale, P., Enright, J., and Barfoot, T. (2011). Sun sensor navigation for planetary rovers: Theory and field testing. *IEEE Trans. Aerosp. Electron. Syst.*, 47(3):1631–1647.

Furgale, P., Rehder, J., and Siegwart, R. (2013). Unified temporal and spatial calibration for multi-sensor systems. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1280–1286.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge.

Gal, Y. and Ghahramani, Z. (2016a). Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *Proc. Int. Conf. Learning Representations (ICLR), Workshop Track*.

Gal, Y. and Ghahramani, Z. (2016b). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proc. Int. Conf. Mach. Learning (ICML)*, pages 1050–1059.

Gallego, G. and Yezzi, A. (2015). A compact formula for the derivative of a 3-D rotation in exponential coordinates. *J. Math. Imaging Vis.*, 51(3):378–384.

Garg, R., Carneiro, G., and Reid, I. (2016). Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *European Conf. on Comp. Vision*, pages 740–756. Springer.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *Int. J. Rob. Res.*, 32(11):1231–1237.

Geiger, A., Ziegler, J., and Stiller, C. (2011). StereoScan: Dense 3D reconstruction in real-time. In *Proc. IEEE Intelligent Vehicles Symp. (IV)*, pages 963–968.

Geman, S., McClure, D. E., and Geman, D. (1992). A nonlinear filter for film restoration and other problems in image processing. *CVGIP: Graphical models and image processing*, 54(4):281–289.

Glocker, B., Izadi, S., Shotton, J., and Criminisi, A. (2013). Real-time rgb-d camera relocalization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 173–179.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Grewal, M. S. and Andrews, A. P. (2010). Applications of kalman filtering in aerospace 1960 to the present [historical perspectives]. *IEEE Control Syst. Mag.*, 30(3):69–78.

Gustafsson, F. and Gustafsson, F. (2000). *Adaptive filtering and change detection*, volume 1. Citeseer.

Haarnoja, T., Ajay, A., Levine, S., and Abbeel, P. (2016). Backprop KF: Learning discriminative deterministic state estimators. In *Proc. Advances in Neural Inform. Process. Syst. (NIPS)*.

Handa, A., Bloesch, M., Pătrăucean, V., Stent, S., McCormac, J., and Davison, A. (2016). gvnn: Neural network library for geometric computer vision. In *Computer Vision – ECCV 2016 Workshops*, pages 67–82. Springer, Cham.

Hartley, R., Trumpf, J., Dai, Y., and Li, H. (2013). Rotation averaging. *Int. J. Comput. Vis.*, 103(3):267–305.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Holland, P. W. and Welsch, R. E. (1977). Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9):813–827.

Hu, H. and Kantor, G. (2015). Parametric covariance prediction for heteroscedastic noise. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, pages 3052–3057.

Huber, P. J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, pages 73–101.

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 *Proc. ACM Int. Conf. Multimedia (MM)*, pages 675–678.

Julier, S. J. and Uhlmann, J. K. (2007). Using covariance intersection for slam. *Robotics and Autonomous Systems*, 55(1):3–20.

Kelly, J., Saripalli, S., and Sukhatme, G. S. (2008). Combined visual and inertial navigation for an unmanned aerial vehicle. In *Proc. Field and Service Robot. (FSR)*, pages 255–264.

Kendall, A. and Cipolla, R. (2016). Modelling uncertainty in deep learning for camera relocalization. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 4762–4769.

Kendall, A. and Cipolla, R. (2017). Geometric loss functions for camera pose regression with deep learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6555–6564.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision?

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.

Kerl, C., Sturm, J., and Cremers, D. (2013). Robust odometry estimation for RGB-D cameras. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 3748–3754.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*. arXiv: 1412.6980.

Ko, J. and Fox, D. (2009). Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc.

Lalonde, J.-F., Efros, A. A., and Narasimhan, S. G. (2011). Estimating the natural illumination conditions from a single outdoor image. *Int. J. Comput. Vis.*, 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. *J. Field Robot.*, 29(3):426–444.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551.

Lee, S., Purushwalkam, S., Cogswell, M., Crandall, D., and Batra, D. (2015). Why M heads are better than one: Training a diverse ensemble of deep networks.

Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual–inertial odometry using nonlinear optimization. *Int. J. Rob. Res.*, 34(3):314–334.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. (2016). End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*

Li, Q., Qian, J., Zhu, Z., Bao, X., Helwa, M. K., and Schoellig, A. P. (2017a). Deep neural networks for improved, impromptu trajectory tracking of quadrotors. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 5183–5189.

Li, R., Wang, S., Long, Z., and Gu, D. (2017b). UnDeepVO: Monocular visual odometry through unsupervised deep learning.

Liu, K., Ok, K., Vega-Brown, W., and Roy, N. (2018). Deep inference for covariance estimation: Learning gaussian noise models for state estimation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1436–1443. IEEE.

Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ma, W.-C., Wang, S., Brubaker, M. A., Fidler, S., and Urtasun, R. (2016). Find your way by observing the sun and other semantic cues.

MacTavish, K. and Barfoot, T. D. (2015). At all costs: A comparison of robust cost functions for camera correspondence outliers. In *Proc. Conf. on Comp. and Robot Vision (CRV)*, pages 62–69.

Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2016). 1 year, 1000 km: The oxford RobotCar dataset. *Int. J. Rob. Res.*

Maimone, M., Cheng, Y., and Matthies, L. (2007). Two years of visual odometry on the mars exploration rovers. *J. Field Robot.*, 24(3):169–186.

Mayor, A. (2019). *Gods and Robots.* Princeton University Press.

McManus, C., Upcroft, B., and Newman, P. (2014). Scene signatures: Localised and point-less features for localisation. In *Proc. Robotics: Science and Systems X.*

Melekhov, I., Ylioinas, J., Kannala, J., and Rahtu, E. (2017). Relative camera pose estimation using convolutional neural networks. In *Proc. Int. Conf. on Advanced Concepts for Intel. Vision Syst.*, pages 675–687. Springer.

Melkumyan, A. and Ramos, F. (2011). Multi-kernel gaussian processes. In *Twenty-Second International Joint Conference on Artificial Intelligence.*

Oliveira, G. L., Radwan, N., Burgard, W., and Brox, T. (2017). Topometric localization with deep learning. *arXiv preprint arXiv:1706.08775.*

Olson, C. F., Matthies, L. H., Schoppers, M., and Maimone, M. W. (2003). Rover navigation using stereo ego-motion. *Robot. Auton. Syst.*, 43(4):215–229.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep exploration via bootstrapped DQN. In *Proc. Advances in Neural Inform. Process. Syst. (NIPS)*, pages 4026–4034.

Peretroukhin, V., Clement, L., Giamou, M., and Kelly, J. (2015a). PROBE: Predictive robust estimation for visual-inertial navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'15)*, pages 3668–3675, Hamburg, Germany.

Peretroukhin, V., Clement, L., and Kelly, J. (2015b). Get to the point: Active covariance scaling for feature tracking through motion blur. In *Proceedings of the IEEE International Conference on Robotics and Automation Workshop on Scaling Up Active Perception*, Seattle, Washington, USA.

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 IEEE International Conference on Robotics and Automation (ICRA'17)*, 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., Kelly, J., and Barfoot, T. D. (2014). Optimizing camera perspective for stereo visual odometry. In *Canadian Conference on Comp. and Robot Vision*, pages 1–7.

Peretroukhin, V., Vega-Brown, W., Roy, N., and Kelly, J. (2016). PROBE-GK: Predictive robust estimation using generalized kernels. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 817–824.

Peretroukhin, V., Wagstaff, B., and Kelly, J. (2019). Deep probabilistic regression of elements of SO(3) using quaternion averaging and uncertainty injection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19) Workshop on Uncertainty and Robustness in Deep Visual Learning*, pages 83–86, Long Beach, California, USA.

Punjani, A. and Abbeel, P. (2015). Deep learning helicopter dynamics models. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 3223–3230.

Ranftl, R. and Koltun, V. (2018). Deep fundamental matrix estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 284–299.

Richter, C. and Roy, N. (2017). Safe visual navigation via deep learning and novelty detection.

Rosen, D. M., Carlone, L., Bandeira, A. S., and Leonard, J. J. (2019). SE-Sync: A certifiably correct algorithm for synchronization over the special euclidean group. *Int. J. Rob. Res.*, 38(2-3):95–125.

Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE Robot. Autom. Mag.*, 18(4):80–92.

Schonberger, J. L., Hardmeier, H., Sattler, T., and Pollefeys, M. (2017). Comparative Evaluation of Hand-Crafted and Learned Local Features. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6959–6968, Honolulu, HI. IEEE.

Sibley, G., Matthies, L., and Sukhatme, G. (2007). Bias reduction and filter convergence for long range stereo. In *Robotics Research*, pages 285–294. Springer Berlin Heidelberg.

Sola, J. (2017). Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*.

Solà, J., Deray, J., and Atchuthan, D. (2018). A micro lie theory for state estimation in robotics.

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*, 15:1929–1958.

Sünderhauf, N., Shirazi, S., Dayoub, F., Upcroft, B., and Milford, M. (2015). On the performance of ConvNet features for place recognition. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, pages 4297–4304.

Sunderhauf, N., Shirazi, S., Jacobson, A., Dayoub, F., Pepperell, E., Upcroft, B., and Milford, M. (2015). Place recognition with ConvNet landmarks: Viewpoint-robust, condition-robust, training-free. In *Proc. Robotics: Science and Systems XII*.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proc. IEEE Conf. Comput. Vision and Pattern Recognition, (CVPR)*, pages 1–9.

Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (2000). Bundle Adjustment — A Modern Synthesis. In Goos, G., Hartmanis, J., van Leeuwen, J., Triggs, B., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, volume 1883, pages 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg.

Tsotsos, K., Chiuso, A., and Soatto, S. (2015). Robust inference for visual-inertial sensor fusion. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 5203–5210.

Umeyama, S. (1991). Least-Squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(4):376–380.

Vega-Brown, W., Bachrach, A., Bry, A., Kelly, J., and Roy, N. (2013). CELLO: A fast algorithm for covariance estimation. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 3160–3167.

Vega-Brown, W. and Roy, N. (2013). CELLO-EM: Adaptive sensor models without ground truth. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pages 1907–1914.

Vega-Brown, W. R., Doniec, M., and Roy, N. G. (2014). Nonparametric Bayesian inference on multivariate exponential families. In *Proc. Advances in Neural Information Proc. Syst. (NIPS) 27*, pages 2546–2554.

Wang, R., Schworer, M., and Cremers, D. (2017a). Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3923–3931, Venice. IEEE.

Wang, S., Clark, R., Wen, H., and Trigoni, N. (2017b). DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2043–2050.

Warren, R. (1976). The perception of egomotion. *Journal of Experimental Psychology: Human Perception and Performance*, 2(3):448.

Wilson, A. G. and Ghahramani, Z. (2011). Generalised wishart processes. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 736–744. AUAI Press.

Yang, F., Choi, W., and Lin, Y. (2016). Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Proc. IEEE Int. Conf. Comp. Vision and Pattern Recognition (CVPR)*, pages 2129–2137.

Yang, N., Wang, R., Stueckler, J., and Cremers, D. (2018). Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *European Conference on Computer Vision (ECCV)*. accepted as oral presentation, arXiv 1807.02570.

Zhang, G. and Vela, P. (2015). Optimally observable and minimal cardinality monocular SLAM. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 5211–5218.

Zhou, B., Krähenbühl, P., and Koltun, V. (2019). Does computer vision matter for action? *Science Robotics*, 4(30).

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in Neural Inform. Process. Syst. (NIPS)*, pages 487–495.

Zhou, T., Brown, M., Snavely, N., and Lowe, D. G. (2017). Unsupervised learning of depth and Ego-Motion from video. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6612–6619.