On learning probabilistic models for state 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 probabilistic models for state estimation

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

The paradigm of learning complex measurement models from training data, rather than crafting these models by hand, has become pervasive in computer vision and robotics. These learned models, however, are often presented in isolation, and it can be unclear to what extent they can be integrated into (or replace) existing high-fidelity geometric and kinematic models within state estimation pipelines. In this work, we explore ways in which these two modelling paradigms can be combined while still preserving the core advantages of both. To this end, we first present two Bayesian machine-learning-based techniques that improve on existing measurement models in visual odometry: (1) a fast, kernel-based Bayesian technique that learns a model for measurement covariances of a stereo camera to produce non-stationary robust costs and (2) a deep Bayesian Convolutional Neural Network model that can infer the direction of the Sun from a single RGB image.

Second, we extend these ideas to incorporate ideas from the field of deep learning. Instead of attempting to replacing existing pipelines with monolithic trained networks, we show that we can use the representational power of deep networks to build corrective, probabilistic models that correct, instead of replace, existing localization pipelines. Notably, we leverage concepts from differential geometry and matrix Lie groups to construct fully differentiable analytic loss functions that permit principled learning of corrections and associated uncertainty.

We validate our methods on several kilometres of sensor data collected in sundry settings such as urban roads, indoor labs, and planetary analogue sites in the high Arctic.

To all those who encouraged my intellectual wanderlust.

Acknowledgements

I would like to acknowledge my parents, my partner-in-crime, and my partner-in-academia.

Contents

Notation

a	:	Symbols in this font are real scalars.
$\mathbf{a}$	:	Symbols in this font are real column vectors.
$\mathbf{A}$	:	Symbols in this font are real matrices.
$\sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{R})$	:	Normally distributed with mean $\boldsymbol{\mu}$ and covariance $\mathbf{R}$.
$E[\cdot]$	:	The expectation operator.
$\underrightarrow{\mathcal{F}}_a$	:	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.
$\mathbf{p}_a^{c,b}$	:	A vector from point b to point c (denoted by the superscript) and expressed in $\underrightarrow{\mathcal{F}}_a$ (denoted by the subscript).
$\mathbf{C}_{a,b}$	:	The 3×3 rotation matrix that transforms vectors from $\underrightarrow{\mathcal{F}}_b$ to $\underrightarrow{\mathcal{F}}_a$: $\mathbf{p}_a^{c,b} = \mathbf{C}_{a,b}\mathbf{p}_b^{c,b}$.
$\mathbf{T}_{a,b}$	:	The 4×4 transformation matrix that transforms homogeneous points from $\underrightarrow{\mathcal{F}}_b$ to $\underrightarrow{\mathcal{F}}_a$: $\mathbf{p}_a^{c,a} = \mathbf{T}_{a,b}\mathbf{p}_b^{c,b}$.

Chapter 1

Introduction

Reliable state estimation is at the heart of mobile robotics.

1.1 Measurement Models

A core component of any state estimation pipeline is the *measurement model*, a mathematical entity that relates observed measurement data to latent state parameters. By quantifying information contained within sensor data, probabilistic measurement models facilitate the construction of complex state estimation architectures that can fuse observations from sensors of varied modality to create rich models of the external world and infer the state of a mobile robot within it. My thesis focuses on egomotion estimation: the problem of accurately and consistently estimating the relative pose of a moving robot. For this task, a variety of different sensors may be useful (e.g., lidar, stereo cameras, or inertial measurement units), and each may allow for various hand-crafted or learned probabilistic measurement models.

For instance, modern visual and inertial egomotion estimation pipelines (Leutenegger et al., 2015; Cvišić and Petrović, 2015; Tsotsos et al., 2015a) have achieved impressive localization accuracy on trajectories spanning several kilometres by carefully extracting and tracking sparse visual features (using *hand-crafted* algorithms) across consecutive images. Simultaneously, significant effort has gone to developing localization pipelines that eschew sparse features in favour of *dense* visual data (Alcantarilla and Woodford, 2016; Forster et al., 2014a), typically relying on loss functions that use direct pixel intensities. Notably, in both the dense and sparse approaches, the visual observation model is often assumed to have an uncertainty model that is static, and known a priori.

In the last several years, a significant part of the state estimation literature has focused on learned visual measurement models through convolutional neural networks

(CNNs). Although initially developed for image classification (LeCun et al., 2015a), CNN-based measurement models have been applied to numerous problems in geometric state estimation (e.g., homography estimation (DeTone et al., 2016), single image depth reconstruction (Garg et al., 2016), camera re-localization (Kendall and Cipolla, 2016), place recognition (Sünderhauf et al., 2015)). A number of recent CNN-based approaches have also tackled the problem of egomotion estimation, often purporting to obviate the need for classical visual localization pipelines by learning pose changes *end-to-end*, directly from image data (e.g., Melekhov et al. (2017), Handa et al. (2016), Oliveira et al. (2017)).

Despite this surge of excitement, significant debate has emerged within the robotics and computer vision communities regarding the extent to which deep models should replace existing geometric state estimation algorithms. Owing to their representational power, deep models may move the onerous task of selecting 'good' (i.e., robust to environmental vagaries and sensor motion) visual features from the roboticist to the learned model. By design, deep models also provide a straight-forward formulation for using *dense* data while being flexible in their loss function, and taking full advantage of modern computing architecture to minimize run time. Despite these potential benefits, current deep regression techniques for state estimation often generalize poorly to new environments, come with few analytical guarantees, and provide only point estimates of latent parameters.

1.2 Outstanding Issues in the Field

1.2.1 The limits of homoscedastic noise models

Although several state-of-the-art state estimation pipelines (Leutenegger et al., 2015; Cvišić and Petrović, 2015) leave observation uncertainty associated with sensor measurements as a static tuning parameter, recent work (Vega-Brown et al., 2014a; Hu and Kantor, 2015) suggests that using a 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. In foot-mounted navigation, the inferred zero velocity detector may be more or less informative depending on the exact type of motion and individual gait. In visual data, inferred visual observations can be degraded not only due to sensor imperfections (e.g. poor intrinsic calibration, digitization effects, motion blur), but also as a result of the observed environment (e.g. self-similar scenes, specular surfaces, textureless environments). Indeed, robust costs Alcantarilla

and Woodford (2016); MacTavish and Barfoot (2015); Agarwal et al. (2013a) and whiteness tests (Tsotsos et al., 2015a) have commonly been used to alleviate the problem of poor noise modelling, but more work is required to better learn uncertainty in complex measurement models.

1.2.2 Deep, learned models with no uncertainty estimates

Although the paradigm of deep neural networks has resulted in several significant achievements in the fields of computer vision (LeCun et al., 2015a), these types of models have largely focused on point estimates (in either regression or classification) without any principled uncertainty estimates. Recently, the regularization techniques of dropout and dropconnect in Convolutional Neural Networks have been linked with approximate variational inference in homoscedastic Gaussian Processes (Gal and Ghahramani, 2016; Kendall and Cipolla, 2016; McClure and Kriegeskorte, 2016), and the statistical technique of *bootstrapping* has been applied to Deep Q Networks (Osband et al., 2016) to infer uncertainty, but both techniques are in their infancy. Recent work (Osband, 2017) has also suggested that the former technique of dropout-based 'uncertainty' is actually a measure of *risk* (i.e., stochasticity in the measurements) and not *uncertainty* over state parameters. Further, the same work showed that even this risk quantification can be arbitrarily bad given a fixed dropout parameter (which is typically the case).

1.2.3 Integration of deep models into state estimation pipelines

To integrate the power of deep networks into state estimation algorithms, the recent literature differs in how to proceed. While some attempt to parametrize geometric transformations in their unconstrained state, and then learn the resulting state within a deep network regression optimization (Costante et al., 2016; Kendall et al., 2015), others integrate deep networks within outer estimation loops (Haarnoja et al., 2016). Yet other work has used the neural network as an error correcting mechanism on top on an existing kinematic or dynamic model (Punjani and Abbeel, 2015). This integration is made more difficult by the lack of uncertainty estimates associated with many learned measurement models in the computer vision and machine learning literature.

1.3 Original Contributions

The original contributions of my dissertation include three architectures that incorporate novel ways of using learned measurement models to improve egomotion estimation:

PROBE, Sun-BCNN, and DPC-Net. Since the last D.E.C. meeting, I focussed on the latter two, and hope to build on DPC-Net in the final part of my thesis.

1.3.1 Predictive Robust Estimation for Sparse Visual Odometry

Predictive Robust Estimation (PROBE) is a technique that uses k-NN regression (original PROBE) or Generalized Kernels (Vega-Brown et al., 2014a) (PROBE-GK) to train a predictive model for heteroscedastic measurement covariance to improve estimator accuracy and consistency. PROBE (Peretroukhin et al., 2015a) and PROBE-GK (Peretroukhin et al., 2016a) were published at IROS 2015 and ICRA 2016, respectively. For more information about PROBE, please refer to my past D.E.C. reports.

1.3.2 Software Sun Sensor using a Bayesian Convolutional Neural Network

Sun-BCNN is a technique to infer a probabilistic estimate of the direction of the sun from a single RGB image using a Bayesian Convolutional Neural Networks (BCNN). The method works much like dedicated sun sensors (Lambert et al., 2012), but requires no additional hardware, and can provide mean and covariance estimates that can be readily incorporated into existing visual odometry frameworks. I worked on this project in collaboration with Lee Clement. While he focussed on integrating Sun-BCNN into the visual estimator, I developed the BCNN architecture and focused on uncertainty modelling. Initial exploratory work was published at ISER 2016, and the BCNN improvement was presented at ICRA 2017. An additional journal paper summarizing the work of the prior two papers, adding data from the Canadian High Arctic and Oxford, and investigating the effect of cloud cover and transfer learning was published in the International Journal of Robotics' Research, Special Issue on Experimental Robotics at the end of 2017.

1.3.3 Deep Pose Corrections (DPC-Net)

Deep Pose Correction (DPC, Figure 1.1) is a novel approach to improving egomotion estimates through pose corrections learned through deep regression. DPC takes as its starting point an efficient, classical localization algorithm that computes high-rate pose estimates. To it, it adds a Deep Pose Correction Network (DPC-Net) that learns low-rate, 'small' *corrections* from training data that are then fused with the original estimates.

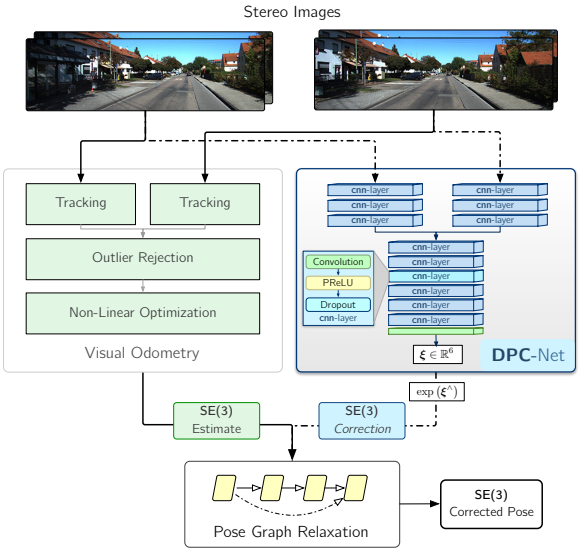


Figure 1.1: The DPC-Net system. Learned SE(3) pose corrections are fused with pose estimates from a classical estimator.

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. DPC-Net was accepted for publication in the proceedings of ICRA 2018, and as part of Robotics and Automation Letters (Peretroukhin and Kelly, 2018).

1.3.4 Probabilistic SE(3) inference using deep networks

In the final part of my dissertation, I am investigating ways in which a network like DPC-Net can produce consistent probabilistic pose (i.e., SE(3)) estimates. By inferring a probability density over the unconstrained Lie algebra coordinates, one can induce a probability density over the group. There are several ways to proceed with the former induction. Although I have already published work on Bayesian CNNs, as mentioned in the introduction, the validity of their uncertainty estimates has come into question. One potential alternative is to merge Gaussian Processes and deep networks through deep kernel learning (Wilson et al., 2016). Although theoretically promising, this technique still scales poorly with training data size.

Instead, I am investigating non-Bayesian approaches that learn a bespoke covariance matrix as part of their output using a log likelihood loss. By parametrizing the precision matrix through a Cholesky decomposition, it is possible to output positive definite matrices without enforcing any constraints on the network output (this approach was presented in the context of deep networks (Haarnoja et al., 2016), and in the context of

heteroscedastic noise modelling in (Hu and Kantor, 2015); it is similar in principal to switchable constraints (Agarwal et al., 2013a)).

Another potential alternative method is bootstrap aggregation (bagging). Bagging can be used to compute confidence bounds by training several models on randomly subsampled data (e.g., uncertainty for deep Q networks (Osband et al., 2016)). To reduce the cost of training multiple models, a number of authors have also suggested training a single network with multiple heads (Lee et al., 2015), (Lakshminarayanan et al., 2017). I am investigating this latter approach, and also combining it with the log likelihood covariance learning.

Chapter 2

Preliminaries

2.1 Stereo Visual Odometry

In our frame-to-frame sparse stereo odometry pipeline, the objective is to find $\mathbf{T}_t \in \mathrm{SE}(3)$, the rigid transform between two subsequent stereo camera poses (note that the temporal index t refers to the set of two stereo camera poses). We begin by rectifying, then stereo and temporally matching the set of 4 images to generate the corresponding locations of a set of N_t visual landmarks in each stereo pair. Each landmark corresponds to a point in space, expressed in homogeneous coordinates in the camera frame as $\boldsymbol{p}_{i,t} := \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \end{bmatrix}^T \in \mathbb{P}^3$. The stereo-camera model, f, projects a landmark expressed in homogeneous coordinates into image space, so that $\mathbf{y}_{i,t}$, the stereo pixel coordinates of landmark i in the first camera pose at time t, is given by

$$\mathbf{y}_{i,t} = \begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix} = f(\boldsymbol{p}_{i,t}) = \mathbf{M}\frac{1}{p_3}\boldsymbol{p}_{i,t}, \tag{2.1}$$

where

$$\mathbf{M} = \begin{bmatrix} f_u & 0 & c_u & f_u\frac{b}{2} \\ 0 & f_v & c_v & 0 \\ f_u & 0 & c_u & -f_u\frac{b}{2} \\ 0 & f_b & c_v & 0 \end{bmatrix}. \tag{2.2}$$

Here, $\{c_u, c_v\}$, $\{f_u, f_v\}$, and b are the principal points, focal lengths and baseline of the stereo camera respectively. Note that in this formulation, the stereo camera frame is centered between the two individual lenses.

We triangulate landmarks in the first camera frame, $\mathbf{y}_{i,t}$, and re-project them into the second frame, $\mathbf{y}'_{i,t}$. We model errors due to sensor noise and quantization as a Gaussian distribution in image space with a known covariance $\mathbf{R}$,

$$p(\mathbf{y}'_{i,t}|\mathbf{y}_{i,t}, \mathbf{T}_t, \mathbf{R}) = \mathcal{N}\left(\mathbf{e}_{i,t}(\mathbf{T}_t); \mathbf{0}, \mathbf{R}\right), \tag{2.3}$$

where

$$\mathbf{e}_{i,t} = \mathbf{y}'_{i,t} - f(\mathbf{T}_t f^{-1}(\mathbf{y}_{i,t})). \tag{2.4}$$

The maximum likelihood transform, $\mathbf{T}_t^*$, is then given by

$$\mathbf{T}_t^* = \underset{\mathbf{T}_t \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} {\mathbf{e}_{i,t}}^T \mathbf{R}^{-1} \mathbf{e}_{i,t}. \tag{2.5}$$

This is a nonlinear least squares problem, and can be solved iteratively using standard techniques. During iteration n, we represent the transform as the product of an estimate $\mathbf{T}^{(n)} \in \mathrm{SE}(3)$ and a perturbation $\delta\boldsymbol{\xi} \in \mathbb{R}^6$ represented in exponential coordinates:

$$\mathbf{T}_t = \exp\left(\delta\boldsymbol{\xi}^\wedge\right)\mathbf{T}_t^{(n)}. \tag{2.6}$$

The wedge operator $(\cdot)^\wedge$ is defined (following Barfoot and Furgale (2014)) as both the map $\mathbb{R}^3 \to \mathfrak{so}(3)$,

$$\boldsymbol{\phi}^\wedge \triangleq \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}, \tag{2.7}$$

and the map $\mathbb{R}^6 \to \mathfrak{se}(3)$,

$$\boldsymbol{\xi}^\wedge \triangleq \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix}. \tag{2.8}$$

Linearizing the transform for small perturbations $\delta\boldsymbol{\xi}$ yields a linear least-squares problem:

$$\mathcal{L}(\delta\boldsymbol{\xi}) = \frac{1}{2} \sum_{i=1}^{N_t} \left(\mathbf{e}_{i,t}^{(n)} - \mathbf{J}_{i,t}^{(n)} \delta\boldsymbol{\xi}\right)^T \mathbf{R}^{-1} \left(\mathbf{e}_{i,t}^{(n)} - \mathbf{J}_{i,t}^{(n)} \delta\boldsymbol{\xi}\right) \tag{2.9}$$

Here, $\mathbf{J}_{i,t}^{(n)}$ is the Jacobian matrix of the reprojection error. The explicit form of the Jacobian matrix is omitted for brevity but can be found in our supplemental materials.[1]

Rearranging, we see the minimizing perturbation is the solution to a linear system of

[1] `http://groups.csail.mit.edu/rrg/peretroukhin_icra16/supplemental.pdf`

equations:

$$\delta\boldsymbol{\xi}^{(n)} = \left(\sum_{i=1}^{N_t} \mathbf{J}_{i,t}^T \mathbf{R}_{,}^{-1} \mathbf{J}_{i,t}\right)^{-1} \sum_{i=1}^{N_t} \mathbf{J}_{i,t}^T \mathbf{R}^{-1} \mathbf{e}_{i,t}^{(n)}. \tag{2.10}$$

We then update the estimated transform and proceed to the next iteration.

$$\mathbf{T}_t^{(n+1)} = \exp\left(\delta\boldsymbol{\xi}^{(n)\wedge}\right) \mathbf{T}_t^{(n)}. \tag{2.11}$$

There are many reasonable choices for both the initial transform $\mathbf{T}_t^{(0)}$ and for the conditions under which we terminate iteration. We initialize the estimated transform to identity, and iteratively perform the update given by eq. (2.11) until we see a relative change in the squared error of less than one percent after an update.

2.2 Robust Estimation

2.2.1 The effects of outliers

2.2.2 M-estimation

2.3 Matrix Lie Groups

2.3.1 Manifolds

2.3.2 Local vs. Global Perturbations

Chapter 3

Predictive Robust Estimation

3.1 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. We assert that 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.

3.2 Related Work

There is a large and growing body of work on the problem of deriving accurate, consistent state estimates from visual data. Although our approach to noise modelling is applicable in other domains, for simplicity we focus our attention on the problem of inferring ego-motion from features extracted from sequential pairs of stereo images; see Sünderhauf and Protzel (2007) for a survey of techniques. The spectrum of alternative approaches to visual state estimation include monocular techniques, which may be feature-based (Scaramuzza and Fraundorfer, 2011a), direct (Irani and Anandan, 2000), or semi-direct (Forster et al., 2014b).

Apart from simply rejecting outliers, a number of recent approaches attempt to select the optimal set of features to produce an accurate localization estimate from tracked visual features. For example, Tsotsos et al. (2015b) amend Random Sample Consensus (RANSAC) with statistical hypothesis testing to ensure that tracked visual features have normally distributed residuals before including them in the estimator. Unlike our predictive approach, their technique relies on the availability of feature tracks, and requires scene overlap to work continuously. In a different approach, Zhang and Vela (2015) choose an optimally observable feature subset for a monocular SLAM pipeline by selecting features with the highest *informativeness* - a measure calculated based on the observability of the SLAM subsystem. 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.

3.3 Scalar k-Nearest Neighbours

In this chapter we present PROBE, a Predictive ROBust Estimation technique that improves localization accuracy in the presence of such effects by building a model of the uncertainty in the affected visual observations. We learn the model in an offline training procedure and then use it online to predict the uncertainty of incoming observations as a function of their location in a predefined *prediction space*. Our model can be learned in completely unknown environments with frequent or infrequent ground truth data.

The primary contributions of this research are a flexible framework for learning the quality of visual features with respect to navigation estimates, and a straightforward way to incorporate this information into a navigation pipeline. On its own, PROBE can produce more accurate estimates than a binary outlier rejection scheme like Random Sample Consensus (RANSAC) **?** because it can simultaneously reduce the influence of outliers while intelligently weighting inliers. PROBE reduces the need to develop finely-tuned uncertainty models for complex sensors such as cameras, and better accounts for the effects observed in complex, dynamic scenes than typical fixed-uncertainty models. While we present PROBE in the context of visual feature-based navigation, we stress that it is not limited to visual measurements and could also be applied to other sensor modalities.

The aim of PROBE is to learn a model for the quality of visual features, with the goal of reducing the impact of deleterious visual effects such as moving objects, motion blur, and shadows on navigation estimates. Feature quality is characterized by a scalar weight, β_i, for each visual feature in an environment. To compute β_i we define a prediction space

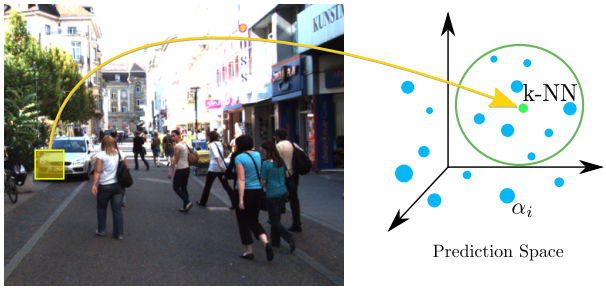


Figure 3.1: PROBE maps image features into a prediction space to predict feature quality (α). Feature quality is a function of the nearest neighbours from training data.

(similar to Vega-Brown et al. (2013)) that consists of a set of visual-inertial predictors computed from the local image region around the feature and the inertial state of the vehicle (Section 3.3.3 details our choice of predictors). We then scale the image covariance of each feature ($\mathbf{R}_a^i$, $\mathbf{R}_b^i$ in **??**) by β_i during the non-linear optimization.

In a similar manner to M-estimation, PROBE achieves robustness by varying the influence of certain measurements. However, in contrast to robust cost functions that weight measurements based purely on estimation error, PROBE weights measurements based on their assessed quality.

To learn the model, we require training data that consists of a traversal through a typical environment with some measure of ground truth for the path, but not for the visual features themselves. Like many machine learning techniques, we assume that the training data is representative of the test environments in which the learned model will be used.

We learn the quality of visual features *indirectly* through their effect on navigation estimates. We define high quality features as those that result in estimates that are close to ground truth. Our framework is flexible enough that we do not require ground truth at every image and we can learn the model based on even a single loop closure error.

3.3.1 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), denoted by $\alpha_{l,s}$ for iteration l and step s, and store it at each feature's position in the prediction space (we denote the set of predictors and associated RMSE value by $\Theta_{l,s}$). The full algorithm is summarized in Figure 3.2. Note that $\alpha_{l,s}$ can be computed at each step, at intermittent steps, or for an entire path,

depending on the availability of ground truth data.

procedure TRAINPROBEMODEL
  for $l \leftarrow 1, totalLearningIterations$ **do**
    for $s \leftarrow 1, totalPathSteps$ **do**
      $f_1, \ldots, f_J \leftarrow visualFeatureSubset(l)$
      $\boldsymbol{\pi}_l^1, \ldots, \boldsymbol{\pi}_l^J \leftarrow predictors(f_1, \ldots, f_J)$
      $\bar{\mathbf{C}}_{ba}, \bar{\mathbf{r}}_a^{ba} \leftarrow poseChange(f_1, \ldots, f_J)$
      $\alpha_{l,s} \leftarrow computeRMSE(\bar{\mathbf{r}}_a^{ba}, \mathbf{r}_{a\ GT}^{ba})$
      $\Theta_{l,s} \leftarrow \left\{\boldsymbol{\pi}_{l,s}^1, \ldots, \boldsymbol{\pi}_{l,s}^J, \alpha_{l,s}\right\}$
    end for
  end for
  return $\boldsymbol{\Theta} = \{\Theta_{l,s}\}$
end procedure

Figure 3.2: The PROBE training procedure.

3.3.2 Evaluation

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 $\overline{\alpha}$, the mean α value of the entire training set.

procedure USEPROBEMODEL($\boldsymbol{\Theta}$)
  for $i \leftarrow 1, totalFeatures$ **do**
    $\boldsymbol{\pi}_i \leftarrow predictors(f_i)$
    $\alpha_1, ..., \alpha_K \leftarrow findKNN(\boldsymbol{\pi}_i, K, \boldsymbol{\Theta})$
    $\beta_i \leftarrow \left(\frac{1}{\overline{\alpha} K} \sum_{k=1}^{K} \alpha_k\right)^{\gamma}$
  end for
  return $\boldsymbol{\beta} = \{\beta_i\}$
end procedure

Figure 3.3: The PROBE evaluation procedure.

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 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.

3.3.3 Prediction Space

A crucial component of our technique is the choice of prediction space. 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:

- Angular velocity and linear acceleration magnitudes
- Local image entropy
- Blur (quantified by the blur metric of **?**)
- Optical flow variance score
- Image frequency composition

We discuss each of these predictors in turn.

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 the IMU are in themselves good predictors of image degradation (e.g., image blur) and hence poor feature tracking.

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

$$S = -\sum_{i=1}^{N} c_i \log_2(c_i), \tag{3.1}$$

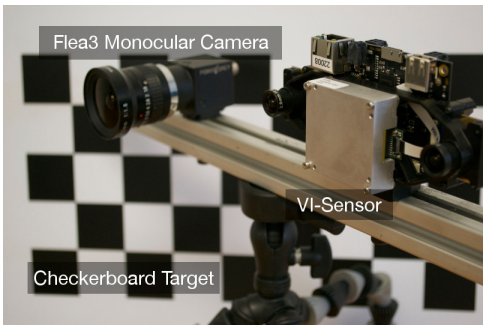


Figure 3.4: The Skybotix VI-Sensor, Point Grey Flea3, and checkerboard target used in our motion blur experiments.

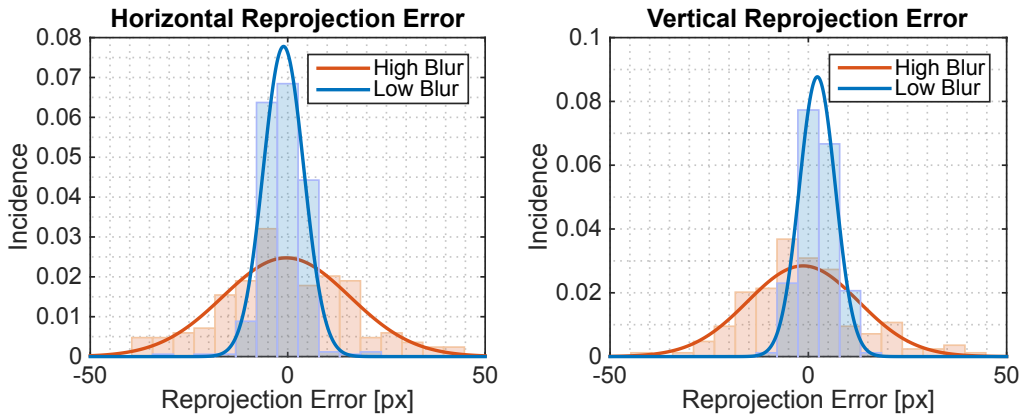


Figure 3.5: Reprojection error of checkerboard corners triangulated from the VI-Sensor and reprojected into the Flea3.

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

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 **??**). Prior to the experiment, we determined the intrinsic and extrinsic calibration parameters of our rig using the Kalibr package **?**. 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 **?**.

We detected checkerboard corners in each camera at synchronized time steps, com-

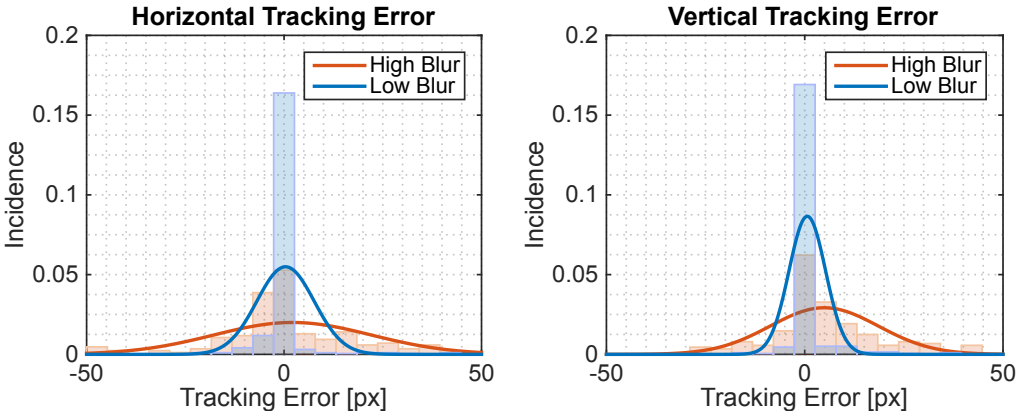


Figure 3.6: Effect of blur on reprojection and tracking error for the slow-then-fast checkerboard dataset. We distinguish between high and low blur by thresholding the blur metric **?**. The variance in both errors increases with blur.

puted 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 **?** and re-detection.

Figure **??** 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.

Optical flow variance score

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.

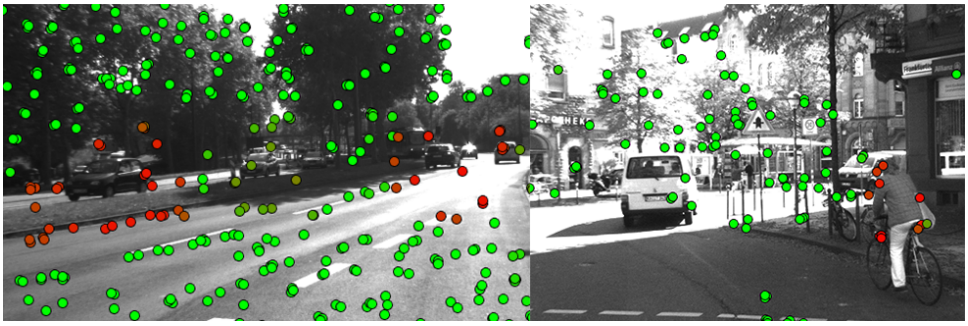


Figure 3.7: The optical flow variance predictor can help in detecting moving objects. Red circles correspond to higher values of the optical flow variance score (i.e., features more likely to belong to a moving object).

We compute this optical flow variance score according to

$$\log\left(\frac{\bar{\sigma}_s^2}{\bar{\sigma}_l^2}\right), \tag{3.2}$$

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 **??** 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.

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

3.4 Generalized Kernels

However, not all features are created equal; most feature-based methods rely on random sample consensus algorithms (Fischler and Bolles, 1981) to partition the extracted features into inliers and outliers, and perform estimation based only on inliers. It is

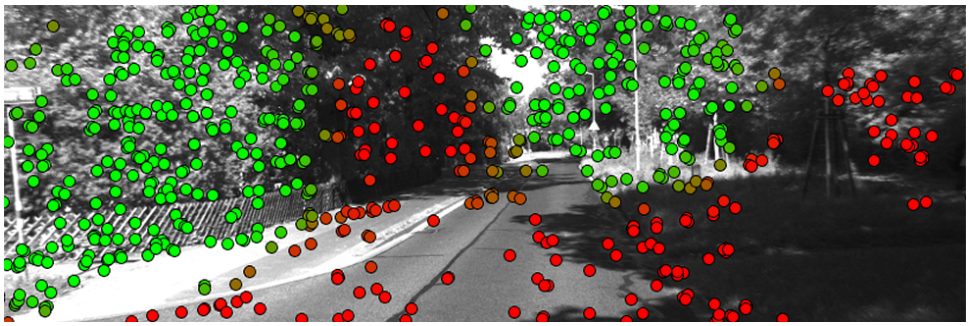


Figure 3.8: A high-frequency predictor can distinguish between regions of high and low texture such as foliage and shadows. Green indicates higher values.

Table 3.1: 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.

Figure 3.9: 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 3.10: Our four-wheeled skid-steered Clearpath Husky rover equipped with Skybotix VI-Sensor and Ashtech DGPS antenna used to collect the outdoor UTIAS dataset.

common to guard against misclassifying an outlier as an inlier by using robust estimation techniques, such as the Cauchy costs employed in Kerl et al. (2013) or the dynamic covariance scaling devised by Agarwal et al. (2013b). These approaches, often grouped under the title of M-estimation, aim to maintain a quadratic influence of small errors, while reducing the contribution of larger errors. The robustness and accuracy of feature-based visual odometry often hinges on the tuning of the parameters of inlier selection and robust estimation. Performance can vary significantly from one environment to the next, and most algorithms require careful tuning to work in a given environment.

In this paper, we describe a principled, data-driven way to build a noise model for visual odometry. We combine our previous work (Peretroukhin et al., 2015b) on predictive robust estimation (PROBE) with our work on covariance estimation (Vega-Brown and Roy, 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 over the noise, 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

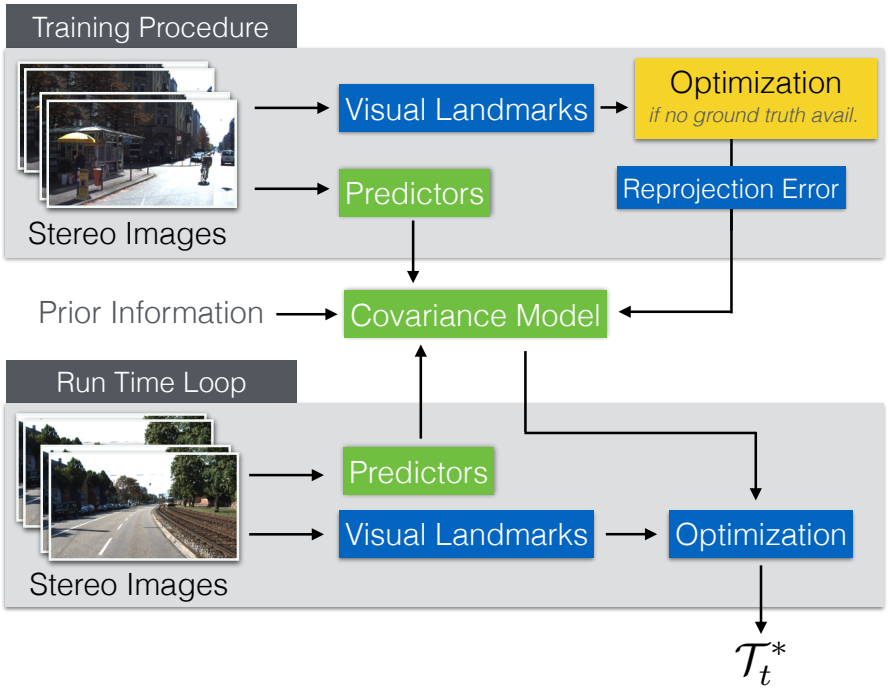


Figure 3.11: Our proposed system builds a predictive noise model for stereo visual odometry. (a) At training time, we extract landmarks from two pairs of stereo images, and use egomotion ground truth to compute reprojection errors to build a covariance model. (b) At run time, we predict a covariance for each visual landmark. We use these covariances in a robust nonlinear least-squares problem, which is solved to estimate the transform between camera poses. (c) If the ground truth egomotion is not known, we iteratively apply an optimization procedure (yellow box) to estimate them.

to produce more accurate odometry estimates. Our pipeline is outlined in Figure 3.11.

3.4.1 Predictive noise models for visual odometry

The process described in the previous section employs a fixed noise covariance $\mathbf{R}$. However, not all landmarks are created equal: differing texture gradients can cause feature localization to degrade in predictable ways, and effects like motion blur can lead to landmarks being less informative. If we had a good estimate of the noise covariance for each landmark, we could simply replace the fixed covariance $\mathbf{R}$ with one that varies for each stereo observation, $\mathbf{R}_{i,t}$. Such a predictive model would allows us to better account for observation errors from a diverse set of noise sources, and incorporate information from landmarks that may otherwise be discarded by a binary outlier rejection scheme.

However, estimating these covariances in a principled way is a nontrivial task. Even when we have reasonable heuristic estimates available, it is difficult to guarantee those estimates will be reliable. Instead of relying solely on such heuristics, we propose to learn these image-space noise covariances from data.

We associate with each landmark $\mathbf{y}_{i,t}$ a vector of *predictors*, $\boldsymbol{\phi}_{i,t} \in \mathbb{R}^M$. Each predictor can be computed using both visual and inertial cues, allowing us to model effects like motion blur and self-similar textures. We then compute the covariance as a function of these predictors, so that $\mathbf{R}_{i,t} = \mathbf{R}(\boldsymbol{\phi}_{i,t})$. 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 $\boldsymbol{\Psi} \in \mathbb{R}^{d \times d} \succ 0$ and a scalar quantity called the degrees of freedom $\nu \in \mathbb{R} > d - 1$:

$$\begin{aligned} p(\mathbf{R}) &= \mathrm{IW}(\mathbf{R}; \boldsymbol{\Psi}, \nu) \\ &= \frac{|\boldsymbol{\Psi}|^{\nu/2}}{2^{\frac{\nu d}{2}} \Gamma_d(\frac{\nu}{2})} |\mathbf{R}|^{-\frac{\nu+d+1}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Psi}\mathbf{R}^{-1}\right)\right). \end{aligned} \tag{3.3}$$

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 $\boldsymbol{\Psi}(\boldsymbol{\phi}) = n\hat{\mathbf{R}}$.

Given a sequence of observations and ground truth transformations,

$$\mathcal{D} = \{\mathcal{I}_t, \mathbf{T}_t\}, \quad t \in [1, N] \tag{3.4}$$

where

$$\mathcal{I}_t = \{\mathbf{y}_{i,t}, \mathbf{y}'_{i,t}, \boldsymbol{\phi}_{i,t}\} \quad i \in [1, N_t], \tag{3.5}$$

we can use the procedure of generalized kernel estimation (Vega-Brown et al., 2014b) to infer a posterior distribution over the covariance matrix $\mathbf{R}_*$ associated with some query predictor vector $\boldsymbol{\phi}_*$:

$$\begin{aligned} p(\mathbf{R}_*|\mathcal{D}, \boldsymbol{\phi}_*) &\propto \prod_{i,t} \mathcal{N}(\mathbf{e}_{i,t}|\mathbf{0}, \mathbf{R}_*)^{k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t})} \\ &\quad \times \mathrm{IW}(\mathbf{R}_*; \boldsymbol{\Psi}(\boldsymbol{\phi}_*), \nu(\boldsymbol{\phi}_*)) \qquad (3.6) \\ &= \mathrm{IW}(\mathbf{R}_*; \boldsymbol{\Psi}_*, \nu_*). \qquad (3.7) \end{aligned}$$

Here, $\mathbf{e}_{i,t} = \mathbf{y}'_{i,t} - f(\mathbf{T}_t f^{-1}(\mathbf{y}_{i,t}))$ 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 $\boldsymbol{\Psi}_*$ and ν_* can be computed in closed form as

$$\boldsymbol{\Psi}_* = \boldsymbol{\Psi}(\boldsymbol{\phi}_*) + \sum_{i,t} k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t}) \mathbf{e}_{i,t}\, {\mathbf{e}_{i,t}}^T, \tag{3.8}$$

$$\nu_* = \nu(\boldsymbol{\phi}_*) + \sum_{i,t} k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t}). \tag{3.9}$$

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

$$\begin{aligned} &p(\mathbf{y}'_{i,t}|\mathbf{T}_t, \mathbf{y}_{i,t}, \mathcal{D}, \boldsymbol{\phi}_{i,t}) \qquad (3.10) \\ &\quad = \int \mathrm{d}\mathbf{R}_{i,t} \mathcal{N}\left(\mathbf{e}_{i,t}; \mathbf{0}, \mathbf{R}_{i,t}\right) \mathrm{IW}(\mathbf{R}_{i,t}; \boldsymbol{\Psi}_*, \nu_*) \qquad (3.11) \\ &\quad = \mathrm{t}_{\nu_*-d+1}\left(\mathbf{e}_{i,t}; \mathbf{0}, \frac{1}{\nu_* - d + 1}\boldsymbol{\Psi}_*\right) \qquad (3.12) \\ &\quad = \frac{\Gamma\left(\frac{\nu_*+1}{2}\right)}{\Gamma\left(\frac{\nu_*-d+1}{2}\right)} |\boldsymbol{\Psi}_*|^{-\frac{1}{2}} \pi^{-\frac{d}{2}} \left(1 + {\mathbf{e}_{i,t}}^T {\boldsymbol{\Psi}_*}^{-1} \mathbf{e}_{i,t}\right)^{-\frac{\nu_*+1}{2}}. \qquad (3.13) \end{aligned}$$

Given a new landmark and predictor vector, we can infer a noise model by evaluating eqs. (3.8) and (3.9). 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 $\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|_2 > \rho$. 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. (3.8) and (3.9) 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,t}, y'_{i,t}, φ_{i,t}} in I_t do
            e_{i,t} = y'_{i,t} − f(T_t f^{-1}(y_{i,t}))
            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

$$\mathbf{T}_t^* = \underset{\mathbf{T}_t \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} (\nu_{i,t} + 1) \log\left(1 + {\mathbf{e}_{i,t}}^T \, \boldsymbol{\Psi}_{i,t}^{-1} \mathbf{e}_{i,t}\right). \tag{3.14}$$

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 (3.14) 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 (3.14) as a subroutine.

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.

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 $\{\mathbf{y}_{i,t}, \mathbf{y}'_{i,t}, \boldsymbol{\phi}_{i,t}\}$ in $\mathcal{I}_t$ **do**
    $\boldsymbol{\Psi}, \nu \leftarrow$ InferNoiseModel($\mathcal{M}$, $\boldsymbol{\phi}_{i,t}$)
    $g(\mathbf{T}) = \mathbf{y}_{i,t} - f(\mathbf{T} f^{-1}(\mathbf{y}'_{i,t}))$
    $\mathcal{L} \leftarrow \mathcal{L} + (\nu + 1) \log \left(1 + g(\mathbf{T})^T \boldsymbol{\Psi}^{-1} g(\mathbf{T})\right)$
  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($\mathcal{M}, \boldsymbol{\phi}_*, \rho$)
    ▷ ρ is the radius of the support of the 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

3.4.2 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:

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

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 follow our 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:

$$Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)}) = \int \left(\prod_{i,t} \mathrm{d}\mathbf{R}_{i,t}\, p(\mathbf{R}_{i,t}|\mathcal{D}_{\setminus i,t}, \mathbf{T}_{1:T}^{(n)}) \right) \times \log \prod_{i,t} p(\mathbf{y}'_{i,t}|\mathbf{y}_{i,t}, \mathbf{T}_t, \mathbf{R}_{i,t}). \quad (3.15)$$

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:

$$Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)}) \cong -\frac{1}{2} \sum_{t=1}^{T} \sum_{i=1}^{N_t} \mathbf{e}_{i,t}{}^T \left(\frac{1}{\nu_{i,t}^{(n)}} \mathbf{\Psi}_{i,t}^{(n)} \right)^{-1} \mathbf{e}_{i,t}. \quad (3.16)$$

The symbol $\cong$ is used to indicate equality up to an additive constant. A derivation of this observation can be found in our supplemental material.

We can iteratively refine our estimate by maximizing the expected log-likelihood

$$\mathbf{T}_{1:T}^{(n+1)} = \underset{\mathbf{T}_{1:T} \in \mathrm{SE}(3)^T}{\operatorname{argmax}} Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)}). \quad (3.17)$$

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:

$$\mathbf{T}_{t}^{(n+1)} = \underset{\mathbf{T}_t \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} \mathbf{e}_{i,t}{}^T \left(\frac{1}{\nu_{i,t}^{(n)}} \mathbf{\Psi}_{i,t}^{(n)} \right)^{-1} \mathbf{e}_{i,t}. \quad (3.18)$$

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 (3.14)) in place of Equation (3.18) 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 part of ongoing work.

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 $\{\mathbf{y}_{i,t}, \mathbf{y}'_{i,t}, \boldsymbol{\phi}_{i,t}\}$ in $\mathcal{I}_t$ **do**
      $\mathbf{e}_{i,t} = \mathbf{y}_{i,t} - f(\mathbf{T}_t^{(0)} f^{-1}(\mathbf{y}'_{i,t}))$
      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 $\{\mathbf{y}_{i,t}, \mathbf{y}'_{i,t}, \boldsymbol{\phi}_{i,t}\}$ in $\mathcal{I}_t$ **do**
        $\boldsymbol{\Psi}, \nu \leftarrow$ InferNoiseModel($\mathcal{M}, \boldsymbol{\phi}_{i,t}$)
        $g(\mathbf{T}) = \mathbf{y}_{i,t} - f(\mathbf{T} f^{-1}(\mathbf{y}'_{i,t}))$
        $\mathcal{L} \leftarrow \mathcal{L} + g(\mathbf{T})^T \left(\frac{1}{\nu}\boldsymbol{\Psi}\right)^{-1} g(\mathbf{T})$
      end for
      $\mathbf{T}_t \leftarrow \operatorname{argmin}_{\mathbf{T} \in \mathrm{SE}(3)} \mathcal{L}(\mathbf{T})$
      $\mathbf{e}_{i,t} = \mathbf{y}_{i,t} - f(\mathbf{T}_t^{(0)} f^{-1}(\mathbf{y}'_{i,t}))$
      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

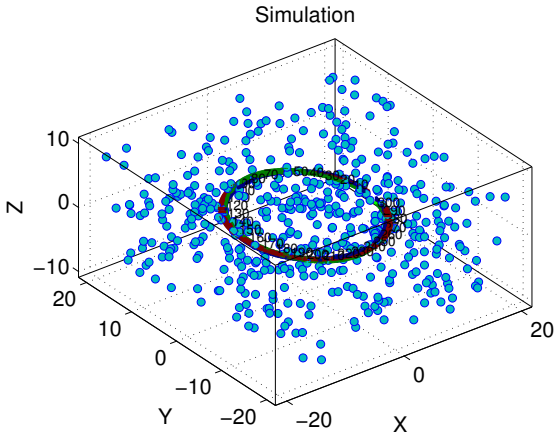


Figure 3.12: Our synthetic world. A stereo camera rig moves through a world with 2000 point features.

3.4.3 Experiments

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 3.13 and **??** document the qualitative and quantitative comparisons of PROBE-GK (trained with and without ground-truth) against two baseline stereo odometry frameworks. Both baseline estimators were implemented based on **??**. 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

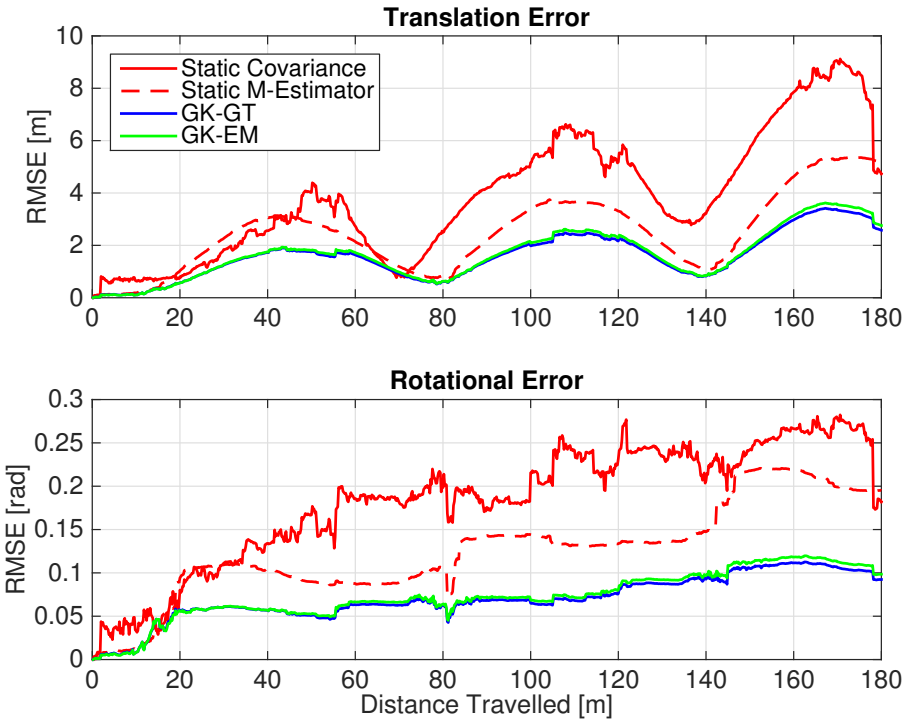


Figure 3.13: A comparison of translational and rotational Root Mean Square Error on simulated data (RMSE) for four different stereo-visual odometry pipelines: two baseline bundle adjustment procedures with and without a robust Student's t cost with a fixed and hand-tuned covariance and degrees of freedom (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 **??**.

achieve nearly identical results to the ground-truth-aided model within 5 iterations.

3.4.4 KITTI

To evaluate PROBE-GK on real environments, we trained and tested several models on the KITTI Vision Benchmark suite (Geiger et al., 2012, 2013a), 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 driving; for our work, we focused on the city, residential and road categories (Figure 3.14). From each category, we chose two separate trials for training and testing.

Figure 3.14: 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.

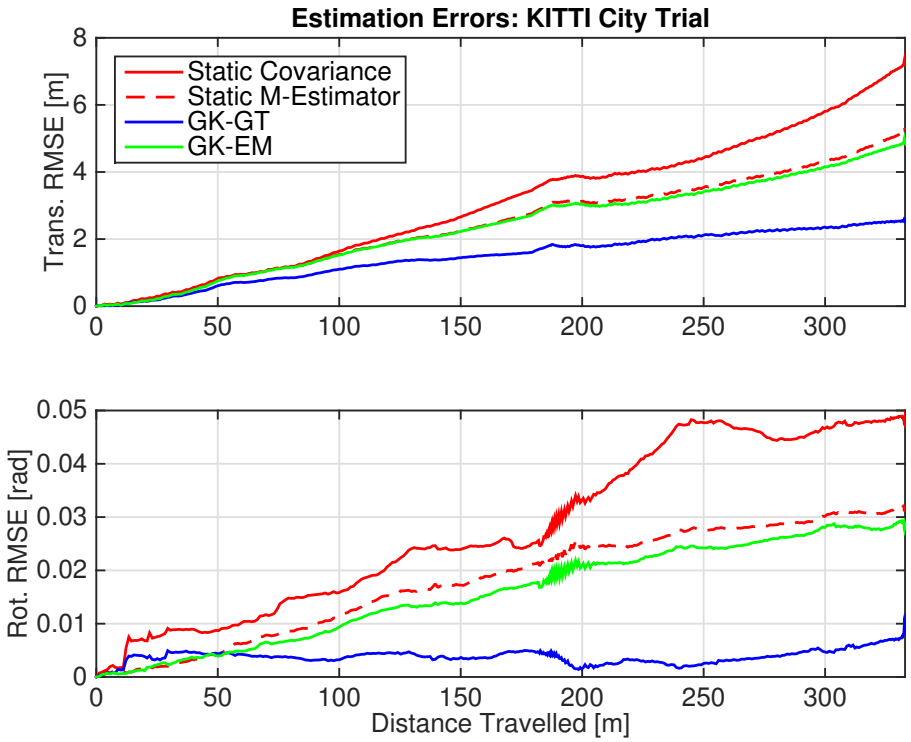


Figure 3.15: RMSE comparison of stereo odometry estimators evaluated on data from the city category in the KITTI dataset. See **??** for a quantitative summary.

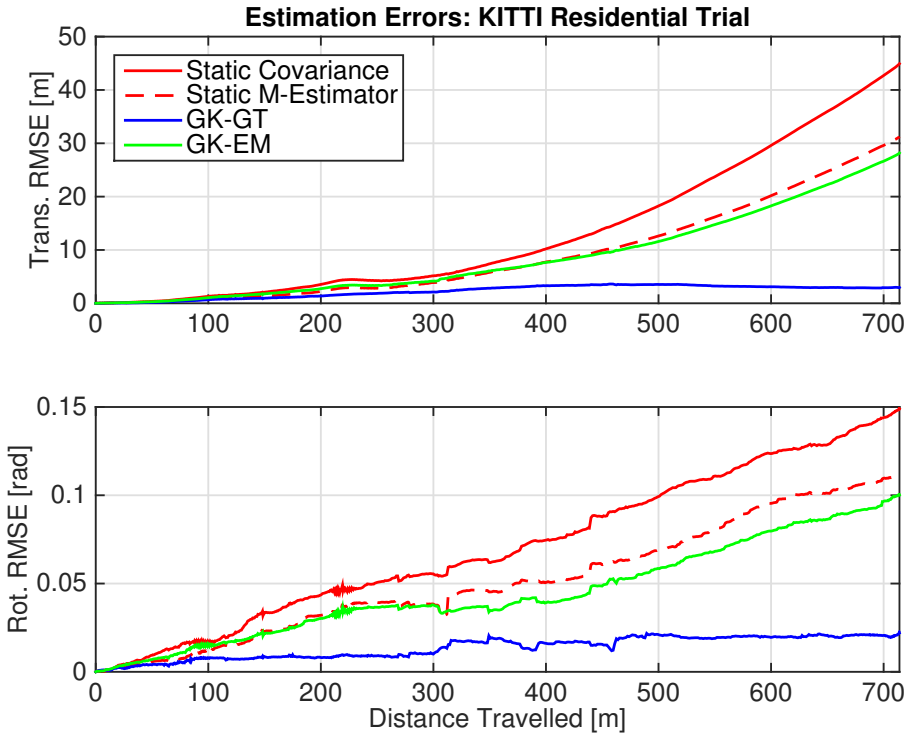


Figure 3.16: RMSE comparison of stereo odometry estimators evaluated on data from the residential category in the KITTI dataset. See **??** for a quantitative summary.

Our prediction space consisted of inertial magnitudes, high and low image frequency coefficients, image entropy, pixel location, and estimated transform parameters. The choice of predictors is motivated by the types of effects we wish to capture (in this case: grassy self-similar textures, as well as shadows, and motion blur). For a more detailed explanation of our choice of prediction space, see our previous work (Peretroukhin et al., 2015b).

?????? show typical results; **??** 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 are still actively exploring why this is the case; 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.

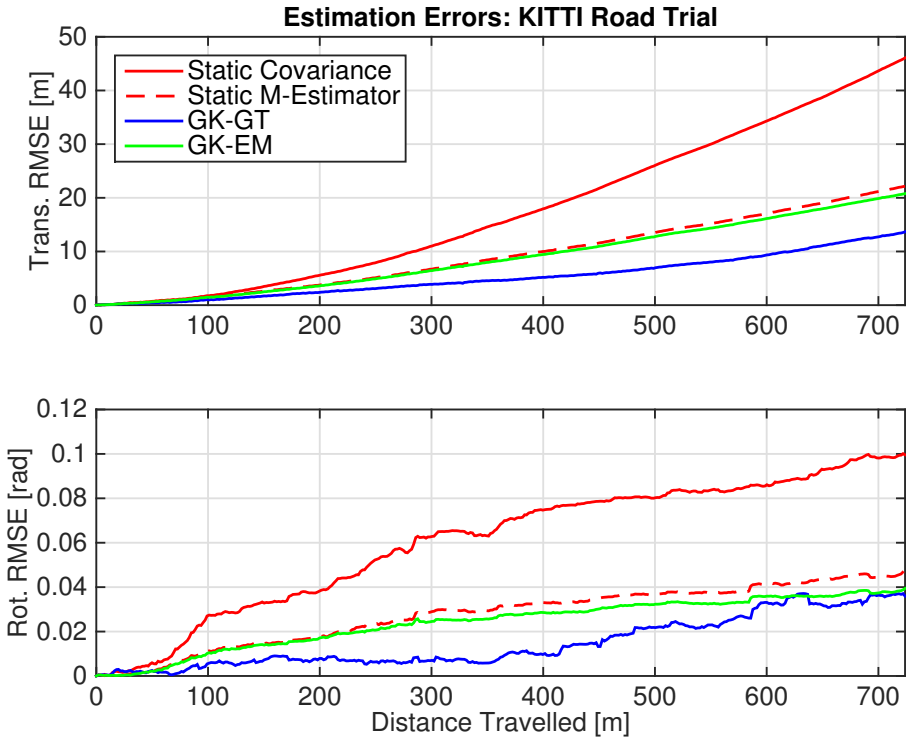


Figure 3.17: RMSE comparison of stereo odometry estimators evaluated on data from the road category in the KITTI dataset. See **??** for a quantitative summary.

Table 3.2: 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

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 Mars-Dome (an indoor Mars analog testing environment) at UTIAS (Figure 3.18) 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 3.19. Note that the GPS data was not

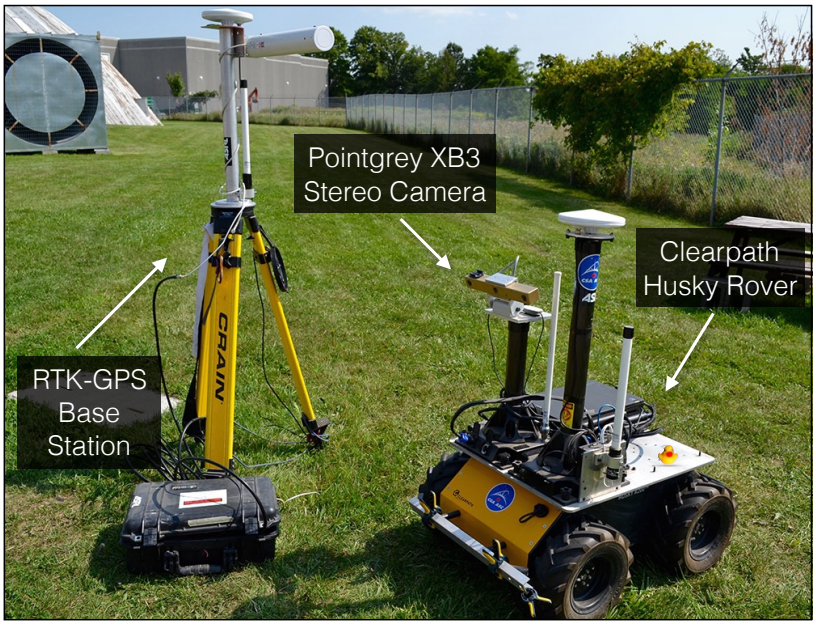


Figure 3.18: Our experimental apparatus: a Clearpath Husky rover outfitted with a PointGrey XB3 stereo camera and a differential GPS receiver and base station.

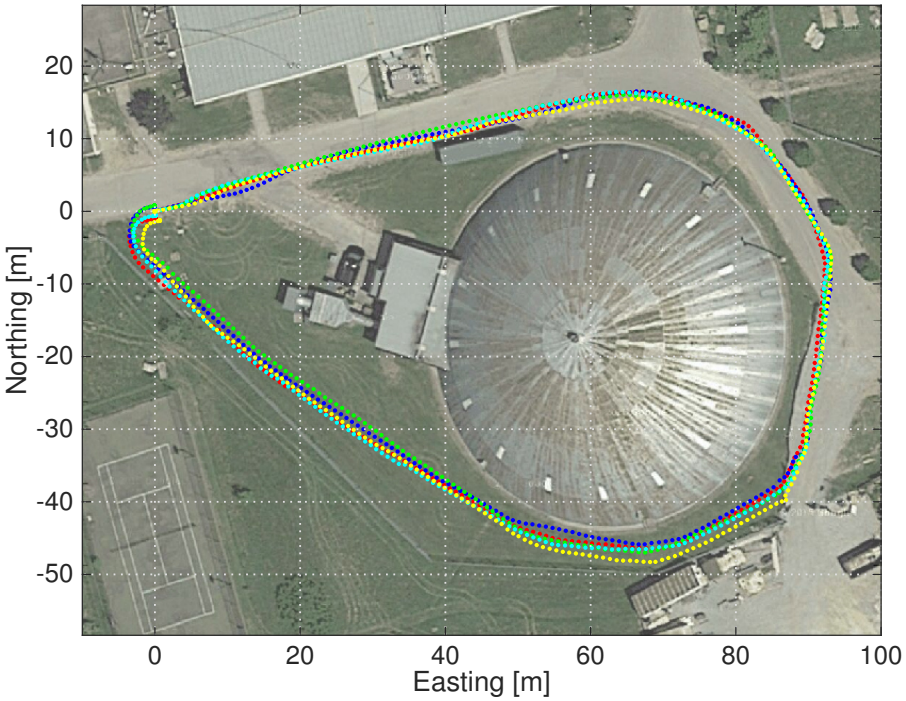


Figure 3.19: GPS ground truth for 5 experimental trials collected near the UTIAS Mars Dome. Each trial is approximately 250 m long.

Table 3.3: 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

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 Expectation Maximization approach. **??** 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). **??** lists the statistics for each test.

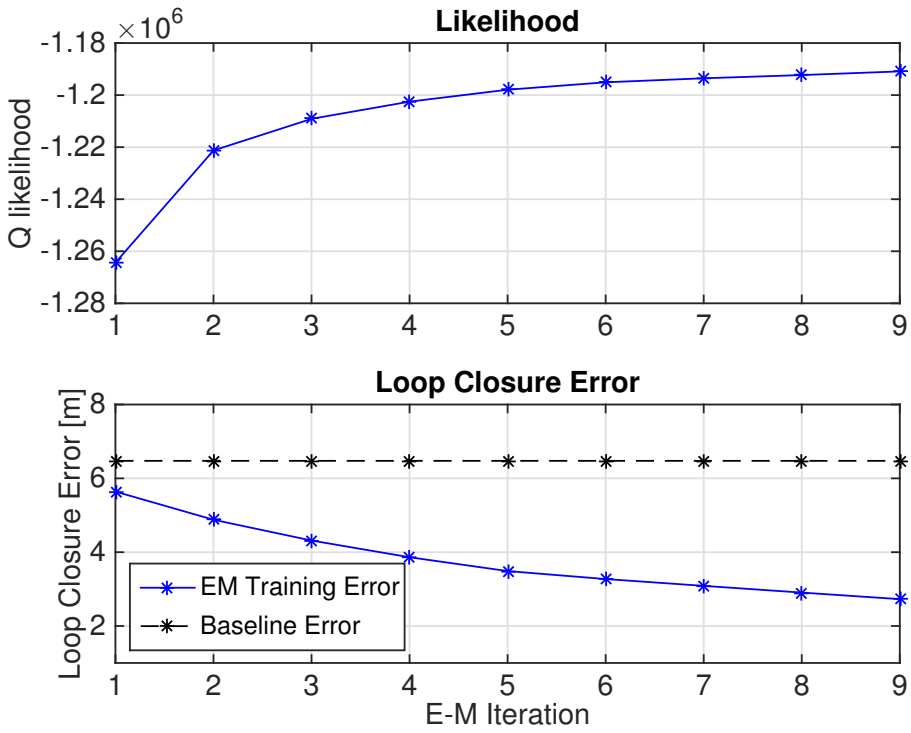


Figure 3.20: Training without ground truth using PROBE-GK-EM on a 250.2m path around the Mars Dome at UTIAS. The likelihood of the data increases with each iteration, and the loop closure error decreases, improving significantly from a baseline static M-estimator.

Chapter 4

Sun-BCNN

4.1 Motivation

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

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

In this work, we seek to answer the question of whether similar reductions in VO

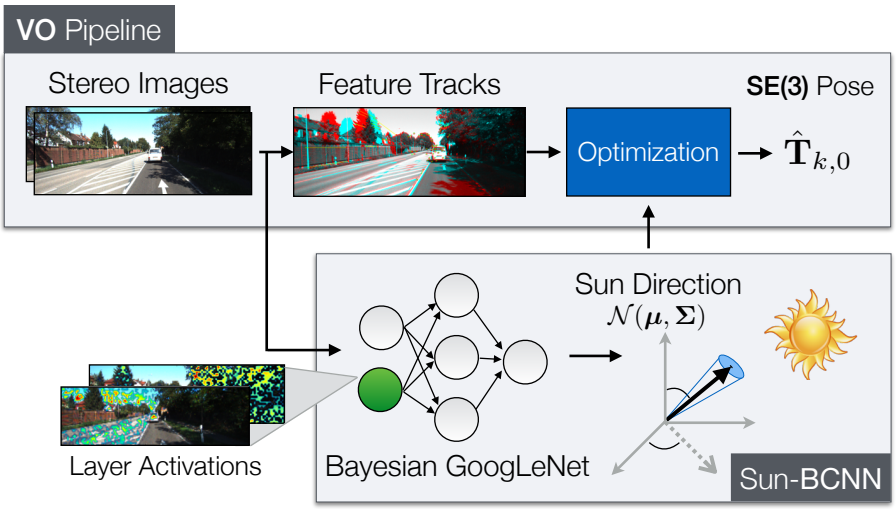


Figure 4.1: Our method uses a Bayesian Convolutional Neural Network (BCNN) to estimate the direction of the sun and to produce a principled uncertainty estimate for each prediction. We incorporate this *virtual sun sensor* into a stereo visual odometry pipeline to reduce estimation error.

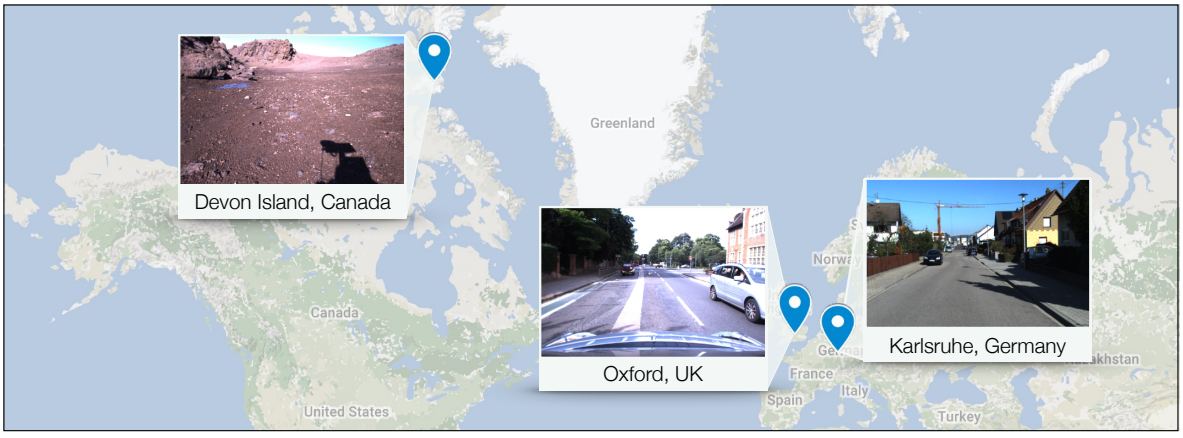


Figure 4.2: We train and test Sun-BCNN in a variety of environments ranging from urban driving in Europe to remote planetary analogue sites in the Canadian High Arctic. (Map data: Google, INEGI, ORION-ME.)

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

Our main contributions can be summarized as follows:

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

2. We demonstrate 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. We learn a 3D unit-length sun direction vector, appropriate for full 6-DOF pose estimation;

4. We present experimental results on over 30 km of visual navigation data in urban (**?**) and planetary analogue (**?**) environments;

5. We investigate the sensitivity of our Bayesian CNN-based sun estimator (Sun-BCNN) to cloud cover, camera and environment changes, and measurement parameterization; and

6. We release Sun-BCNN as open-source [1].

The remainder of this paper begins with a discussion of related work, followed by an overview of the theory underlying BCNNs and a discussion of our model architecture, implementation, and training procedure. We then outline our chosen visual odometry pipeline, which is based on a two-frame bundle adjustment optimization, and describe how observations of the sun can be incorporated directly into the motion estimation problem following the technique of **?**. 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

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

the KITTI odometry benchmark training set (**?**), as well as a further 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 (**?**). 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 (**?**). We also examine the impact of different methods for computing the mean and covariance of a norm-constrained vector on the accuracy and consistency of the estimated sun directions.

4.2 Related Work

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

Independent of the estimator, VO exhibits superlinear error growth, and is particularly sensitive to errors in orientation (**?**Cvišić and Petrović, 2015). One way to reduce orientation error is to incorporate observations of a landmark whose position or direction in the navigation frame is known *a priori*. The sun is an example of such a known directional landmark. Accordingly, hardware sun sensors have been used to improve the accuracy of VO in planetary analogue environments (e.g., the Sinclair Interplanetary SS-411 sun sensor used by **?** and **?**), while the MERs articulated their Pancam apparatus to directly image the sun (**??**). 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 (**?**). Some of these methods have been based on hand-crafted illumination cues such as shadows and variation in sky brightness (**??**), 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., 2015a). 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 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 **?** noted, however, this can be misleading because these values can be unjustifiably large for test points far away from training data. To address this, **?** 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 (**?**) on camera pose regression.

We build on on previous work by **?**, who demonstrated empirically that techniques for single-image sun estimation based on hand-crafted models (**?**) 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 **?**. We also build on the work of Peretroukhin et al. (2017a), who presented preliminary experimental results comparing Sun-BCNN against the method of **?** and its VO-informed variant (**?**) as well as the Sun-CNN of Ma et al. (2016) on the KITTI odometry benchmark (**?**Geiger et al., 2013b), 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 **?**). Furthermore, our Bayesian CNN includes a dropout layer after every convolutional and fully connected layer (as outlined by **?** but not done by Kendall and Cipolla (2016)), which produces more principled covariance outputs.

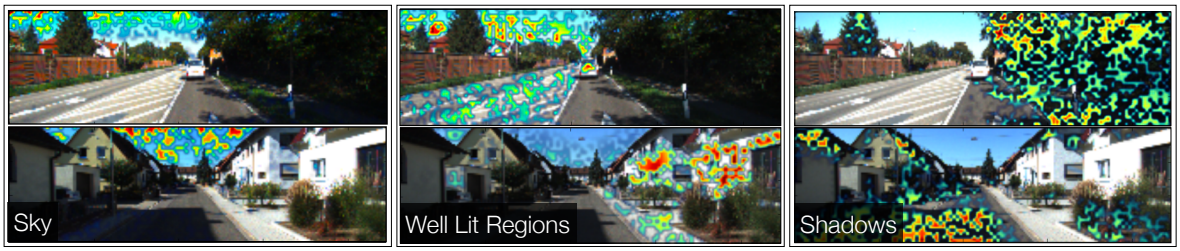


Figure 4.3: Three `conv1` layer activation maps superimposed on two images from the KITTI odometry benchmark (**?**) `00` and `04` for three selected filters. Each filter picks out salient parts of the image that aid in sun direction inference.

4.3 Indirect Sun Detection using a Bayesian Convolutional Neural Network

We use a Bayesian Convolutional Neural Network (BCNN) 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 **?** and Ma et al. (2016), who demonstrated that a CNN-based sun detector can substantially outperform hand-crafted models such as that of **?** both in terms of measurement accuracy and in its application to a VO task.

We choose a deep neural network structure based on GoogLeNet (**?**) due to its use in past work that adapted it for orientation regression (Kendall and Cipolla, 2016; **?**). Unlike Ma et al. (2016), we choose to transfer weights trained on the MIT Places dataset (**?**) rather than ImageNet (**?**). 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.

4.3.1 Cost Function

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

$$\mathcal{L}\left(\hat{\mathbf{s}}_k\right) = 1 - \left(\hat{\mathbf{s}}_k \cdot \mathbf{s}_k\right). \tag{4.1}$$

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 (4.1) since both vectors have unit length:

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

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

4.3.2 Uncertainty Estimation

Following recent work on Bayesian Convolutional Neural Networks (BCNNs) (**???**), we modify our model architecture to enable the computation of principled covariance estimates associated with each predicted sun direction. To achieve computationally tractable Bayesian inference with a CNN architecture, BCNNs exploit a connection between stochastic regularization (e.g., dropout, a widely used technique in deep learning to mitigate overfitting) and approximate variational inference of a Bayesian Neural Network. We outline the technique here briefly, and refer the reader to **?** for more details.

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

$$q(\boldsymbol{w}_i) = \mathbf{M}_i \text{diag}\left\{\{b_j^i\}_{j=1}^{K_i}\right\}, \tag{4.2}$$

$$b_j^i \in \text{Bernoulli}(p_i). \tag{4.3}$$

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

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

$\boldsymbol{w}$:

$$\mathbb{E}\left[\hat{\mathbf{s}}^*\right]_k = \hat{\bar{\mathbf{s}}}_k^* \approx \frac{1}{N}\sum_{n=1}^{N} \hat{\mathbf{s}}_k^*(\mathbf{x}_k^*, \boldsymbol{w}^n) \tag{4.4}$$

$$\mathbb{E}\left[\hat{\mathbf{s}}_k^* \; \hat{\mathbf{s}}^{*\,T}_k\right] \approx \tau^{-1}\mathbf{1} + \frac{1}{N}\sum_{n=1}^{N} \hat{\mathbf{s}}_k^*(\mathbf{x}_k^*, \boldsymbol{w}^n) \; \hat{\mathbf{s}}_k^*(\mathbf{x}_k^*, \boldsymbol{w}^n)^T - \hat{\bar{\mathbf{s}}}_k^* \; \hat{\bar{\mathbf{s}}}_k^{*\,T}, \tag{4.5}$$

where $\mathbf{1}$ is the identity matrix, and $\boldsymbol{w}^n$ is a sample from $q(\boldsymbol{w})$ (obtained by sampling the network with dropout). The model precision, τ, is computed as

$$\tau = \frac{pl^2}{2M\lambda}, \tag{4.6}$$

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

Following **?**, 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 (4.4) and (4.5).

4.3.3 Implementation and Training

We implement our network in Caffe (**?**), using the `L2Norm` layer from the Caffe-SL fork[2] 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 4.3 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 **?**.

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

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

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 (see Equation (4.6)) 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 (**?**); and 2) we assume the observation noise is homoscedastic. As noted by **?**, the BCNN can be made heteroscedastic by learning the model precision during training, but this extension is outside the scope of this work.

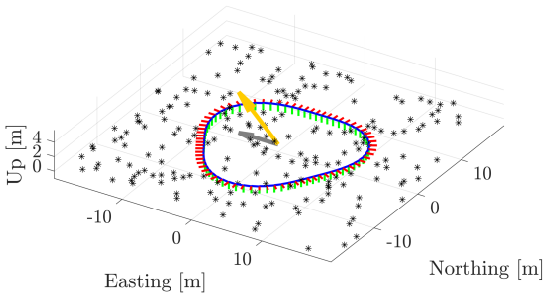
Data Partitioning

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.

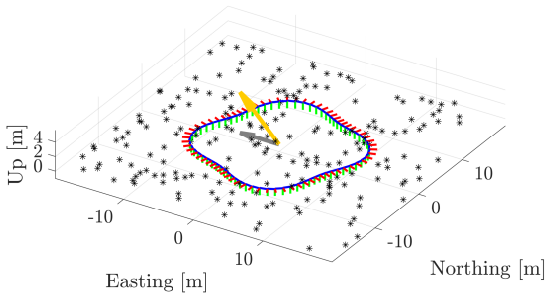
4.4 Simulation Experiments

We assess the benefit of incorporating sun observations of varying quality by conducting a series of simulation experiments consisting of a stereo camera moving along loopy trajectories of varying shapes through a simulated field of point landmarks, with a single static directional landmark representing the sun. Figure 4.4 shows several such loopy trajectories.

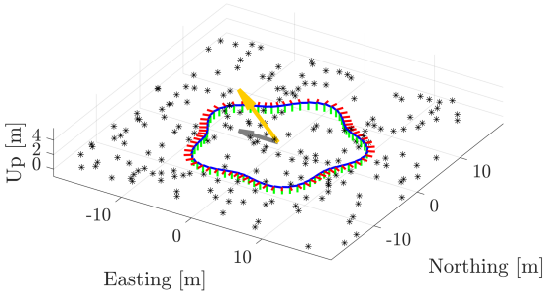
We simulate the sun at 45° of zenith and an arbitrary azimuth angle, and corrupt observations of the ground truth sun vector with artificial noise such that the mean angular distance (a non-negative quantity) between the observed and true sun direction is 0°, 10°, 20°, and 30°, labeling these conditions *GT-Sun-0*, *GT-Sun-10*, *GT-Sun-20*, and *GT-Sun-30*, respectively. In our experiments, we treated the measurement noise as



(a) "Triangle" trajectory



(b) "Square" trajectory



(c) "Star" trajectory

Figure 4.4: One loop of the "Triangle", "Square", and "Star" trajectories, consisting primarily of translation and yaw rotation. Landmarks are shown as black asterisks, and the simulated sun direction is indicated with a yellow arrow along with its projection, in grey, on the EN-plane.

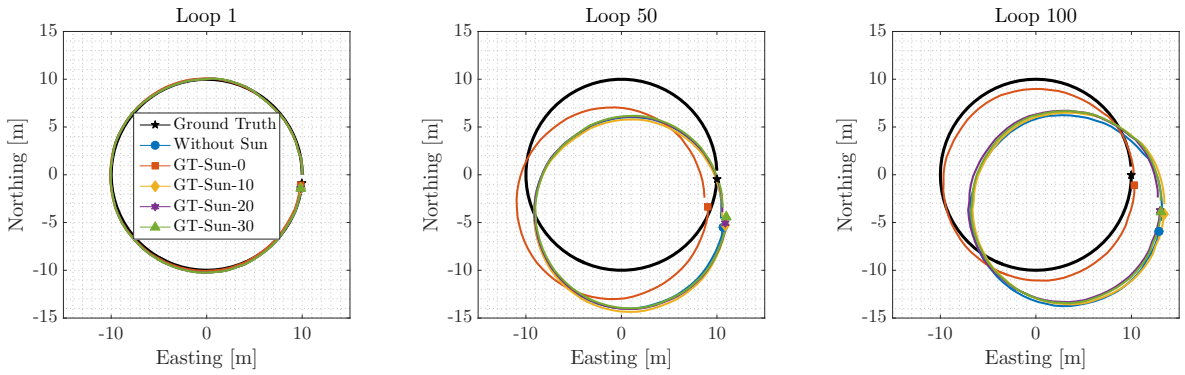
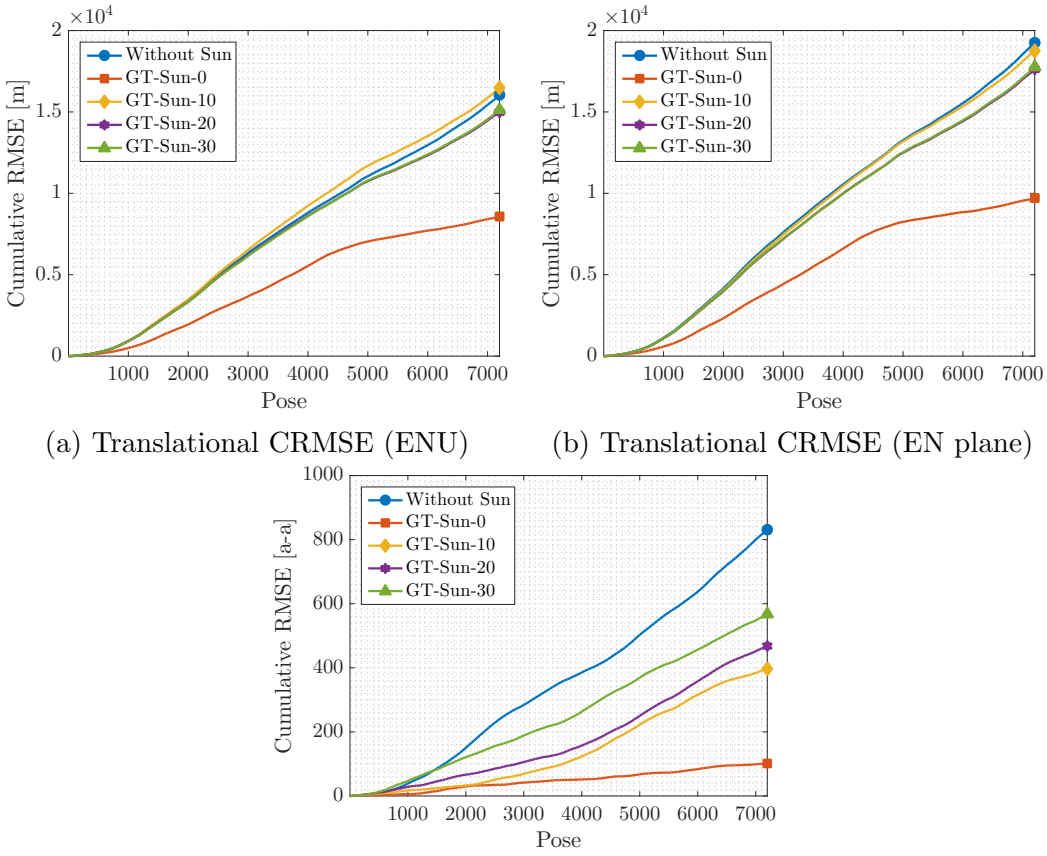


Figure 4.5: Selected segments of a 100-loop "Circle" trajectory, without sun corrections, and with sun corrections corrupted by varying levels of artificial Gaussian noise. The effect of VO drift can be clearly seen, as well as the benefit of incorporating observations of a directional landmark such as the sun.



(a) Translational CRMSE (ENU)

(b) Translational CRMSE (EN plane)

(c) Rotational CRMSE

Figure 4.6: Cumulative root mean squared error (CRMSE) of a simulated 100-loop circular trajectory, without sun corrections, and with sun corrections corrupted by varying levels of artificial Gaussian noise. The accumulated estimation error is greatly reduced by incorporating observations of the sun, and the benefit decreases as these observations become noisier.

an additive quantity sampled from a zero-mean isotropic 3D Gaussian distribution, and renormalized the resulting vectors to enforce the unit-norm constraint.

We simulate the sun at 45° of zenith and an arbitrary azimuth angle, and corrupt observations of the ground truth sun vector with artificial noise such that the mean angular distance (a non-negative quantity computed from the dot product) between the ground truth and noisy sun vectors is 0°, 10°, 20°, or 30°. We label these conditions *GT-Sun-0*, *GT-Sun-10*, *GT-Sun-20*, and *GT-Sun-30*, respectively. We generated these noisy measurements by first sampling 3-vectors from an isotropic zero-mean multivariate Gaussian distribution, then adding these vectors to the ground truth sun vector, and finally normalizing the result to unit length. We chose the covariance of this distribution to yield the desired average angular distance in each case. Note that although the distribution from which we sample noise vectors is zero-mean, the average angular distances will not be zero-mean because angular distance is non-negative.

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

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

Since our VO pipeline does not incorporate loop closures, the effects of drift in the VO solution can be clearly seen by examining individual loops in the camera trajectory. Figure 4.5 shows three loops from the "Circle" trajectory, demonstrating that the VO solution drifts significantly from the true trajectory by the 100th loop. Figure 4.6 plots the translational and rotational cumulative root mean squared error (CRMSE) for this trajectory, which measures the growth in total estimation error over time. Figure 4.6c in particular highlights the significant effect of sun sensing on rotational error, where we see a clear progression in estimation error as the sun direction observations become more noisy.

?? shows that while all four simulation trajectories display consistent and predictable

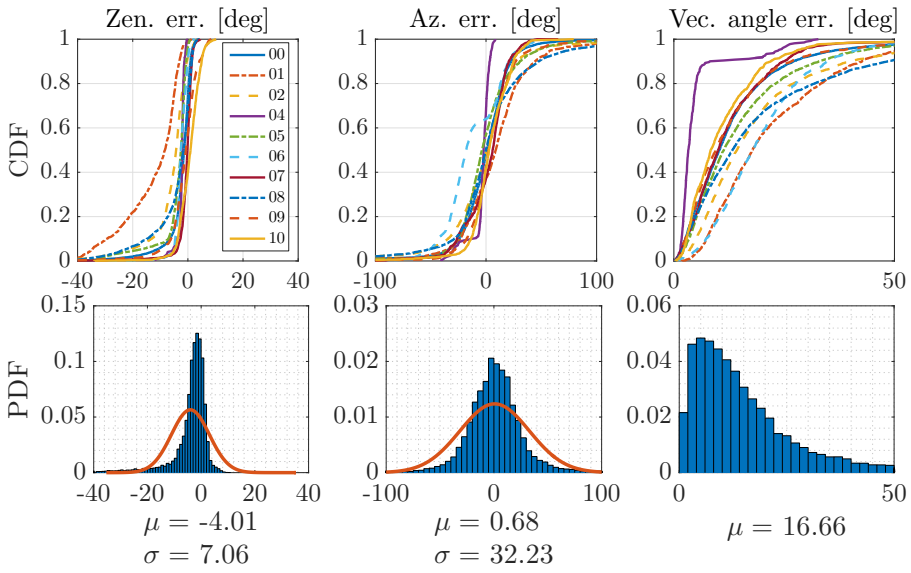


Figure 4.7: Distributions of azimuth error, zenith error, and angular distance for Sun-BCNN compared to ground truth over each test sequence in the KITTI dataset. *Top row*: Cumulative distributions of errors for each test sequence individually. *Bottom row:* Histograms and Gaussian fits of aggregated errors.

reductions in rotational average root mean squared error (ARMSE), this is not always the case for translational ARMSE. This is because translational errors are only partially induced by rotational errors, with the remainder made up of 'sliding' motions orthogonal to the direction of travel. These non-rotational errors are highly dependent on the specific trajectory, where more or less of the observed feature tracks can be explained by a sliding motion instead of a rotation. Due to the coupling of translational and rotational errors, correcting for rotational error in such cases may actually worsen the translational error (e.g., on the "Triangle" sequence).

While we do not implement this in our work, we speculate that incorporating an appropriate motion model into our VO formulation would significantly mitigate the impact of these errors by, for example, imposing a nonholonomic constraint on a ground vehicle or accounting for the dynamics of a quadcopter.

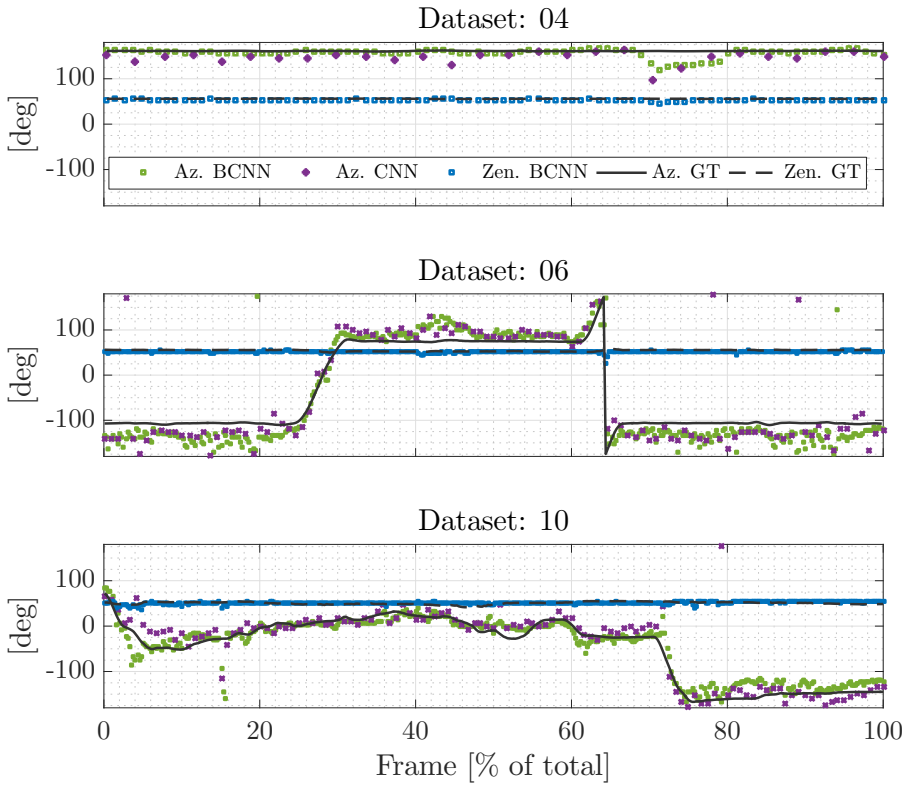


Figure 4.8: Azimuth (Sun-CNN and Sun-BCNN) and zenith (Sun-BCNN only) predictions over time for KITTI test sequences `04`, `06` and `10`. Sun-CNN is trained and tested on every tenth image, whereas Sun-BCNN is trained and tested on every image. In our VO experiments, we use the Sun-BCNN predictions of every tenth image to make a fair comparison.

Table 4.1: 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 $\tau^{-1} = 0.015$.

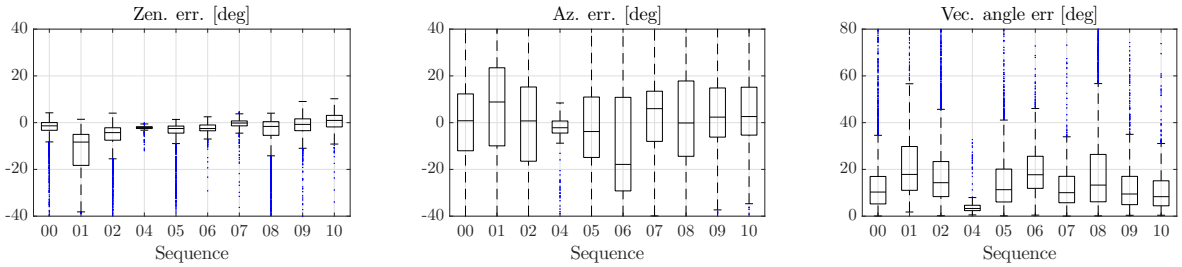
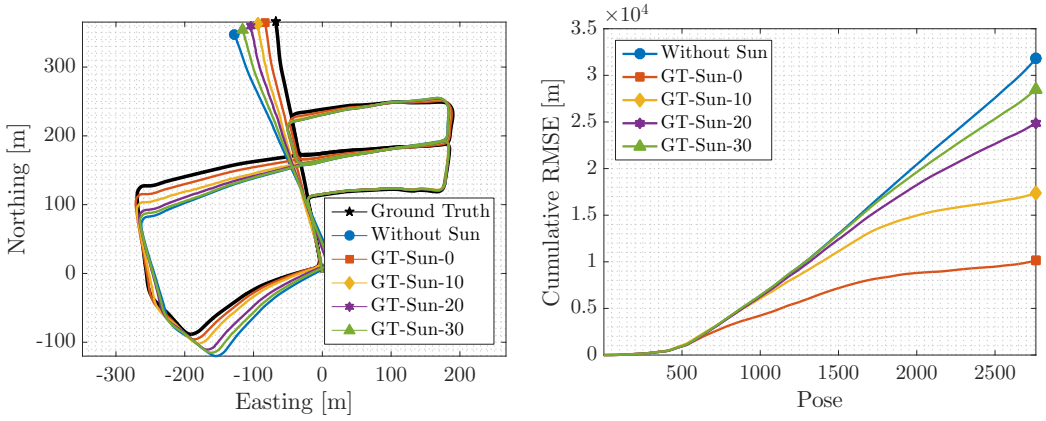
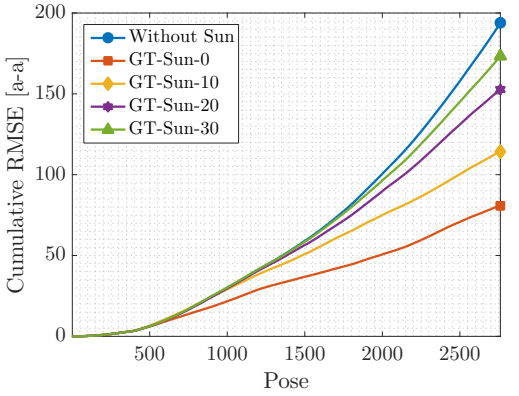


Figure 4.9: Box-and-whiskers plot of final test errors on all ten KITTI odometry sequences (c.f. Table 4.1).



(a) VO and ground truth trajectories

(b) Translational CRMSE (EN-plane)



(c) Rotational CRMSE

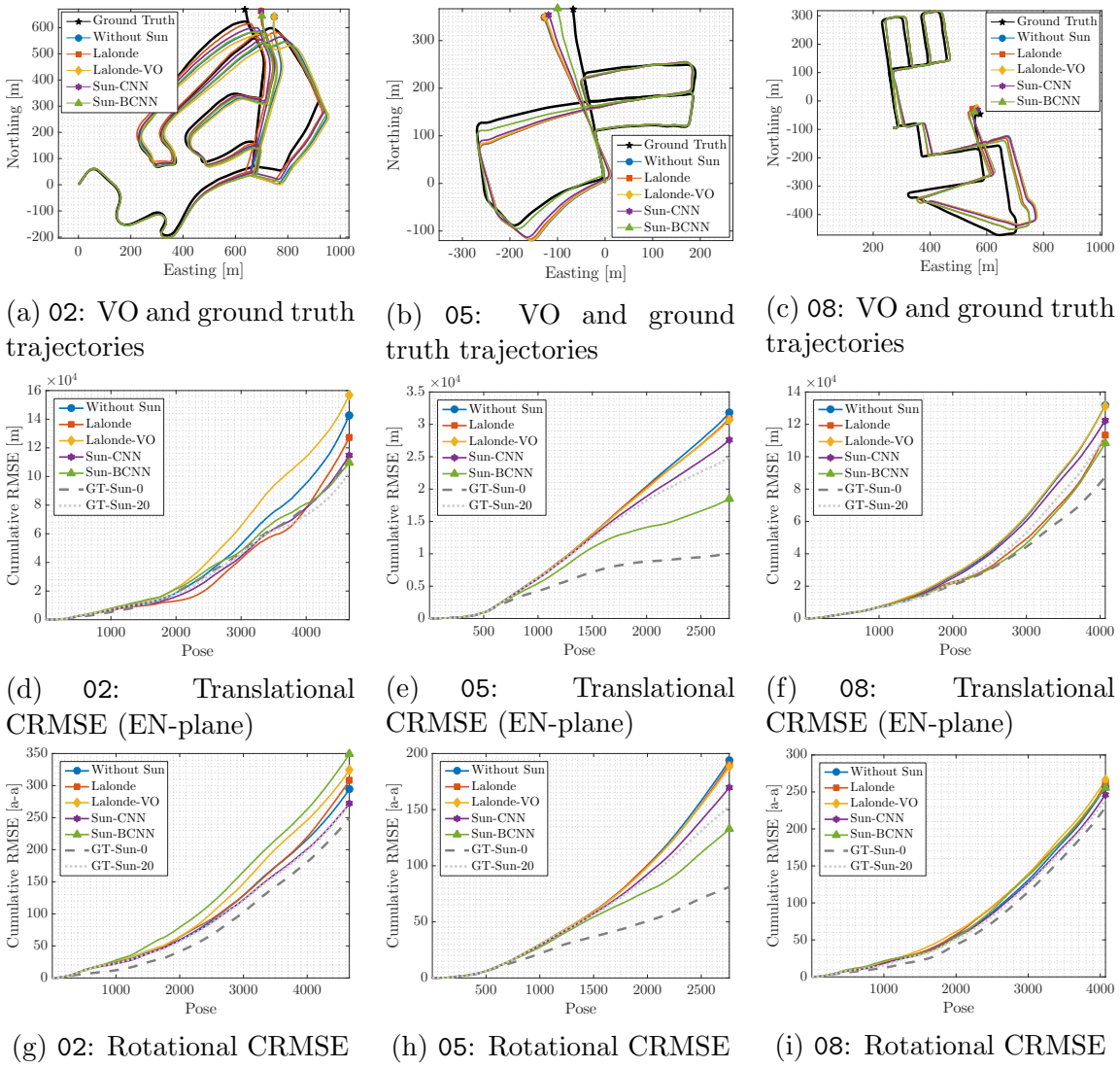
Figure 4.10: VO results for KITTI odometry sequence `05` using simulated sun measurements at every tenth pose. We observe a clear progression in cumulative root mean squared error (CRMSE) in translation and rotation as noise in the simulated sun measurements increases.

4.5 Urban Driving Experiments: The KITTI Odometry Benchmark

We investigated the performance of Sun-BCNN on the KITTI odometry benchmark training set (**?**Geiger et al., 2013b), which consists of 21.6 km of urban driving data[3]. 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 4.11: 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.

[3]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 4.12: 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-VO (**?**), 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 4.2: 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.

4.5.1 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 (4.4) with $N = 25$ and then re-normalized the resulting vector to preserve unit length. To obtain the required covariance on azimuth and zenith angles, we sampled the vector outputs, converted them to azimuth and zenith angles using **??**, and then applied Equation (4.5). We investigate the impact of this parametrization (as opposed to working in azimuth and zenith coordinates directly) later in this paper. As shown in Table 4.1, 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).

Figure 4.9 and **??** plot the error distributions for azimuth, zenith, and angular distance for all ten KITTI odometry sequences, while **??** 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.

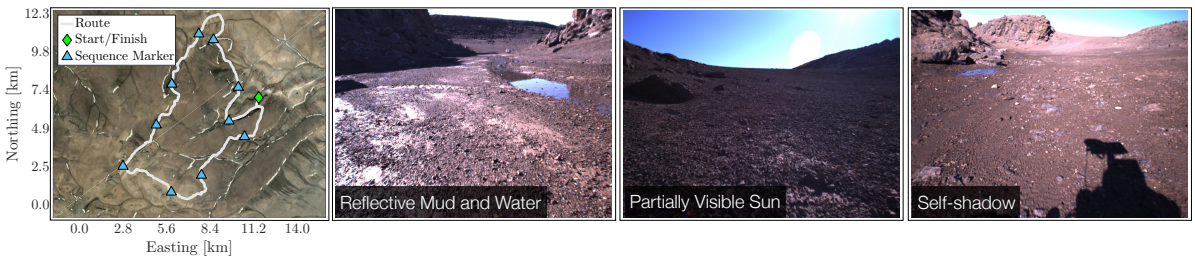


Figure 4.13: GPS track and sample images from the Devon Island traverse, with the start of each sequence highlighted. The Devon Island dataset is conducive to visual sun sensing due to the presence of strong environmental shadows, reflective surfaces such as mud and water, occasionally visible sun, and self-shadowing by the sensor platform. (Map data: Google, DigitalGlobe)

Table 4.1 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.

4.5.2 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 **?** and its VO-informed variant (**?**), 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 4.10 shows the results we obtained using simulated sun measurements on sequence 05, in which the basic VO suffers from substantial orientation drift.[4] 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 **??**, 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, **??** shows that rotational ARMSE nevertheless improves on all ten sequences when low-noise simulated sun measurements are included. This implies that

[4]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.

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 4.11 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 4.12 shows the results we obtained for sequences 02, 05, and 08 using the Sun-CNN of 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 **?** in its original and VO-informed (**?**) 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 **?**.

?? 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.

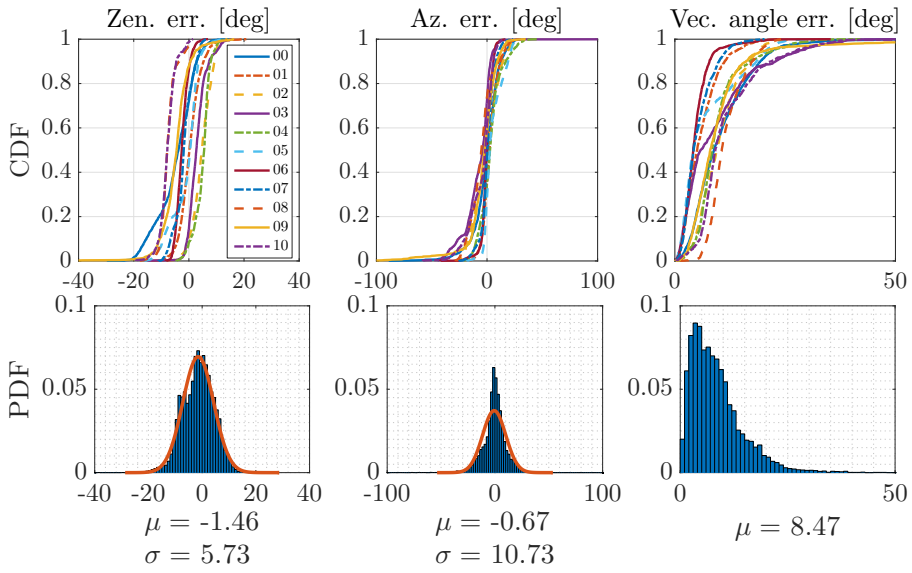


Figure 4.14: (Devon Island) Distributions of azimuth error, zenith error, and angular distance for Sun-BCNN compared to ground truth over each test sequence. *Top row*: Cumulative distributions of errors for each test sequence individually. *Bottom row:* Histograms and Gaussian fits of aggregated errors.

4.6 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 (**?**), which consists of various sensor data collected using a mobile sensor platform traversing a 10 km loop on Devon Island in the Canadian High Arctic (Figure 4.13). The rugged landscape of Devon Island (Figure 4.13) 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 **?** 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 4.13, 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

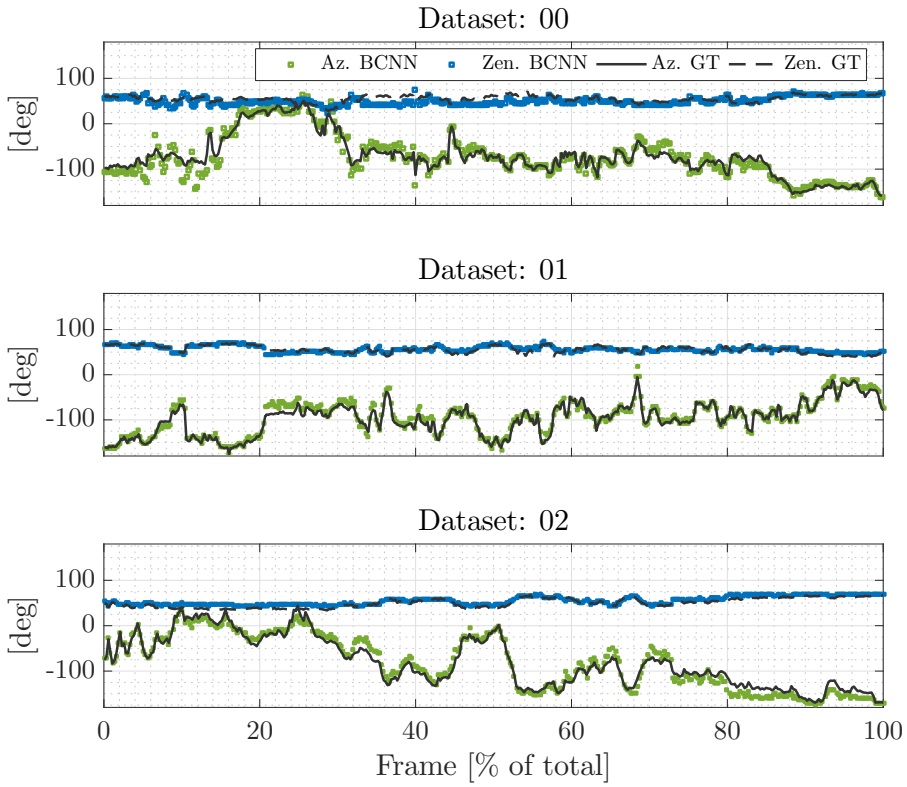


Figure 4.15: Azimuth (Sun-BCNN azimuth and zenith predictions over time for Devon Island test sequences `00`, `01` and `12`. Sun-BCNN is trained and tested on all frames (in our VO experiments, we use the Sun-BCNN predictions of every tenth image to make a fair comparison).

Table 4.3: 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 $\tau^{-1} = 0.01$.

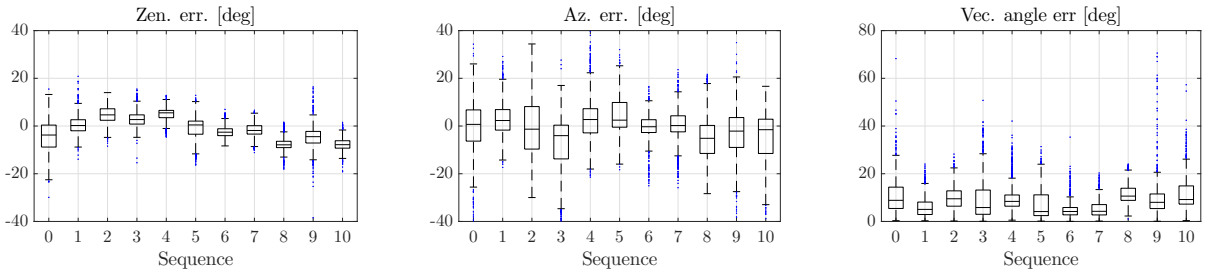


Figure 4.16: Box-and-whiskers plot of final test errors on Devon Island odometry sequences (c.f. **??**).

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 4.13). 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.

4.6.1 Sun-BCNN Test Results

As in our experiments with the KITTI odometry benchmark, we obtained the mean estimated sun vector by evaluating Equation (4.4) 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 **??**, and then applied Equation (4.5). As shown in **??**, 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).

???? plot the error distributions for azimuth, zenith, and angular distance for all 11 Devon Island odometry sequences, while **??** 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. **??**), 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.

?? summarizes the test errors and ANEES of each sequence numerically, while **????** plot the error distributions for azimuth, zenith, and angular distance for each sequence. **??** 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.

4.6.2 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 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 (**?**) and Lalonde-VO

(**?**) methods.

Table 4.4: 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.

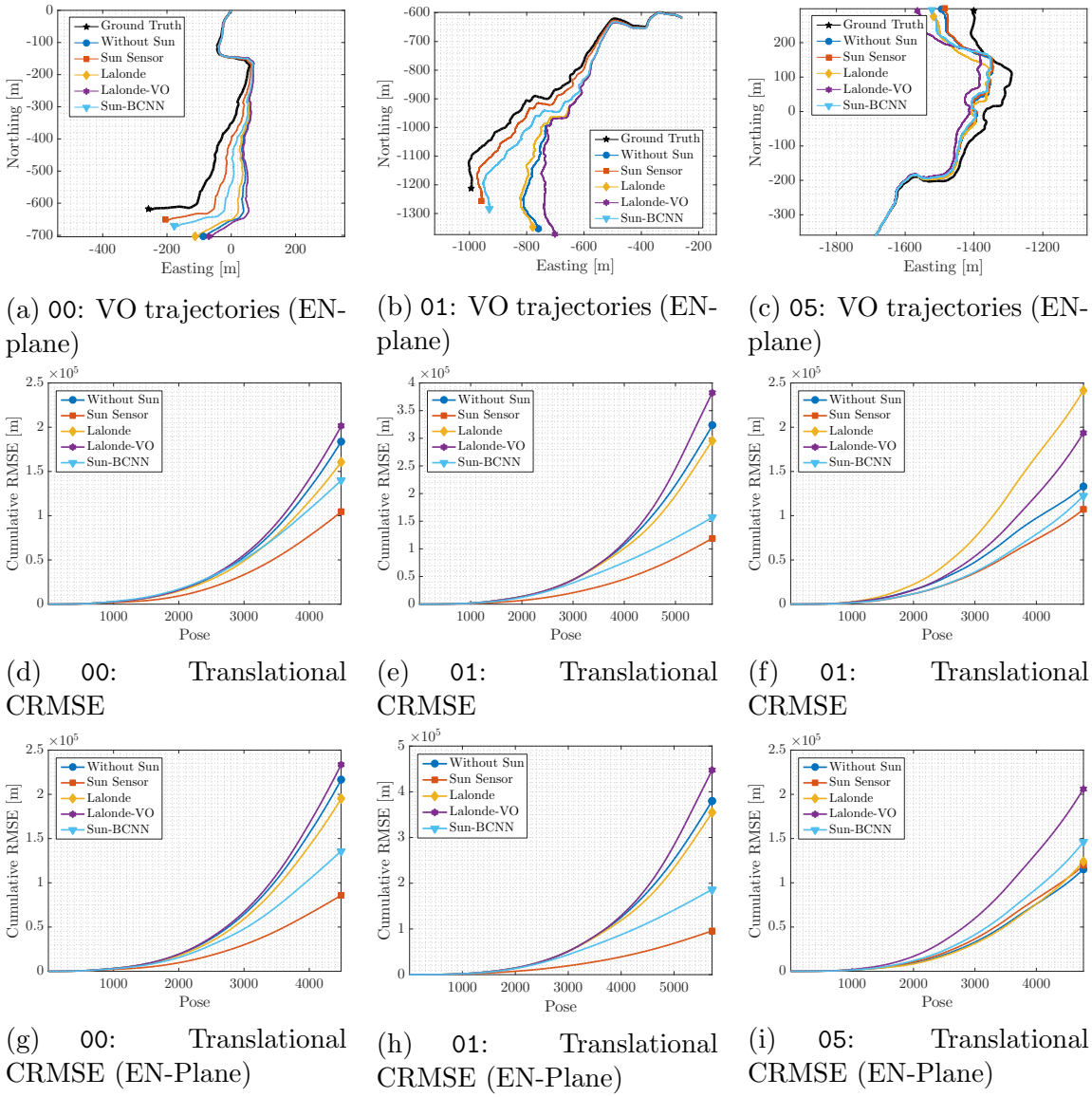
Sequence	00	01	02	03	04	05	06	07	08	09	10
Length [km]	0.9	1.1	1.0	1.0	0.9	1.0	1.1	1.0	0.9	0.7	0.6
Trans. ARMSE [m]											
Without Sun	40.93	56.51	41.58	42.04	30.52	27.82	58.91	40.04	47.22	11.39	12.94
Hardware Sun Sensor	23.26	20.79	9.79	22.03	30.79	22.47	24.14	29.59	47.97	6.26	8.50
Lalonde	35.77	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**

?? 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.

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

4.7 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.



(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 4.17: 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. **??**).

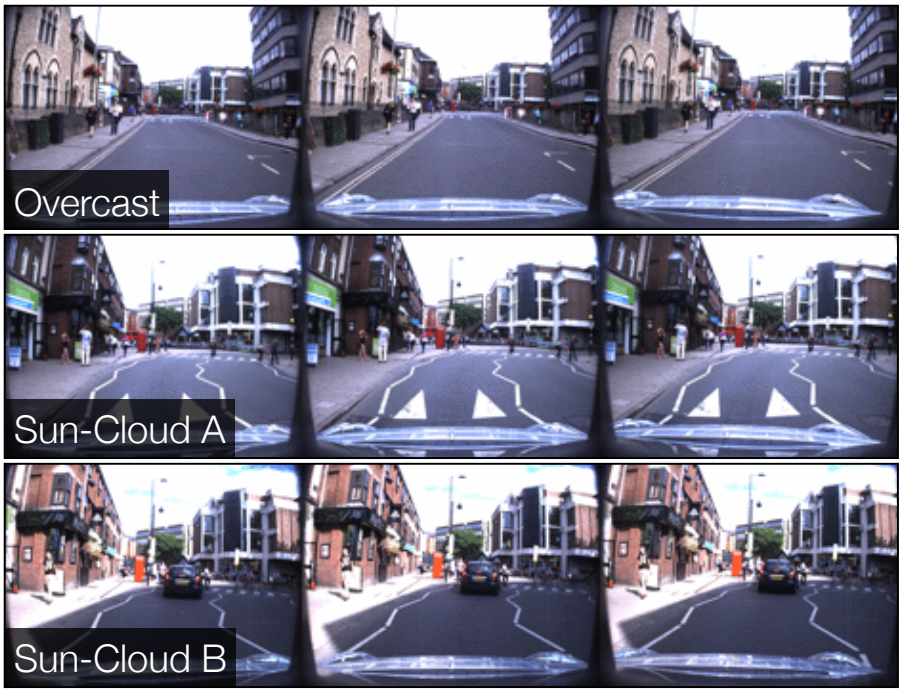


Figure 4.18: Sample images of approximately the same location taken from three different Oxford Robotcar sequences we used to investigate the effect of cloud cover on Sun-BCNN.

4.7.1 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 **??**, 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 (**?**), 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.

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 4.18 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%),

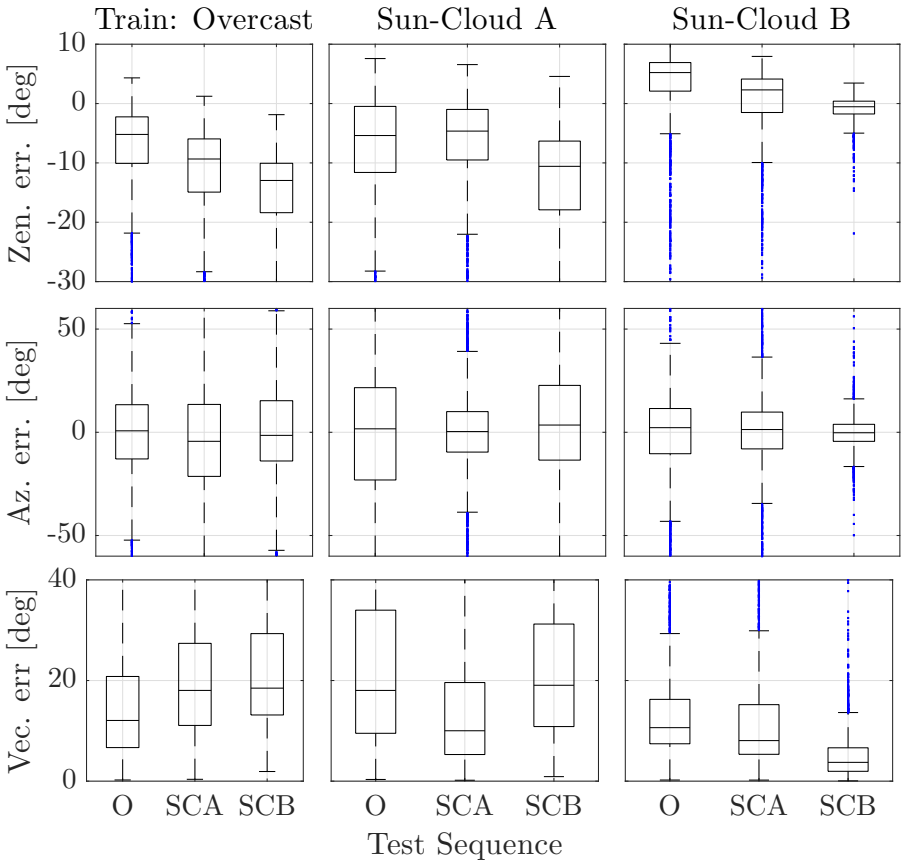


Figure 4.19: Box-and-whiskers plot for zenith, azimuth and vector angle errors for nine different combinations of train-test sequences taken from the Oxford Robotcar dataset. Each column corresponds to a different training sequence, and each plot contains three different test sequences. In the bottom legend, we use the labels O: *Overcast*, SCA: *Sun-Cloud A*, SCB: *Sun-Cloud B*.

validation (10%) and test (10%) images, and then train and test Sun-BCNN on each of the nine train-test permutations.

Results

Figure 4.19 shows the results of these experiments with box and whisker plots for azimuth, zenith and vector angle errors while **??** 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.

4.7.2 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 **?**) is more appropriate for the task of illumination estimation.

Procedure

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

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

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

Results

?? shows the results of these experiments with box and whisker plots for azimuth, zenith and vector angle errors while **??** 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).

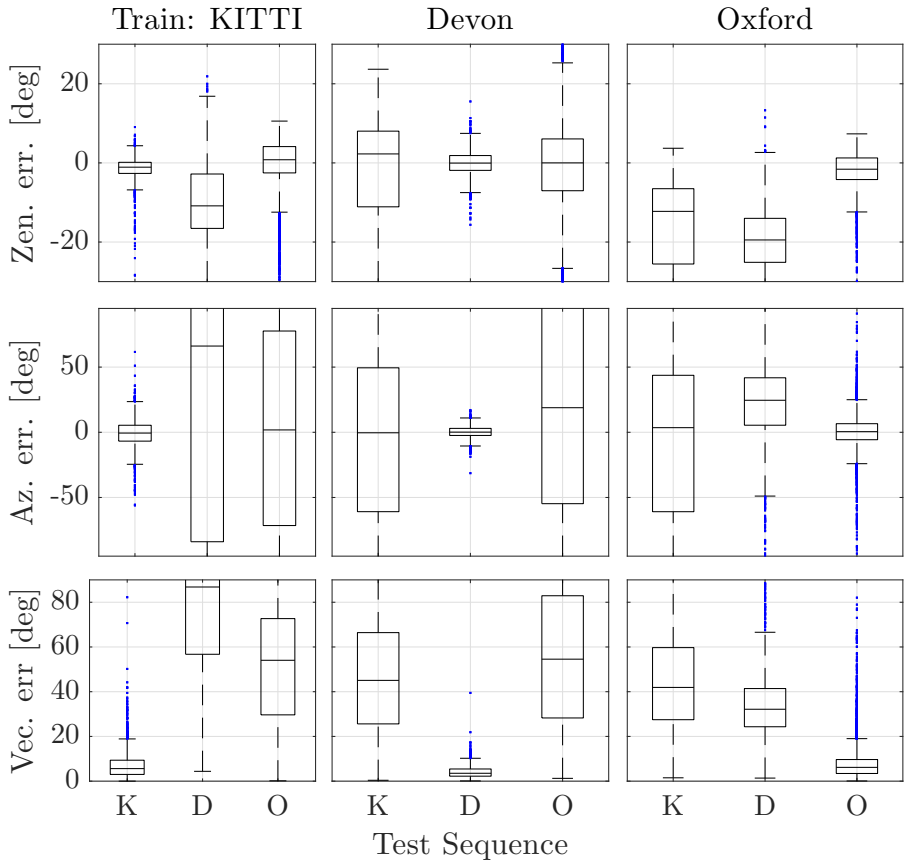


Figure 4.20: Box-and-whiskers plot for zenith, azimuth and vector angle errors for nine different combinations of train-test datasets. Each column corresponds to a different training sequence, and each plot contains three different test sequences. In the bottom legend, we use the labels K: KITTI, D: 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.

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.

4.7.3 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 (4.4) and (4.5)

Table 4.5: 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 4.6: 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

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 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 (4.4) 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 **??**.

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 **??**. We then evaluate Equation (4.4) 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 4.7 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 (4.5) 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 **??**.

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 (4.5) on the angles themselves.

Table 4.7: 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

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 **??** 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.

Table 4.8: 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

Chapter 5

Deep Pose Corrections for Visual Localization

5.1 Introduction

Deep convolutional neural networks (CNNs) are at the core of many state-of-the-art classification and segmentation algorithms in computer vision LeCun et al. (2015b). 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. In this work, we focus on visual odometry (VO): the task of computing the egomotion of a camera through an unknown environment with no external positioning sources. 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. (2016a, 2015a)), much recent work Costante et al. (2016); Clark et al. (2017); Kendall and Cipolla (2017); Melekhov et al. (2017); Oliveira et al. (2017) has been devoted to completely replacing the estimator with a CNN-based system.

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 recent results Peretroukhin et al. (2017a) that demonstrate that CNNs can infer difficult-to-model geometric quantities (e.g., the direction of the sun) to improve an

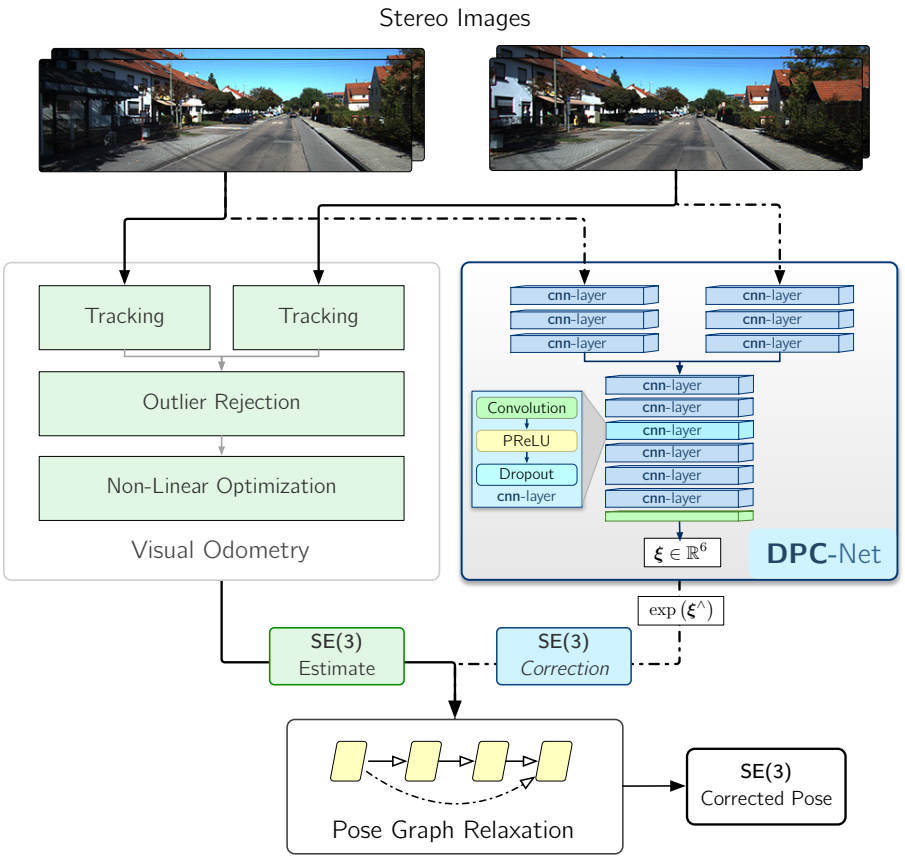


Figure 5.1: We propose a Deep Pose Correction network (DPC-Net) that learns SE(3) *corrections* to classical visual localizers.

existing localization estimate. In a similar vein, we propose a system (Figure 5.1) 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 miscalibration or environmental effects.

Although in this work we focus on visual data, the DPC-Net architecture can be readily modified to learn 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 SE(3) geodesic distance.

In summary, the main contributions of this work are

1. the formulation of a novel deep corrective approach to egomotion estimation,

2. a novel cost function for deep SE(3) regression that naturally balances translation and rotation errors, and

3. an open-source implementation of DPC-Net in `PyTorch`[1].

5.2 Related Work

Visual state estimation has a rich history in mobile robotics. We direct the reader to Scaramuzza and Fraundorfer (2011b) and Cadena et al. (2016) for detailed surveys of what we call *classical*, geometric approaches to visual odometry (VO) and visual simultaneous localization and mapping (SLAM).

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. (2015b). 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,

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

most successful applications of deep networks have aimed at replacing a specific subsystem 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, that these deep CNN-based approaches do not yet report state-of-the-art localization accuracy, focusing instead on proof-of-concept validation. Indeed, at the time of writing, the most accurate visual odometry approach on the KITTI odometry benchmark leaderboard[2] remains a variant of a sparse feature-based pipeline with carefully hand-tuned optimization Cvišić and Petrović (2015).

Taking inspiration from recent results that show that CNNs can be used to inject global orientation information into a visual localization pipeline Peretroukhin et al. (2017a), and leveraging ideas from recent approaches to trajectory tracking in the field of controls Li et al. (2017); 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*.

[2]See `http://www.cvlibs.net/datasets/kitti/eval_odometry.php`.

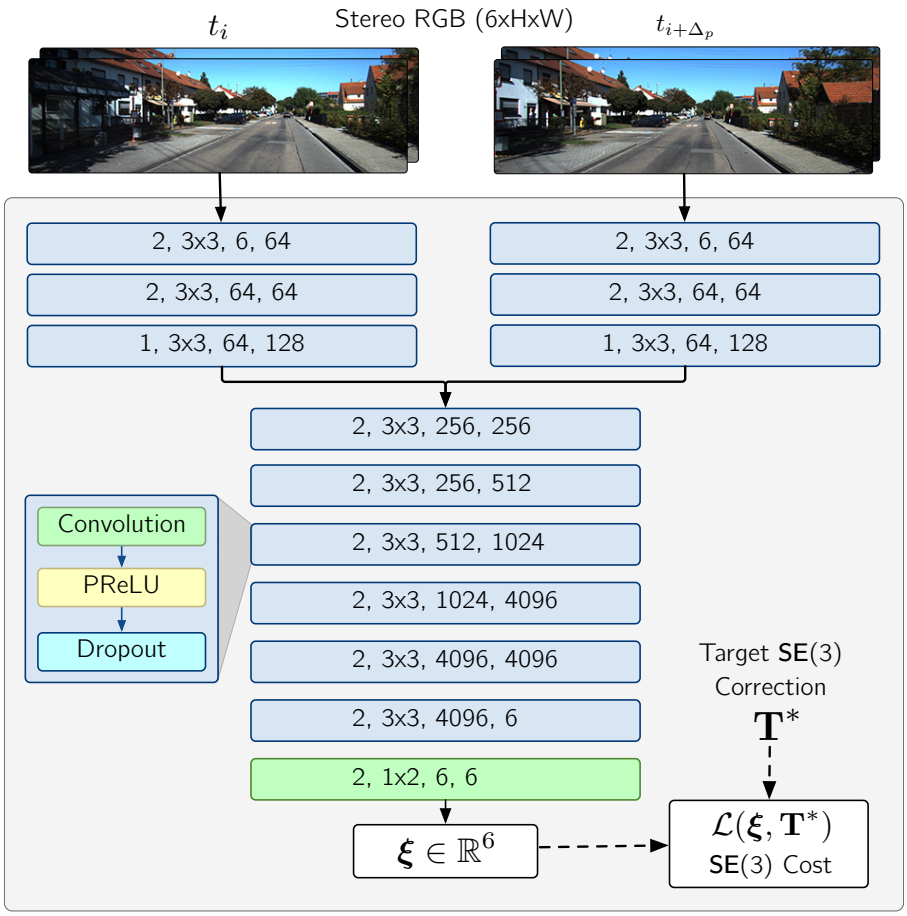


Figure 5.2: The Deep Pose Correction network with stereo RGB image data. The network learns a map from two stereo pairs to a vector of Lie algebra coordinates. Each darker blue block consists of a convolution, a PReLU non-linearity, and a dropout layer. We opt to not use MaxPooling in the network, following Handa et al. (2016). The labels correspond to the stride, kernel size, and number of input and output channels of for each convolution.

5.3 System Overview: Deep Pose Correction

We base our network structure on that of Handa et al. (2016) but learn 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 5.2 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 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 $\underrightarrow{\boldsymbol{\mathcal{F}}}_i$, $\underrightarrow{\boldsymbol{\mathcal{F}}}_{i+\Delta_p}$ 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[3], $\hat{\mathbf{T}}_{i,i+\Delta_p}$, 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,

$$\mathbf{T}^*_i = \mathbf{T}_{i,i+\Delta_p}\hat{\mathbf{T}}^{-1}_{i,i+\Delta_p}, \tag{5.1}$$

from two pairs of stereo images (collectively referred to as $\mathbf{I}_{t_i,t_{i+\Delta_p}}$) captured at t_i and $t_{i+\Delta_p}$[4]. 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 SE(3) corrections.

5.3.1 Loss Function: Correcting SE(3) Estimates

One 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 and Cipolla (2017); Melekhov et al. (2017); Oliveira et al. (2017)). This however does not account for the possible correlation between the two losses, and requires the careful tuning of a scalar weight.

In this work, we instead choose to parametrize our correction prediction as $\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge)$, where $\boldsymbol{\xi} \in \mathbb{R}^6$, a vector of Lie algebra coordinates, is the output of our network

[3]We use $\hat{(\cdot)}$ to denote an estimated quantity throughout the paper.

[4]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.

(similar to Handa et al. (2016)). We define a loss for $\boldsymbol{\xi}$ as

$$\mathcal{L}(\boldsymbol{\xi}) = \frac{1}{2} g(\boldsymbol{\xi})^T \boldsymbol{\Sigma}^{-1} g(\boldsymbol{\xi}), \tag{5.2}$$

where

$$g(\boldsymbol{\xi}) \triangleq \log\left(\exp\left(\boldsymbol{\xi}^\wedge\right) \mathbf{T}^{*-1}\right)^\vee. \tag{5.3}$$

Here, $\boldsymbol{\Sigma}$ is the covariance of our estimator (expressed using unconstrained Lie algebra coordinates), and $(\cdot)^\wedge$, $(\cdot)^\vee$ convert vectors of Lie algebra coordinates to matrix vectorspace quantities and back, respectively. Given two stereo image pairs, we use the output of DPC-Net, $\boldsymbol{\xi}$, to correct our estimator as follows:

$$\hat{\mathbf{T}}^{\text{corr}} = \exp\left(\boldsymbol{\xi}^\wedge\right) \hat{\mathbf{T}}, \tag{5.4}$$

where we have dropped subscripts for clarity.

5.3.2 Loss Function: SE(3) Covariance

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

$$\boldsymbol{\Sigma} = \frac{1}{N-1} \sum_{i=1}^{N} \left(\boldsymbol{\xi}_i^* - \overline{\boldsymbol{\xi}^*}\right) \left(\boldsymbol{\xi}_i^* - \overline{\boldsymbol{\xi}^*}\right)^T, \tag{5.5}$$

where

$$\boldsymbol{\xi}_i^* \triangleq \log\left(\mathbf{T}_i^*\right)^\vee, \quad \overline{\boldsymbol{\xi}^*} \triangleq \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\xi}_i^*. \tag{5.6}$$

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 Kendall and Cipolla (2017) based on the diagonal components, but contains relatively large off-diagonal terms.

5.3.3 Loss Function: SE(3) Jacobians

In order to use Equation (5.2) 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

$$\frac{\partial \mathcal{L}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = g(\boldsymbol{\xi})^T \boldsymbol{\Sigma}^{-1} \frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}. \tag{5.7}$$

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

$$\exp\left((\boldsymbol{\xi} + \delta\boldsymbol{\xi})^\wedge\right) \approx \exp\left((\boldsymbol{\mathcal{J}}\delta\boldsymbol{\xi})^\wedge\right) \exp\left(\boldsymbol{\xi}^\wedge\right), \tag{5.8}$$

where $\boldsymbol{\mathcal{J}} \triangleq \boldsymbol{\mathcal{J}}(\boldsymbol{\xi})$ 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

$$\begin{aligned} \log\left(\mathbf{T}_1\mathbf{T}_2\right)^\vee &= \log\left(\exp\left({\boldsymbol{\xi}_1}^\wedge\right)\exp\left({\boldsymbol{\xi}_2}^\wedge\right)\right)^\vee \\ &\approx \left\{ \begin{array}{ll} \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_2)^{-1}\boldsymbol{\xi}_1 + \boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_1 \text{ small} \\ \boldsymbol{\xi}_1 + \boldsymbol{\mathcal{J}}(-\boldsymbol{\xi}_1)^{-1}\boldsymbol{\xi}_2 & \text{if } \boldsymbol{\xi}_2 \text{ small.} \end{array} \right\} \end{aligned} \tag{5.9}$$

See Barfoot (2017) for a detailed treatment of matrix Lie groups and their use in state estimation.

Deriving $\frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$, Method I

If we assume that only $\boldsymbol{\xi}$ is 'small', we can apply Equation (5.9) directly to define

$$\frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \boldsymbol{\mathcal{J}}(-\boldsymbol{\xi}^*)^{-1}, \tag{5.10}$$

with $\boldsymbol{\xi}^* \triangleq \log\left(\mathbf{T}^*\right)^\vee$. Although attractively compact, note that this expression for $\frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$ 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').

Deriving $\frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$, Method II

Alternatively, we can linearize Equation (5.3) about $\boldsymbol{\xi}$, by considering a small change $\delta\boldsymbol{\xi}$ and applying Equation (5.8):

$$g(\boldsymbol{\xi} + \delta\boldsymbol{\xi}) = \log\left(\exp\left((\boldsymbol{\xi} + \delta\boldsymbol{\xi})^\wedge\right)\mathbf{T}^{*-1}\right)^\vee \tag{5.11}$$

$$\approx \log\left(\exp\left((\boldsymbol{\mathcal{J}}\delta\boldsymbol{\xi})^\wedge\right)\exp\left(\boldsymbol{\xi}^\wedge\right)\mathbf{T}^{*-1}\right)^\vee. \tag{5.12}$$

Now, assuming that $\boldsymbol{\mathcal{J}}\delta\boldsymbol{\xi}$ is 'small', and using Equation (5.3), Equation (5.9) gives:

$$g(\boldsymbol{\xi}+\delta\boldsymbol{\xi}) \approx \boldsymbol{\mathcal{J}}(g(\boldsymbol{\xi}))^{-1}\boldsymbol{\mathcal{J}}(\boldsymbol{\xi})\delta\boldsymbol{\xi} + g(\boldsymbol{\xi}). \tag{5.13}$$

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

$$\frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \boldsymbol{\mathcal{J}}(g(\boldsymbol{\xi}))^{-1}\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}). \tag{5.14}$$

Although slightly more computationally expensive, this expression makes no assumptions about the 'magnitude' of our correction and works reliably for any target. Note further that if $\boldsymbol{\xi}$ is small, then $\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}) \approx \mathbf{1}$ and $\exp\left(\boldsymbol{\xi}^{\wedge}\right) \approx \mathbf{1}$. Thus,

$$g(\boldsymbol{\xi}) \approx \log\left(\mathbf{T}^{*-1}\right)^{\vee} = -\boldsymbol{\xi}^{*}, \tag{5.15}$$

and Equation (5.14) becomes

$$\frac{\partial g(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = \boldsymbol{\mathcal{J}}(-\boldsymbol{\xi}^{*})^{-1}, \tag{5.16}$$

which matches *Method I*. To summarize, to apply back-propagation to Equation (5.2), we use Equation (5.7) and Equation (5.14).

5.3.4 Loss Function: Correcting SO(3) Estimates

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

$$\mathcal{L}(\boldsymbol{\phi}, \mathbf{C}_{*}) = \frac{1}{2} f(\boldsymbol{\phi})^{T} \boldsymbol{\Sigma}^{-1} f(\boldsymbol{\phi}), \tag{5.17}$$

where

$$f(\boldsymbol{\phi}) \triangleq \log\left(\exp\left(\boldsymbol{\phi}^{\wedge}\right) \mathbf{C}_{*}{}^{-1}\right)^{\vee}. \tag{5.18}$$

Equations (5.8) and (5.9) have analogous SO(3) formulations:

$$\exp\left((\boldsymbol{\phi}+\delta\boldsymbol{\phi})^{\wedge}\right) \approx \exp\left((\mathbf{J}\delta\boldsymbol{\phi})^{\wedge}\right)\exp\left(\boldsymbol{\phi}^{\wedge}\right), \tag{5.19}$$

and

$$\begin{aligned}\log\left(\mathbf{C}_1\mathbf{C}_2\right)^{\vee} &= \log\left(\exp\left(\boldsymbol{\phi}_1{}^{\wedge}\right)\exp\left(\boldsymbol{\phi}_2{}^{\wedge}\right)\right)^{\vee} \\ &\approx \begin{cases} \mathbf{J}(\boldsymbol{\phi}_2)^{-1}\boldsymbol{\phi}_1 + \boldsymbol{\phi}_2 & \text{if } \boldsymbol{\phi}_1 \text{ small} \\ \boldsymbol{\phi}_1 + \mathbf{J}(-\boldsymbol{\phi}_1)^{-1}\boldsymbol{\phi}_2 & \text{if } \boldsymbol{\phi}_2 \text{ small,} \end{cases}\end{aligned} \tag{5.20}$$

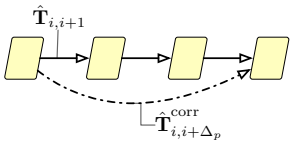


Figure 5.3: We apply pose graph relaxation to fuse high-rate visual localization estimates ($\hat{\mathbf{T}}_{i,i+1}$) with low-rate deep pose corrections ($\hat{\mathbf{T}}^{\text{corr}}_{i,i+\Delta_p}$).

where $\mathbf{J} \triangleq \mathbf{J}(\boldsymbol{\phi})$ is the left SO(3) Jacobian. Accordingly, the final loss Jacobians are identical in structure to Equation (5.7) and Equation (5.14), with the necessary SO(3) replacements.

5.3.5 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 5.3), we solve the non linear minimization problem

$$\{\mathbf{T}_{t_i,n}\}_{i=0}^{\Delta_p} = \underset{\{\mathbf{T}_{t_i,n}\}_{i=0}^{\Delta_p} \in \text{SE(3)}}{\text{argmin}} \quad \mathcal{O}_t, \tag{5.21}$$

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:

$$\mathcal{O}_t = \mathcal{O}_v + \mathcal{O}_c. \tag{5.22}$$

The former cost sums over each estimated transform,

$$\mathcal{O}_v \triangleq \sum_{i=0}^{\Delta_p - 1} \mathbf{e}_{t_i,t_{i+1}}^T \boldsymbol{\Sigma}_v^{-1} \mathbf{e}_{t_i,t_{i+1}}, \tag{5.23}$$

while the latter incorporates a single pose correction,

$$\mathcal{O}_c \triangleq \mathbf{e}_{t_0,t_{\Delta_p}}^T \boldsymbol{\Sigma}_c^{-1} \mathbf{e}_{t_0,t_{\Delta_p}}, \tag{5.24}$$

with the pose error defined as

$$\mathbf{e}_{1,2} = \log\left(\hat{\mathbf{T}}_{1,2}\mathbf{T}_2\mathbf{T}_1^{-1}\right)^{\vee}. \tag{5.25}$$

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

5.4 Experiments

To assess the power of our proposed deep corrective paradigm, we trained DPC-Net on visual data with localization estimates from a sparse stereo visual odometry (S-VO) estimator. In addition to training the full SE(3) DPC-Net, we modified the loss function and input data to learn simpler 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.

5.4.1 Training & Testing

For all experiments, we used the KITTI odometry benchmark training set Geiger et al. (2013b). 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[5]. We resized all images to $[400, 120]$ pixels, approximately preserving their original aspect ratio[6]. For non-distorted data, we use $\Delta_p \in [3, 4, 5]$ 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[7] 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.

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

[6]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.

[7]If a sequence has M poses, we collect $M - \Delta_p$ training samples for each Δ_p.

Rotation

To train rotation-only corrections, we extracted the SO(3) component of $\mathbf{T}_i^*$ and trained our network using Equation (5.17). 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[8].

Yaw

To further simplify the corrections, we extracted a single-degree-of-freedom yaw rotation correction angle[9] from $\mathbf{T}_i^*$, and trained DPC-Net with monocular images and a mean squared loss.

5.4.2 Estimators

Sparse Visual Odometry

To collect $\mathbf{T}_i^*$, we first used a frame-to-frame sparse visual odometry pipeline similar to that presented in Peretroukhin et al. (2016a). We briefly outline the pipeline here.

Using the open-source `libviso2` package Geiger et al. (2011), we detect and track sparse stereo image key-points, $\mathbf{y}_{l,t_{i+1}}$ and $\mathbf{y}_{l,t_i}$, 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, $\mathbf{\Sigma}_\mathbf{y}$,

$$\mathbf{e}_{l,t_i} = \mathbf{y}_{l,t_{i+1}} - f(\mathbf{T}_{t_{i+1},t_i} f^{-1}(\mathbf{y}_{l,t_i})) \tag{5.26}$$

$$\sim \mathcal{N}\mathbf{0}\mathbf{\Sigma}_\mathbf{y}, \tag{5.27}$$

where $f(\cdot)$ is the stereo camera projection function. 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, $\mathbf{T}_{t+1,t}^*$, through a Gauss-Newton minimization:

$$\mathbf{T}_{t_{i+1},t_i}^* = \underset{\mathbf{T}_{t_{i+1},t_i} \in \mathrm{SE(3)}}{\operatorname{argmin}} \sum_{l=1}^{N_{t_i}} \mathbf{e}_{l,t_i}^T \, {\mathbf{\Sigma}_\mathbf{y}}^{-1} \mathbf{e}_{l,t_i}. \tag{5.28}$$

Figure 5.4: 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

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = \left(1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6\right) \begin{bmatrix} x_n \\ y_n \end{bmatrix}, \tag{5.29}$$

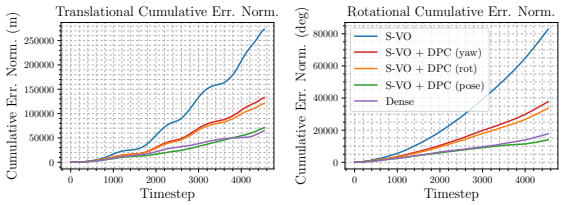
where $r = \sqrt{x_n^2 + y_n^2}$. We set the distortion coefficients, κ_1, κ_2, and κ_3 to $-0.3, 0.2, 0.01$ respectively, to approximately match the KITTI radial distortion parameters. We solved Equation (5.29) iteratively and used bilinear interpolation to compute the distorted images for every stereo pair in a sequence. Finally, we cropped each distorted image to remove any whitespace. Figure 5.4 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.

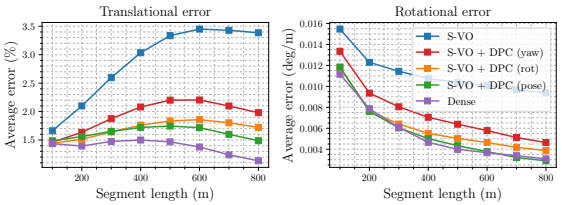
Dense Visual Odometry

Finally, we present localization estimates from a computationally-intensive keyframe-based dense, direct visual localization pipeline **?** that 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

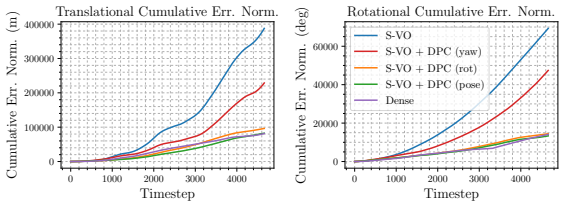
[8] The *stereo* VO estimator remained unchanged.
[9] We define yaw in the camera frame as the rotation about the camera's vertical y axis.



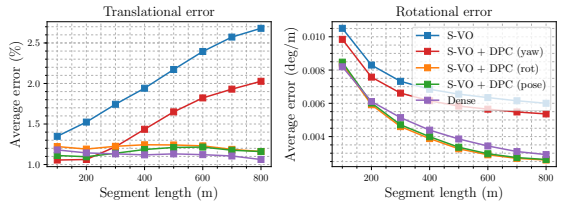
(a) Sequence 00: Cumulative Errors.



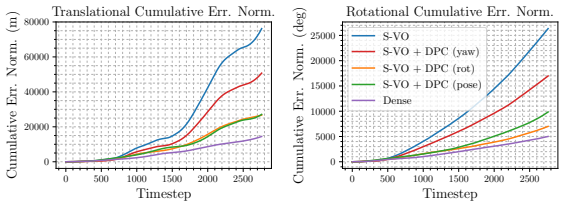
(b) Sequence 00: Segment Errors.



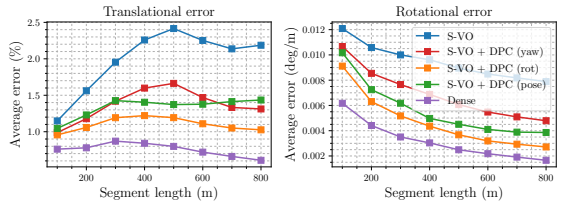
(c) Sequence 02: Cumulative Errors.



(d) Sequence 02: Segment Errors.



(e) Sequence 05: Cumulative Errors.



(f) Sequence 05: Segment Errors.

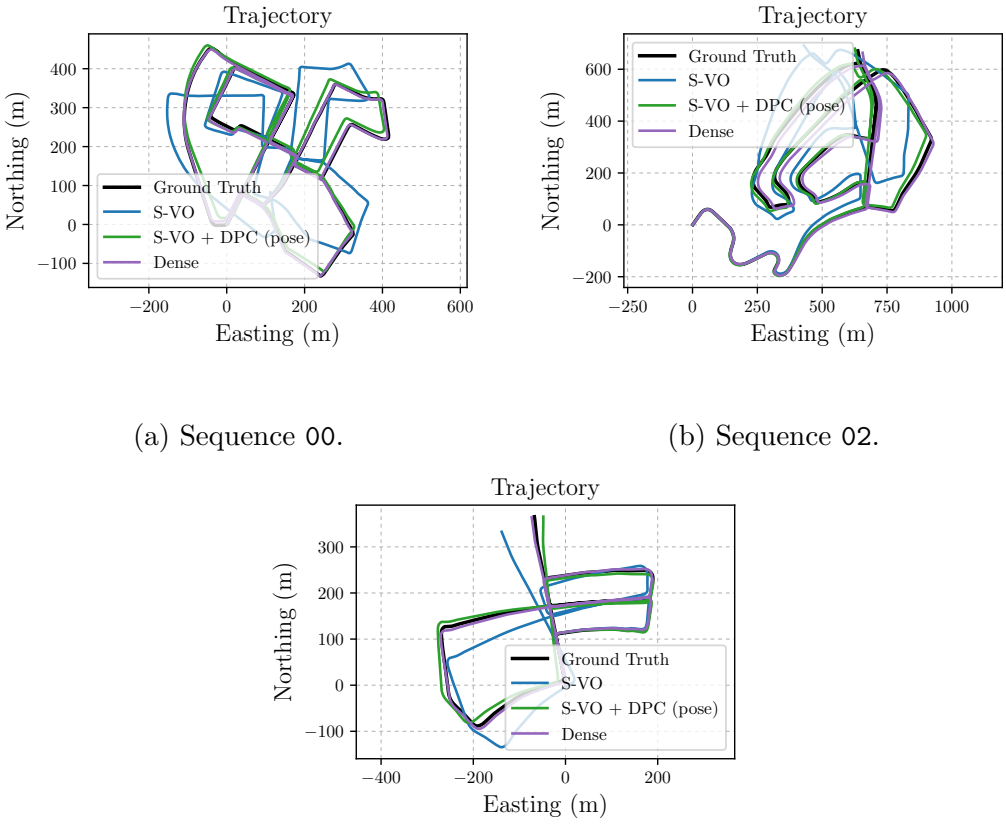
Figure 5.5: c-ATE and mean segment errors for S-VO with and without DPC-Net.

keyframe.

We used this dense, 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.

5.4.3 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.



(a) Sequence 00.

(b) Sequence 02.

(c) Sequence 05.

Figure 5.6: Top down projections for S-VO with and without DPC-Net.

Mean Absolute Trajectory Error (m-ATE)

The mean absolute trajectory error averages the magnitude of the rotational or translational error[10] 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

$$\mathbf{e}_{\text{m-ATE}} \triangleq \frac{1}{N}\sum_{p=1}^{N}\left\|\log\left(\hat{\mathbf{T}}_{p,0}^{-1}\mathbf{T}_{p,0}\right)^{\vee}\right\|. \tag{5.30}$$

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

$$\mathbf{e}_{\text{c-ATE}}(q) \triangleq \sum_{p=1}^{q}\left\|\log\left(\hat{\mathbf{T}}_{p,0}^{-1}\mathbf{T}_{p,0}\right)^{\vee}\right\|. \tag{5.31}$$

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.

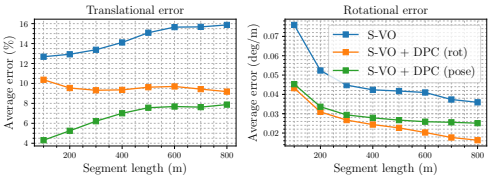
Segment Error

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

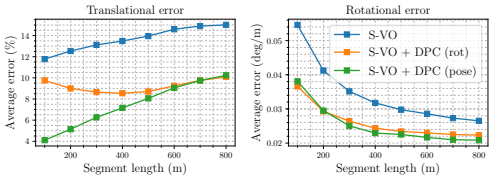
$$\mathbf{e}_{\text{seg}}(s) \triangleq \frac{1}{sN_s}\sum_{p=1}^{N_s}\left\|\log\left(\hat{\mathbf{T}}_{p+s_p,p}^{-1}\mathbf{T}_{p+s_p,p}\right)^{\vee}\right\|, \tag{5.32}$$

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).

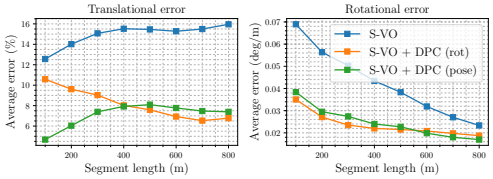
[10] For brevity, the notation $\log(\cdot)^{\vee}$ returns rotational or translational components depending on context.



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



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



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

Figure 5.7: c-ATE and segment errors for S-VO with radially distorted images with and without DPC-Net.

5.5 Results & Discussion

5.5.1 Correcting Sparse Visual Odometry

Figure 5.5 plots 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 5.1 summarize the results quantitatively, while Figure 5.6 plots the North-East projection of each trajectory. On average, DPC-Net trained with the full 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 SE(3) corrections performed slightly worse than their 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 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).

Table 5.1: 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.

5.5.2 Distorted Images

Figure 5.7 plots mean segment errors for the radially distorted dataset. On average, DPC-Net trained with the full 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 5.2). 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 SE(3) and 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.

Table 5.2: 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**

Chapter 6

Simultaneous Localization and Learning: Optimizing Deep SE(3) Measurement Models

6.1 Motivation

We present a novel way to use error-state Kalman filters to optimize deep probabilistic measurement models. Unlike traditional approaches that separate learning into two distinct training and evaluation phases, we instead use the modern tools of deep learning to embed a hyper-parametric model that can be optimized in tandem with trajectory estimates. Extending our prior work, we use matrix Lie groups to develop a parametrization with analytically differentiable loss functions and uncertainty propagation for 6 degree-of-freedom pose estimation. We show that by learning residuals to a predicted state, we can use the method of uncertainty injection onto the manifold to predict meaningful covariance estimates. As calibrated uncertainty estimates are crucial to reliable state estimation, we leverage ideas from the method of the statistical bootstrap to create a network we call a HydraNet that can produce covariance estimates that incorporate both epistemic and aleatoric uncertainty. Finally, we show how our system can be used to learn deep probabilistic measurement models in both two dimensions with planar Lidar data, and in three dimensions with stereo cameras.

6.2 Global Lie Algebra Coordinates

A basic composition in SE(3) is given by,

$$\mathbf{T}_c = \mathbf{T}_a \mathbf{T}_b. \tag{6.1}$$

We choose instead to use the composition in Lie algebra, or exponential coordinates, $\boldsymbol{\xi} = \mathrm{Log}\left(\mathbf{T}\right) = \log\left(\mathbf{T}\right)^{\vee}$ (and similarly, $\mathbf{T} = \mathrm{Exp}\left(\boldsymbol{\xi}\right) = \exp\left(\boldsymbol{\xi}^{\wedge}\right)$), such that,

$$\boldsymbol{\xi}_c = g(\boldsymbol{\xi}_a, \boldsymbol{\xi}_b) = \mathrm{Log}\left(\mathrm{Exp}\left(\boldsymbol{\xi}_a\right)\mathrm{Exp}\left(\boldsymbol{\xi}_b\right)\right). \tag{6.2}$$

This then allows us to derive the associated linear system in two ways.
Left perturbations:

$$\delta\boldsymbol{\xi}_c = \delta\boldsymbol{\xi}_a + \mathrm{Ad}\left(\mathbf{T}_a\right)\delta\boldsymbol{\xi}_b. \tag{6.3}$$

Middle perturbations:

$$\delta\boldsymbol{\xi}_c = \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_c)^{-1}\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_a)\delta\boldsymbol{\xi}_a + \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_c)^{-1}\mathrm{Ad}\left(\mathbf{T}_a\right)\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_b)\delta\boldsymbol{\xi}_b. \tag{6.4}$$

See the supplementary material for a derivation.
We can now define the loss function - negative log likelihood:

$$\mathcal{L}(\boldsymbol{\xi}, \boldsymbol{\xi}_t, \boldsymbol{\Sigma}) = \left\|\boldsymbol{\xi} \ominus \boldsymbol{\xi}_t\right\|_{\boldsymbol{\Sigma}} + \log|\boldsymbol{\Sigma}| \tag{6.5}$$

where $\boldsymbol{\xi}_a \ominus \boldsymbol{\xi}_b = g(\boldsymbol{\xi}_a, -\boldsymbol{\xi}_b)$

6.3 Experiments

6.4 Consistent Uncertainty Estimates

6.5 System

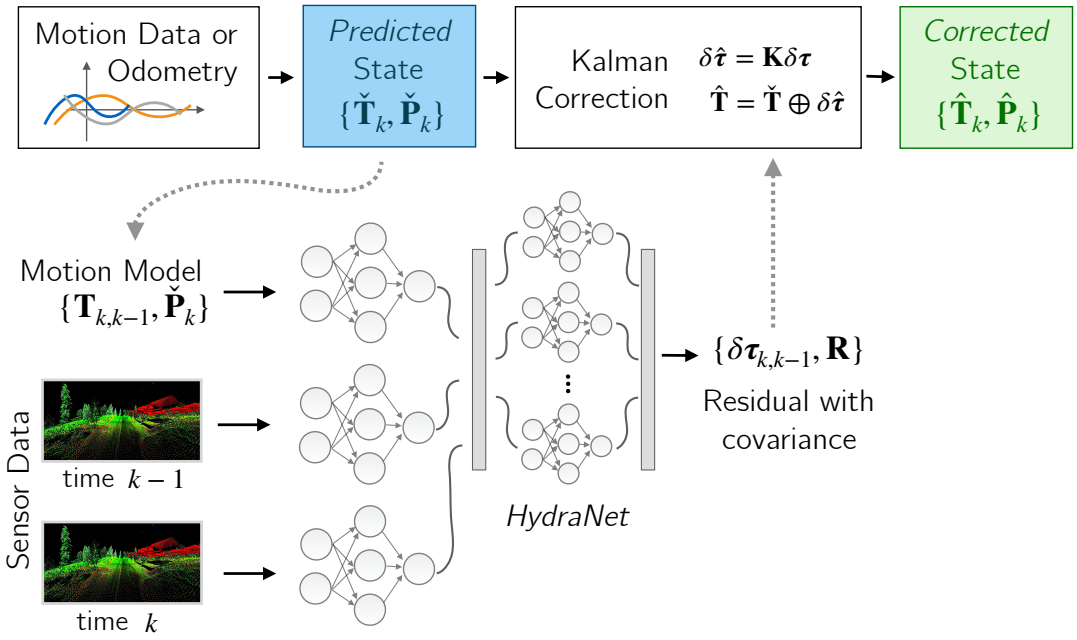
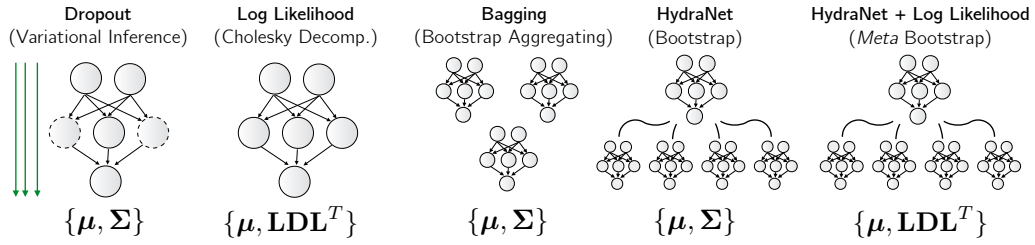
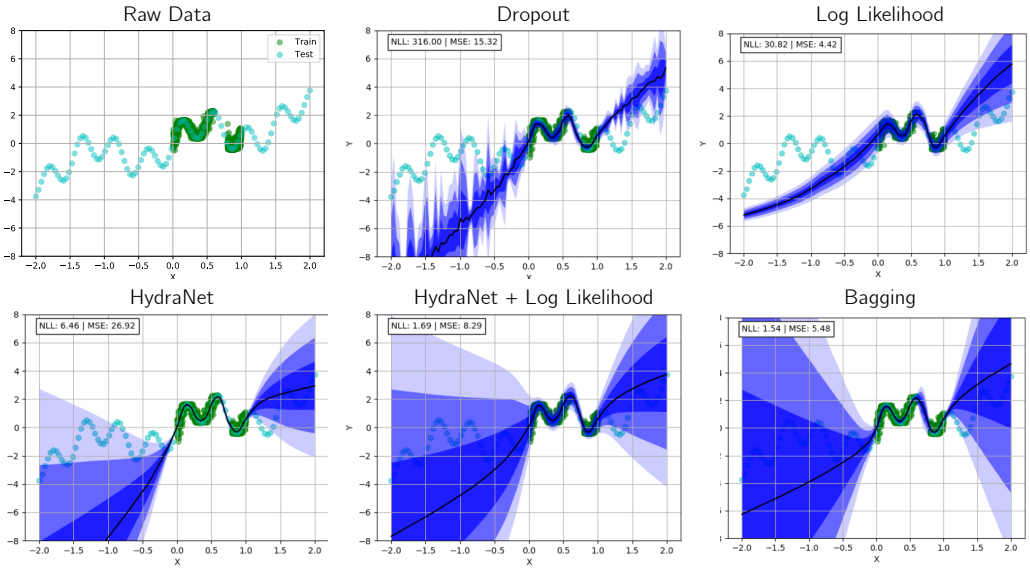


Figure 6.1: We present a novel way to train deep probabilistic measurement models with sparse ground truth data. Our architecture uses a multi-headed network we call a HydraNet to output residuals to a predicted state given sensor data and the state itself.

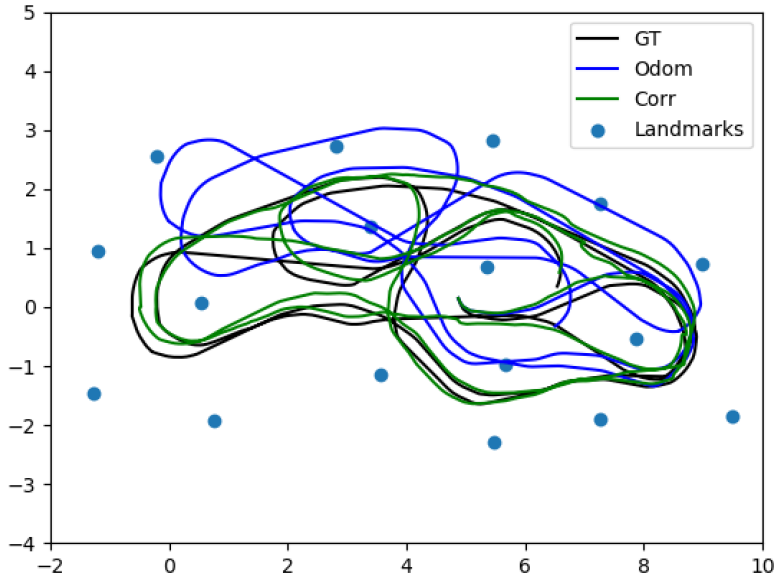


(a) Different proposed uncertainty schemes for deep networks.

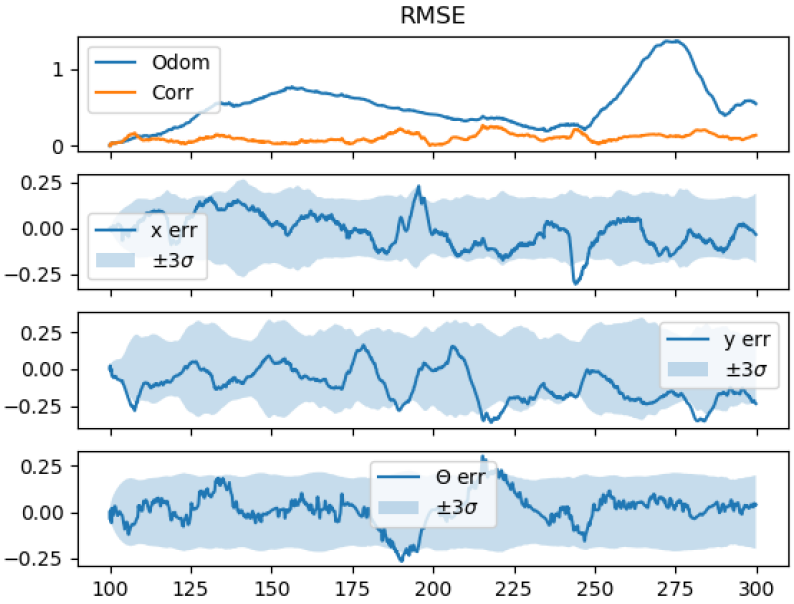


(b) Comparisons of uncertainty predictions from five different probabilistic neural networks. Shades of blue represent multiples of σ. Training and test data come from the generating function $y = x + \sin(\alpha x) + \sin(\beta x)$.

Figure 6.2: Investigations into uncertainty with deep neural networks. The network consists of two layers, with one dimensional inputs and outputs. Bagging uses 10 models and HydraNet uses 10 heads.



(a) Top down view on a 2D dataset with external groundtruth every 10 seconds.



(b) Errors on a 2D dataset with external groundtruth every 10 seconds.

Figure 6.3: 2D investigations.

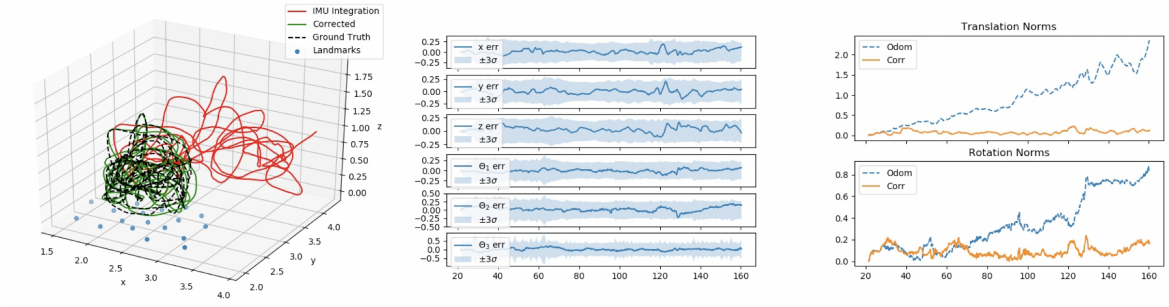


Figure 6.4: Errors on a 3D dataset with external groundtruth every 15 seconds.

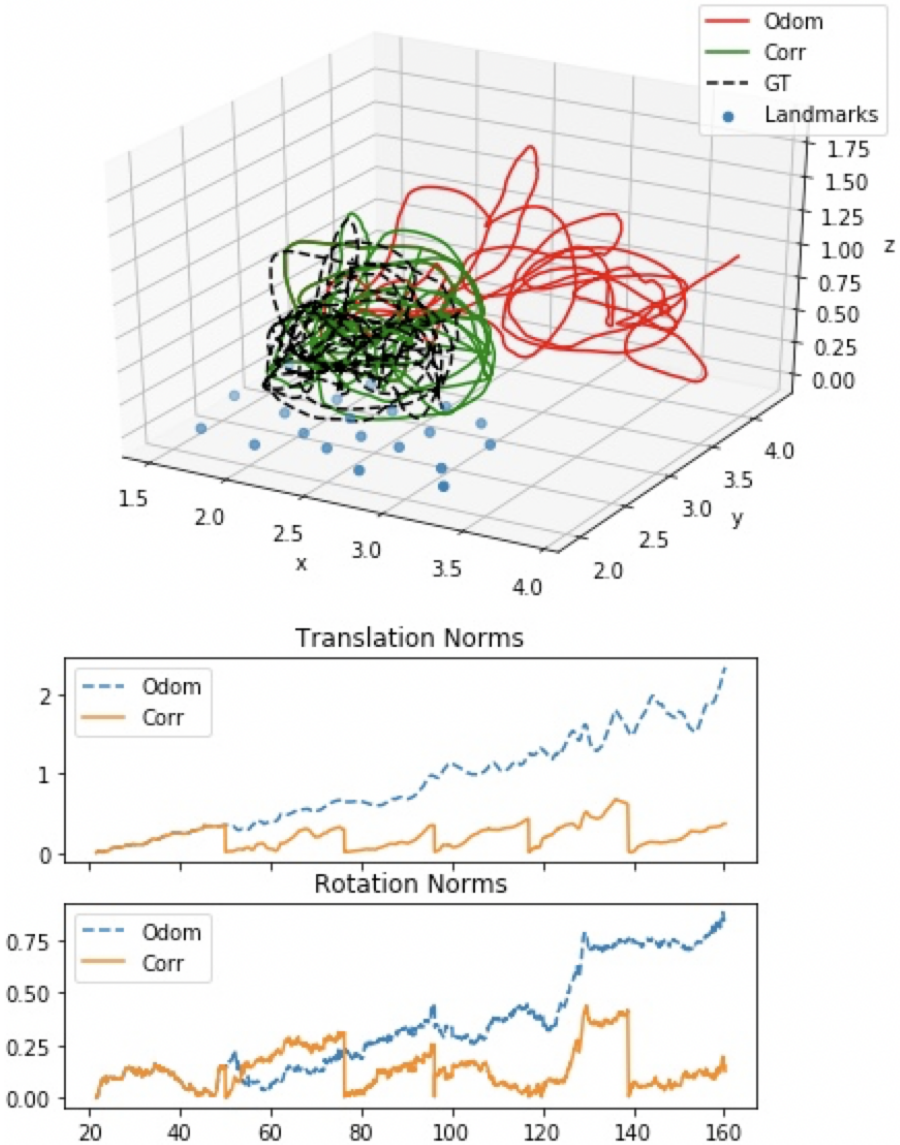


Figure 6.5: The same 3D data with a traditional Kalman filter with the same ground truth.

Chapter 7

Conclusion

In conclusion, I have presented several pieces of work aimed at using modern machine learning to improve visual odometry.

7.1 Summary of Publications

Deep Learned Models

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

Sun BCNN

- Peretroukhin, V., Clement, L., and Kelly, J. (2018). Inferring sun direction to improve visual odometry: A deep learning approach. *International Journal of Robotics Research*

- Peretroukhin, V., Clement, L., and Kelly, J. (2017b). 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

- 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

Predictive Robust Estimation

- Peretroukhin, V., Vega-Brown, W., Roy, N., and Kelly, J. (2016b). PROBE-GK: Predictive robust estimation using generalized kernels. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'16)*, pages 817–824, Stockholm, Sweden

- Peretroukhin, V., Clement, L., Giamou, M., and Kelly, J. (2015c). 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. (2015d). 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

Other learning

- Wagstaff, B., Peretroukhin, V., and Kelly, J. (2017). Improving foot-mounted inertial navigation through real-time motion classification. In *Proc. Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN'17)*, Sapporo, Japan

Bibliography

Agarwal, P., Tipaldi, G. D., Spinello, L., Stachniss, C., and Burgard, W. (2013a). Robust map optimization using dynamic covariance scaling. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 62–69.

Agarwal, P., Tipaldi, G. D., Spinello, L., Stachniss, C., and Burgard, W. (2013b). Robust map optimization using dynamic covariance scaling. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 62–69.

Alcantarilla, P. F. and Woodford, O. J. (2016). Noise models in feature-based stereo visual odometry.

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. Robot.*, 30(3):679–693.

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.

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.

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.

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.

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.

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014a). SVO: Fast semi-direct monocular visual odometry. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 15–22.

Forster, C., Pizzoli, M., and Scaramuzza, D. (2014b). SVO: Fast semi-direct monocular visual odometry. In *Proc. IEEE Int. Conf. Robot. Automat.(ICRA)*, pages 15–22. IEEE.

Gal, Y. and Ghahramani, Z. (2016). Bayesian convolutional neural networks with Bernoulli approximate variational inference. In *Proc. Int. Conf. Learning Representations (ICLR), Workshop Track*.

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. (2013a). Vision meets robotics: The KITTI dataset. *Int. Journal Robot. Research (IJRR)*.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013b). Vision meets robotics: The KITTI dataset. *Int. J. Rob. Res.*, 32(11):1231–1237.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Geiger, A., Ziegler, J., and Stiller, C. (2011). StereoScan: Dense 3D reconstruction in real-time. In *Proc. Intelligent Vehicles Symp. (IV)*, pages 963–968. IEEE.

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.

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.

Irani, M. and Anandan, P. (2000). About direct methods. In *Vision Algorithms: Theory and Practice*, pages 267–277. Springer.

Kendall, A., Alex, K., Matthew, G., and Roberto, C. (2015). PoseNet: A convolutional network for Real-Time 6-DOF camera relocalization. In *Proc. of IEEE Int. Conf. on Computer Vision (ICCV)*.

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.

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.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proc. Advances in Neural Inform. Process. Syst. (NIPS)*, pages 6405–6416.

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. (2015a). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Bengio, Y., and Hinton, G. (2015b). Deep learning. *Nature*, 521(7553):436–444.

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. (2017). Deep neural networks for improved, impromptu trajectory tracking of quadrotors. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 5183–5189.

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.

McClure, P. and Kriegeskorte, N. (2016). Representation of uncertainty in deep neural networks through sampling.

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.

Oliveira, G. L., Radwan, N., Burgard, W., and Brox, T. (2017). Topometric localization with deep learning.

Osband, I. (2017). Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout. In *Proc. of Workshop on "Bayesian Deep Learning", Advances in Neural Inform. Process. Syst. (NIPS)*.

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 *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Syst. (IROS)*, pages 3668–3675.

Peretroukhin, V., Clement, L., Giamou, M., and Kelly, J. (2015b). PROBE: Predictive robust estimation for visual-inertial navigation. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pages 3668–3675.

Peretroukhin, V., Clement, L., Giamou, M., and Kelly, J. (2015c). 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. (2015d). 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. (2017a). Reducing drift in visual odometry by inferring sun direction using a bayesian convolutional neural network. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*.

Peretroukhin, V., Clement, L., and Kelly, J. (2017b). 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*.

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

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. (2016a). PROBE-GK: Predictive robust estimation using generalized kernels. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 817–824.

Peretroukhin, V., Vega-Brown, W., Roy, N., and Kelly, J. (2016b). PROBE-GK: Predictive robust estimation using generalized kernels. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'16)*, pages 817–824, Stockholm, Sweden.

Punjani, A. and Abbeel, P. (2015). Deep learning helicopter dynamics models. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 3223–3230.

Scaramuzza, D. and Fraundorfer, F. (2011a). Visual odometry [tutorial]. *IEEE Robot. Automat. Mag.*, 18(4):80–92.

Scaramuzza, D. and Fraundorfer, F. (2011b). Visual odometry [tutorial]. *IEEE Robot. Autom. Mag.*, 18(4):80–92.

Sünderhauf, N. and Protzel, P. (2007). Stereo odometry: a review of approaches. *Chemnitz University of Technology Technical Report*.

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*.

Tsotsos, K., Chiuso, A., and Soatto, S. (2015a). Robust inference for visual-inertial sensor fusion. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 5203–5210.

Tsotsos, K., Chiuso, A., and Soatto, S. (2015b). Robust inference for visual-inertial sensor fusion. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 5203–5210.

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. (2014a). Nonparametric Bayesian inference on multivariate exponential families. In *Proc. Advances in Neural Information Proc. Syst. (NIPS) 27*, pages 2546–2554.

Vega-Brown, W. R., Doniec, M., and Roy, N. G. (2014b). Nonparametric bayesian inference on multivariate exponential families. In *Advances in Neural Information Processing Systems 27*, pages 2546–2554.

Wagstaff, B., Peretroukhin, V., and Kelly, J. (2017). Improving foot-mounted inertial navigation through real-time motion classification. In *Proc. Int. Conf. on Indoor Positioning and Indoor Navigation (IPIN'17)*, Sapporo, Japan.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. (2016). Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594.

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.

Zhang, G. and Vela, P. (2015). Optimally observable and minimal cardinality monocular SLAM. In *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, pages 5211–5218.