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

January 12, 2019

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. My thesis focuses on ways in which these two modelling efforts can be combined while still preserving the core advantages of both. To this end, I have published work on 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. More recently, I have leveraged both of these two prior efforts to create a framework I call Deep Pose Correction (DPC). DPC relies on a novel loss function, based on matrix Lie groups, that permits a deep network to learn SE(3) corrections to existing visual pose estimators. By pairing a DPC network with a visual localization pipeline, one can combine the proven durability of classical algorithms with the representational power of modern machine learning techniques.

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

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 [**?** **?** **?**] 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 [**?** **?**], 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 [**?**], CNN-based measurement models have been applied to numerous problems in geometric state estimation (e.g., homography estimation [?], single image depth reconstruction [?], camera re-localization [?], place recognition [?]). 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., [?], [?], [?]).

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

1.2.1 The limits of homoscedastic noise models

Although several state-of-the-art state estimation pipelines [? ?] leave observation uncertainty associated with sensor measurements as a static tuning parameter, recent work [? ?] 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 [? ? ?] and whiteness tests [?] 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 [?], 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 [? ? ?], and the statistical technique of *bootstrapping* has been applied to Deep Q Networks [?] to infer uncertainty, but both techniques are in their infancy. Recent work [?] 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 [? ?], others integrate deep networks within outer estimation loops [?]. Yet other work has used the neural network as an error correcting mechanism on top on an existing kinematic or dynamic model [?]. 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 [?] (PROBE-GK) to train a predictive model for heteroscedastic measurement covariance to improve estimator accuracy and consistency. PROBE [?] and PROBE-GK [?] were figures/introduction/dpc_high_level.pdf

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

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 [**?**], 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. 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 [**?**].

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 [**?**]. 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 [**?**], and in the context of heteroscedastic noise modelling in [**?**]; it is similar in principal to switchable constraints [**?**]).

Another potential alternative method is bootstrap aggregation (bagging). Bagging can be used to compute confidence bounds by training several models on randomly sub-sampled data (e.g., uncertainty for deep Q networks [**?**]). To reduce the cost of training multiple models, a number of authors have also suggested training a single network with multiple heads [**?**], [**?**]. I am investigating this latter approach, and also combining it with the log likelihood covariance learning.

Chapter 2

Predictive Robust Estimation

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:

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

The wedge operator $(\cdot)^\wedge$ is defined (following **?**]) 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:

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

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 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 Scalar k-Nearest Neighbours

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.

In this paper 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.

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

figures/probe/feature_space.pdf

Figure 2.1: 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.

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 (similar to [**?**]) 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 2.2.3 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.

2.2.1 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 2.2. 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.

```
1: procedure trainPROBEModel
2:     for $l \leftarrow 1, totalLearningIterations$ do
3:         for $s \leftarrow 1, totalPathSteps$ do
4:             $f_1, \ldots, f_J \leftarrow visualFeatureSubset(l)$
5:             $\boldsymbol{\pi}_l^1, \ldots, \boldsymbol{\pi}_l^J \leftarrow predictors(f_1, \ldots, f_J)$
6:             $\bar{\mathbf{C}}_{ba}, \bar{\mathbf{r}}_a^{ba} \leftarrow poseChange(f_1, \ldots, f_J)$
7:             $\alpha_{l,s} \leftarrow computeRMSE(\bar{\mathbf{r}}_a^{ba}, {\mathbf{r}_a^{ba}}_{GT})$
8:             $\Theta_{l,s} \leftarrow \left\{\boldsymbol{\pi}_{l,s}^1, \ldots, \boldsymbol{\pi}_{l,s}^J, \alpha_{l,s}\right\}$
9:         end for
10:    end for
11:    return $\boldsymbol{\Theta} = \{\Theta_{l,s}\}$
12: end procedure
```

Figure 2.2: The PROBE training procedure.

2.2.2 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.

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.

```
1: procedure usePROBEModel($\boldsymbol{\Theta}$)
2:     for $i \leftarrow 1, totalFeatures$ do
3:         $\boldsymbol{\pi}_i \leftarrow predictors(f_i)$
4:         $\alpha_1, ..., \alpha_K \leftarrow findKNN(\boldsymbol{\pi}_i, K, \boldsymbol{\Theta})$
5:         $\beta_i \leftarrow \left(\frac{1}{\bar{\alpha}K}\sum_{k=1}^{K}\alpha_k\right)^{\gamma}$
6:     end for
7:     return $\boldsymbol{\beta} = \{\beta_i\}$
8: end procedure
```

Figure 2.3: The PROBE evaluation procedure.

2.2.3 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 `figures/probe/tricifix2.jpg`

Figure 2.4: The Skybotix VI-Sensor, Point Grey Flea3, and checkerboard target used in our motion blur experiments.

`figures/probe/reprojectionError.pdf`

(a) Reprojection error of checkerboard corners triangulated from the VI-Sensor and reprojected into the Flea3.

`figures/probe/trackingError.pdf`

(b) Tracking error of KLT-tracked checkerboard corners in the left VI-Sensor camera compared to directly re-detected corners.

Figure 2.5: Effect of blur on reprojection and tracking error for the slow-then-fast checkerboard dataset. We distinguish between high and low blur by thresholding the blur metric [?]. The variance in both errors increases with blur.

predictor of its quality. We evaluate the entropy S in an image patch by sorting pixel intensities into N bins and computing

$$S = -\sum_{i=1}^{N} c_i \log_2(c_i), \tag{2.12}$$

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

We compute this optical flow variance score according to

$$\log\left(\frac{\bar{\sigma}_s^2}{\bar{\sigma}_l^2}\right), \tag{2.13}$$

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 figures/probe/flowPredictorCombined.png

Figure 2.6: The optical flow variance predictor can help in detecting moving objects. Red circles correspond to higher values of the optical flow variance score (i.e., features more likely to belong to a moving object).

figures/probe/highFreqPredictor.png

Figure 2.7: A high-frequency predictor can distinguish between regions of high and low texture such as foliage and shadows. Green indicates higher values.

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).

2.3 Generalized Kernels

However, not all features are created equal; most feature-based methods rely on random sample consensus algorithms [**?**] to partition the extracted features into inliers and outliers, and perform estimation based only on inliers. It is common to guard against misclassifying an outlier as an inlier by using robust estimation techniques, such as the Cauchy costs employed in **?**] or the dynamic covariance scaling devised by **?**]. 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

figures/probe/city-res-road.png

Figure 2.8: 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.

Table 2.1: 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)	
Trial	Type	Path Length	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
`26_drive_0104`	City [1]	245.1 m	0.977 m	4.43 m	0.850 m	3.46 m
`29_drive_0071`	City [1]	234.0 m	5.44 m	30.3 m	5.44 m	30.4 m
`26_drive_0117`	City [1]	322.5 m	2.29 m	9.07 m	2.29 m	9.07 m
`30_drive_0027`	Residential [1, †]	667.8 m	4.22 m	12.2 m	4.30 m	10.6 m
`26_drive_0022`	Residential [2]	515.3 m	2.21 m	3.99 m	2.66 m	6.09 m
`26_drive_0023`	Residential [2]	410.8 m	1.64 m	8.20 m	1.77 m	8.27 m
`26_drive_0027`	Road [3]	339.9 m	1.63 m	8.75 m	1.63 m	8.65 m
`26_drive_0028`	Road [3]	777.5 m	4.31 m	16.9 m	3.72 m	13.1 m
`30_drive_0016`	Road [3]	405.0 m	4.56 m	19.5 m	3.33 m	14.6 m
UTIAS Outdoor	Snowy parking lot	302.0 m	7.24 m	10.1 m	7.02 m	10.6 m
UTIAS Indoor	Lab interior	32.83 m	—	0.854 m	—	0.738 m

[1] Trained using sequence `09_26_drive_0005`. [2] Trained using sequence `09_26_drive_0046`. [3] Trained using sequence
[†] This residential trial was evaluated with a model trained on a sequence from the city category because of several mo
better represented in that training dataset.

`figures/probe/husky2.jpg`

Figure 2.9: Our four-wheeled skid-steered Clearpath Husky rover equipped with Skybotix VI-Sensor and Ashtech DGPS antenna used to collect the outdoor UTIAS dataset.

figures/probe-gk/system_overview.pdf

Figure 2.10: 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.

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 [?] on predictive robust estimation (PROBE) with our work on covariance estimation [?] 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 to produce more accurate odometry estimates. Our pipeline is outlined in Figure 2.10.

2.3.1 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, [**?**]). 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 $\nu \in \mathbb{R} > d-1$:

$$\begin{aligned} p\left(\mathbf{R}\right) &= \mathrm{IW}\left(\mathbf{R}; \boldsymbol{\Psi}, \nu\right) \\ &= \frac{|\boldsymbol{\Psi}|^{\nu/2}}{2^{\frac{\nu d}{2}}\Gamma_d(\frac{\nu}{2})} |\mathbf{R}|^{-\frac{\nu+d+1}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}\left(\boldsymbol{\Psi}\mathbf{R}^{-1}\right)\right). \end{aligned} \tag{2.14}$$

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

Given a sequence of observations and ground truth transformations,

$$\mathcal{D} = \{\mathcal{I}_t, \mathbf{T}_t\}, \quad t \in [1, N] \tag{2.15}$$

where

$$\mathcal{I}_t = \{\mathbf{y}_{i,t}, \mathbf{y}'_{i,t}, \boldsymbol{\phi}_{i,t}\} \quad i \in [1, N_t], \tag{2.16}$$

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

$$\begin{aligned} p(\mathbf{R}_*|\mathcal{D}, \boldsymbol{\phi}_*) &\propto \prod_{i,t} \mathcal{N}(\mathbf{e}_{i,t}|\mathbf{0}, \mathbf{R}_*)^{k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t})} \\ &\qquad \times \mathrm{IW}(\mathbf{R}_*; \boldsymbol{\Psi}(\boldsymbol{\phi}_*), \nu(\boldsymbol{\phi}_*)) \quad (2.17) \\ &= \mathrm{IW}(\mathbf{R}_*; \boldsymbol{\Psi}_*, \nu_*). \quad (2.18) \end{aligned}$$

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

$$\boldsymbol{\Psi}_* = \boldsymbol{\Psi}(\boldsymbol{\phi}_*) + \sum_{i,t} k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t}) \mathbf{e}_{i,t} \, \mathbf{e}_{i,t}{}^T , \tag{2.19}$$

$$\nu_* = \nu(\boldsymbol{\phi}_*) + \sum_{i,t} k(\boldsymbol{\phi}_*, \boldsymbol{\phi}_{i,t}). \tag{2.20}$$

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

$$\begin{aligned} & p(\mathbf{y}'_{i,t} | \mathbf{T}_t, \mathbf{y}_{i,t}, \mathcal{D}, \boldsymbol{\phi}_{i,t}) && (2.21) \\ & = \int \mathrm{d}\mathbf{R}_{i,t} \mathcal{N}(\mathbf{e}_{i,t}; \mathbf{0}, \mathbf{R}_{i,t}) \, \mathrm{IW}(\mathbf{R}_{i,t}; \boldsymbol{\Psi}_*, \nu_*) && (2.22) \\ & = \mathrm{t}_{\nu_* - d + 1} \left(\mathbf{e}_{i,t}; \mathbf{0}, \frac{1}{\nu_* - d + 1} \boldsymbol{\Psi}_* \right) && (2.23) \\ & = \frac{\Gamma(\frac{\nu_*+1}{2})}{\Gamma(\frac{\nu_*-d+1}{2})} |\boldsymbol{\Psi}_{\boldsymbol{*}}|^{-\frac{1}{2}} \pi^{-\frac{d}{2}} \left(1 + \mathbf{e}_{i,t}{}^T \, \boldsymbol{\Psi}_{\boldsymbol{*}}{}^{-1} \mathbf{e}_{i,t} \right)^{-\frac{\nu_*+1}{2}} . && (2.24) \end{aligned}$$

Given a new landmark and predictor vector, we can infer a noise model by evaluating eqs. (2.19) and (2.20). In order to accelerate this computation, it is helpful to choose a kernel function with finite support: that is, $k(\boldsymbol{\phi}, \boldsymbol{\phi}') = 0$ if $\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|_2 > \rho$. 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. (2.19) and (2.20) in $\mathcal{O}(\log N + \log N_t)$ time. Algorithm 1 describes the procedure for building this model.

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

```
function BuildCovarianceModel($\mathcal{D}$)
    Initialize an empty spatial index $\mathcal{M}$
    for all $\mathcal{I}_t, \mathbf{T}_t$ in $\mathcal{D}$ do
        for all $\{\mathbf{y}_{i,t}, \mathbf{y}'_{i,t}, \boldsymbol{\phi}_{i,t}\}$ in $\mathcal{I}_t$ do
            $\mathbf{e}_{i,t} = \mathbf{y}'_{i,t} - f(\mathbf{T}_t f^{-1}(\mathbf{y}_{i,t}))$
            Insert $\boldsymbol{\phi}_{i,t}$ into $\mathcal{M}$ and store $\mathbf{e}_{i,t}$ at its location
        end for
    end for
    return $\mathcal{M}$
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

$$\mathbf{T}_t^* = \underset{\mathbf{T}_t \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} (\nu_{i,t}+1) \log\left(1 + {\mathbf{e}_{i,t}}^T \, \boldsymbol{\Psi}_{i,t}^{-1} \mathbf{e}_{i,t}\right). \tag{2.25}$$

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

Solving nonlinear optimization problems with the form of Equation (2.25) is a well-studied and well-understood task, and software packages to perform this computation are readily available. Algorithm 2 describes the procedure for computing the transform between a new image pair, treating the optimization of Equation (2.25) as a subroutine.

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

```
function ComputeTransform($\mathcal{I}_t$, $\mathcal{M}$)
    for all $\{\mathbf{y}_{i,t}, \mathbf{y}'_{i,t}, \boldsymbol{\phi}_{i,t}\}$ 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($\mathcal{M}, \boldsymbol{\phi}_*, \rho$)
                                        ▷ $\rho$ 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
```

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.

2.3.2 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 [**?**] 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:

$$Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)}) = \int \left(\prod_{i,t} \mathrm{d}\mathbf{R}_{i,t}\, p(\mathbf{R}_{i,t}|\mathcal{D}_{\setminus i,t}, \mathbf{T}_{1:T}^{(n)}) \right) \times \log \prod_{i,t} p(\mathbf{y}'_{i,t}|\mathbf{y}_{i,t}, \mathbf{T}_t, \mathbf{R}_{i,t}). \quad (2.26)$$

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:

$$Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)}) \cong -\frac{1}{2} \sum_{t=1}^{T} \sum_{i=1}^{N_t} \mathbf{e}_{i,t}{}^T \left(\frac{1}{\nu_{i,t}^{(n)}} \mathbf{\Psi}_{i,t}^{(n)} \right)^{-1} \mathbf{e}_{i,t}. \quad (2.27)$$

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

$$\mathbf{T}_{1:T}^{(n+1)} = \underset{\mathbf{T}_{1:T} \in \mathrm{SE}(3)^T}{\operatorname{argmax}}\; Q(\mathbf{T}_{1:T}|\mathbf{T}_{1:T}^{(n)}). \quad (2.28)$$

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

$$\mathbf{T}_t^{(n+1)} = \underset{\mathbf{T}_t \in \mathrm{SE}(3)}{\operatorname{argmin}} \sum_{i=1}^{N_t} \mathbf{e}_{i,t}{}^T \left(\frac{1}{\nu_{i,t}^{(n)}} \mathbf{\Psi}_{i,t}^{(n)} \right)^{-1} \mathbf{e}_{i,t}. \tag{2.29}$$

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 [**?**]. It is also possible to use a robust loss function (Equation (2.25)) in place of Equation (2.29) 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
                $\mathbf{\Psi}, \nu \leftarrow$ InferNoiseModel($\mathcal{M}, \boldsymbol{\phi}_{i,t}$)
                $g(\mathbf{T}) = \mathbf{y}_{i,t} - f(\mathbf{T} f^{-1}(\mathbf{y}'_{i,t}))$
                $\mathcal{L} \leftarrow \mathcal{L} + g(\mathbf{T})^T \left(\frac{1}{\nu}\mathbf{\Psi}\right)^{-1} g(\mathbf{T})$
            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
```

2.3.3 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 `figures/probe-gk/sim_world.pdf`

Figure 2.11: Our synthetic world. A stereo camera rig moves through a world with 2000 point features.

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 2.12 and Table 2.2 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 [**?**]). These benchmarks served as baseline estimators (with and without robust costs) that used fixed covariance matrices and did not include a predictive component.

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

2.3.4 KITTI

To evaluate PROBE-GK on real environments, we trained and tested several models on the KITTI Vision Benchmark suite [**?** **?**], 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, figures/probe-gk/simComparison_sim_rerun.pdf

Figure 2.12: 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 2.2.

figures/probe-gk/KittiEnvironments.pdf

Figure 2.13: 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.

figures/probe-gk/kittiComparison-0011-0009-15-Sep-2015-4way.pdf

Figure 2.14: RMSE comparison of stereo odometry estimators evaluated on data from the city category in the KITTI dataset. See Table 2.2 for a quantitative summary.

figures/probe-gk/kittiComparison-0035-0036-15-Sep-2015-4way.pdf

Figure 2.15: RMSE comparison of stereo odometry estimators evaluated on data from the residential category in the KITTI dataset. See Table 2.2 for a quantitative summary.

we focused on the city, residential and road categories (Figure 2.13). From each category, we chose two separate trials for training and testing.

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 [**?**].

Figures 2.14 to 2.16 show typical results; Table 2.2 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 figures/probe-gk/kittiComparison-0070-0028-15-Sep-2015-4way.pdf

Figure 2.16: RMSE comparison of stereo odometry estimators evaluated on data from the road category in the KITTI dataset. See Table 2.2 for a quantitative summary.

Table 2.2: 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	
	Length [m]	Fixed Covar.	Static M-Estimator	GK-GT	GK-EM	Fixed Covar.	Static M-Estima
Synthetic	180	3.87	2.49	1.59	1.66	0.18	0.13
City	332.9	3.84	2.99	1.69	2.87	0.032	0.021
Residential	714.1	13.48	9.37	1.97	8.80	0.068	0.050
Road	723.8	17.69	9.38	5.24	8.87	0.060	0.027

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.

UTIAS

To further investigate the capability of our EM approach, we evaluated PROBE-GK on experimental data collected at the University of Toronto Institute for Aerospace Studies (UTIAS). For this experiment, we drove a Clearpath Husky rover outfitted with an Ashtech DG14 Differential GPS, and a PointGrey XB3 stereo camera around the MarsDome (an indoor Mars analog testing environment) at UTIAS (Figure 2.17) 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 figures/probe-gk/experiment_wide_border.pdf

Figure 2.17: Our experimental apparatus: a Clearpath Husky rover outfitted with a PointGrey XB3 stereo camera and a differential GPS receiver and base station.

figures/probe-gk/xb3_rtk_fig.pdf

Figure 2.18: GPS ground truth for 5 experimental trials collected near the UTIAS Mars Dome. Each trial is approximately 250 m long.

Table 2.3: 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

figures/probe-gk/trainingStats.pdf

Figure 2.19: 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.

stereo camera to record the stereo images. The approximate trajectory for all 5 trials, as recorded by GPS, is shown in Figure 2.18. Note that the GPS data was not used during training, and only recorded for reference.

For the prediction space in our experiments, we mimicked the KITTI experiments, omitting inertial magnitudes as no inertial data was available. We trained PROBE-GK without ground truth, using the Expectation Maximization approach. Figure 2.19 shows the likelihood and loop closure error as a function of EM iteration.

The EM approach indeed produced significant error reductions on the training dataset after just a few iterations. Although it was trained with no ground truth information, our PROBE-GK model was used to produce significant reductions in the loop closure errors of the remaining 4 test trials. This reinforced our earlier hypothesis: the EM method works well when the training trajectory more closely resembles the test trials (as was the case in this experiment). Table 2.3 lists the statistics for each test.

Chapter 3

Sun-BCNN

3.1 Introduction

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 [**?**], 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.

figures/sun-bcnn/system_overview.pdf

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

figures/sun-bcnn/all_datasets_gps.pdf

Figure 3.2: 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.)

In this work, we seek to answer the question of whether similar reductions in VO drift can be obtained solely from the image stream already being used to compute VO. The main idea here is that by reasoning over more than just the geometric information available from a standard RGB camera, we can improve existing VO techniques without needing to rely on a dedicated sun sensor or specially oriented camera. In particular, we leverage recent advances in Bayesian Convolutional Neural Networks (BCNNs) to demonstrate how we can build and train a deep model capable of inferring the direction of the sun from a single RGB image. Moreover, we show that our network, 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 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

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

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.

Bibliography